Mikko-Jussi Laakso

Promoting Programming Learning

Engagement, Automatic Assessment with Immediate Feedback in Visualizations

Turku Centre *for* Computer Science

TUCS Dissertations
No 131,November 2010

Promoting Programming Learning

Engagement, Automatic Assessment with Immediate Feedback in Visualizations

Mikko-Jussi Laakso

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in the lecture hall Beta in the ICT building on 11.12.2010 at 12 o'clock.

University of Turku
Turku Centre for Computer Science (TUCS)
Department of Information Technology
Joukahaisenkatu 3–5, 20520 Turku

2010

SUPERVISORS

Professor **Tapio Salakoski**

Department of Information Technology

University of Turku

Finland

Professor **Lauri Malmi** and Docent **Ari Korhonen**

Department of Computer Science and Engineering

Aalto University

Finland

REVIEWERS

Professor **Tom Naps**

Department of Computer Science

University of Wisconsin – Oshkosh

United States of America

Dr **Arnolds Pears**

Department of Computer Systems

University of Uppsala

Sweden

OPPONENT

Professor **Erkki Sutinen**

School of Computing

University of Eastern Finland

Finland

ISBN 978-952-12-2485-0 (printed)
ISBN 978-952-12-2486-7 (electronic)
ISSN 1239-1883

Abstract

The skill of programming is a key asset for every computer science student. Many studies have shown that this is a hard skill to learn and the outcomes of programming courses have often been substandard. Thus, a range of methods and tools have been developed to assist students' learning processes. One of the biggest fields in computer science education is the use of visualizations as a learning aid and many visualization based tools have been developed to aid the learning process during last few decades.

Studies conducted in this thesis focus on two different visualization-based tools TRAKLA2 and ViLLE. This thesis includes results from multiple empirical studies about what kind of effects the introduction and usage of these tools have on students' opinions and performance, and what kind of implications there are from a teacher's point of view.

The results from studies in this thesis show that students preferred to do web-based exercises, and felt that those exercises contributed to their learning. The usage of the tool motivated students to work harder during their course, which was shown in overall course performance and drop-out statistics.

We have also shown that visualization-based tools can be used to enhance the learning process, and one of the key factors is the higher and active level of engagement (see. Engagement Taxonomy by Naps et al., 2002). The automatic grading accompanied with immediate feedback helps students to overcome obstacles during the learning process, and to grasp the key element in the learning task.

These kinds of tools can help us to cope with the fact that many programming courses are overcrowded with limited teaching resources. These tools allows us to tackle this problem by utilizing automatic assessment in exercises that are most suitable to be done in the web (like tracing and simulation) since its supports students' independent learning regardless of time and place.

In summary, we can use our course's resources more efficiently to increase the quality of the learning experience of the students and the

teaching experience of the teacher, and even increase performance of the students.

There are also methodological results from this thesis which contribute to developing insight into the conduct of empirical evaluations of new tools or techniques. When we evaluate a new tool, especially one accompanied with visualization, we need to give a proper introduction to it and to the graphical notation used by tool. The standard procedure should also include capturing the screen with audio to confirm that the participants of the experiment are doing what they are supposed to do. By taken such measures in the study of the learning impact of visualization support for learning, we can avoid drawing false conclusion from our experiments.

As computer science educators, we face two important challenges. Firstly, we need to start to deliver the message in our own institution and all over the world about the new – scientifically proven – innovations in teaching like TRAKLA2 and ViLLE. Secondly, we have the relevant experience of conducting teaching related experiment, and thus we can support our colleagues to learn essential know-how of the research based improvement of their teaching. This change can transform academic teaching into publications and by utilizing this approach we can significantly increase the adoption of the new tools and techniques, and overall increase the knowledge of best-practices.

In future, we need to combine our forces and tackle these universal and common problems together by creating multi-national and multi-institutional research projects. We need to create a community and a platform in which we can share these best practices and at the same time conduct multi-national research projects easily.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Tapio Salakoski, for his patience and continuous support during this learning process and to help me reach this goal. Thank you, Tapio, for your valuable guidance and support in those moments when I was unsure and hesitated. You have a great ability to guide me quickly back to the right track.

I am also very grateful to Professor Lauri Malmi and Docent Ari Korhonen for their excellent guidance and collaboration during this process. I was able to join the Software Visualization Group at the beginning of my PhD mission, which fast-tracked the start of my PhD-studies. Together with Ari and Lauri, I have conducted many experiments and produced many co-authored publications. It has been a pleasure to work with you both.

I am grateful to have had highly talented academics, Professor Tom Naps and Dr Arnold Pears, as reviewers for my PhD-thesis. Thank you for your valuable and encouraging comments about my work. I would also like to give my warm regards to Professor Erkki Sutinen who agreed to act as the opponent of my thesis. In addition, I am grateful to Dr Judy Sheard for the guidance and important feedback in relation to finishing and polishing up my thesis during my visit to Monash University.

Team work is an essential skill in all aspect of life as it is the case for this thesis as well. I have had the privilege to work with extremely

talented and dedicated people in many different research projects. Teemu Rajala, Erkki Kaila and I have executed many different research projects with success. Teemu and Erkki worked long hours and did extraordinary work when it was needed. It has been a great pleasure to work with you both. In addition, I would like to thank Dr Niko Myller and Dr Linda Mannila for our co-authored work. It has been a great experience to work with you.

Many PhD-theses in computer science education involve carrying out experiments in practical courses. I would like to thank Docent Jouni Järvinen and Mia Peltomäki, who have been extremely flexible and easy to work with in allowing different experiments to be conducted in their courses.

I would like to express my gratitude to many different institutions and foundations who have financially supported this process like Turku Centre for Computer Science (TUCS), Network project on basic programming studies to promote best practices and its partners: Department of Information Technology at University of Turku, Department of Computer Science and Engineering at Aalto University, and Department of Computer Science at University of Tampere. In addition, the last phase included visits to Monash University and the University of Melbourne which were supported by TUCS, Nokia Foundation and Turku University Foundation, and both universities in Melbourne.

I would like to thank many different people in these organizations including Director Hannu Tenhunen and administrative people Irmeli Laine and Tomi Mäntylä at TUCS, the administrative people at

University of Turku: Maarit Pöyhönen, Maria Prusila, Päivi Rastas and Heli Vilhonen, Dr Angela Carbone at Monash University, and Professor Roger Hagraft and Dr Linda Stern at University of Melbourne. In addition, there are many people in the aforementioned organizations that have influenced my work in some way or another like Dr Jorma Boberg, Professor Jouni Isoaho, Dr Kai Kimppa, Professor Timo Knuutila, Dr Ville Leppänen, Professor Olli Nevalainen, Dr Antti Tuomisto, Dr Seppo Virtanen at University of Turku, and other colleagues in many different organizations in Finland including Professor Hannu-Matti Järvinen, Dr Ville Karavirta, Dr Vesa Lappalainen, Dr Uolevi Nikula, Johanna Olson, Piia Räsänen, and all the people of the LeTech-group at Aalto University.

I would like to say warm thanks to all my friends in Finland and in Australia. My family is grateful to Anja and Esko for their support during our visit in Australia, and to my wife's parents (Sirkka and Markku) and their families for their support in this process.

I am extremely grateful to my parents Marja-Leena and Veikko for their continuous support, encouragement and life-guidance during this learning process and throughout my life. Dad, now it is your turn to complete your PhD-thesis.

Most of all, I would like to thank my lovely family and it is to them that I dedicate this work. Tiila, Oona, Onni-Eemeli, and Fiona, you are the centre of my life, and you are making my life extraordinary and worthwhile.

October 2010, Melbourne

Mikko-Jussi Laakso

To my lovely family

Contents

List of original publications included in the thesis

P1. Laakso, M.-J., Salakoski, T., Korhonen, A., and Malmi, L. (2004). Automatic assessment of exercises for algorithms and data structures - a case study with TRAKLA2. In Proceedings of Kolin Kolistelut/Koli Calling---Fourth Finnish/Baltic Sea Conference on Computer Science Education. Helsinki University of Technology, 28-36.

P2. 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.

P3. Laakso, M.-J., Salakoski, T., and Korhonen, A. (2005). The feasibility of automatic assessment and feedback. Proceedings of Cognition and Exploratory Learning in Digital Age (CELDA), Lisbon: IADIS Press, 113–122.

P4. Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Journal of Educational Technology and Society. 12(2), 267–282.

P5. Rajala, T., Laakso, M.-J., Kaila, E. and Salakoski, T. (2008). Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. Journal of Information Technology Education: Innovations in Practice, 7, IIP 15-32.

P6. Laakso, M.-J., Rajala, T., Kaila, E. and Salakoski, T. (2008). The Impact of Prior Experience in Using a Visualization Tool on Learning to Program. Proceedings of CELDA 2008, Freiburg, Germany, 129-136.

Co-authored publications not included in this thesis

- Kaila, E., Rajala, T., Laakso M.-J., Salakoski, T. (2010). Effects of course-long use of a program visualization tool. In Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103 (Brisbane, Australia, January 01 - 01, 2010). T. Clear and J. Hamer, Eds. Conferences in Research and Practice in Information Technology Series. Australian Computer Society, Darlinghurst, Australia, 97-106.

- Chinn, D., Sheard, J., Carbone, A., Laakso M.-J. (2010). Study habits of CS1 students: what do they do outside the classroom? In Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103 (Brisbane, Australia, January 01 - 01, 2010). T. Clear and J. Hamer, Eds. Conferences in Research and Practice in Information Technology Series. Australian Computer Society, Darlinghurst, Australia, 53-62.

- Rajala, T., Salakoski, T., Kaila, E., Laakso, M.-J. (2010). How does collaboration affect algorithm learning? A case study using TRAKLA2 algorithm visualization tool, in proceedings of the Education Technology and Computer (ICETC), 2nd International Conference on vol.3, pp.V3-504-V3-508, 22-24, June. doi: 10.1109/ICETC.2010.5529489

- Rajala, T., Kaila, E., Laakso, M.-J. Salakoski, T. (2009) Effects of Collaboration in Program Visualization. Appeared in the proceedings of the Technology Enhanced Learning Conference, TELearn 2009, Taipei, Taiwan.

- Kaila, E., Rajala, T., Laakso, M.-J. and Salakoski, T. (2009). Effects, Experiences and Feedback from Studies of a Program Visualization Tool. Informatics in Education, 8(1):17–34.
- Korhonen, A., Laakso, M.-J., and Myller, N. (2009). How does algorithm visualization affect collaboration? Video Analysis of Engagement and Discussions. In: Joaquim Filipe and José Cordeiro eds. Proceedings of the 5th International Conference on Web Information Systems and Technologies. INSTICC — Institute for Systems and Technologies of Information, Control and Communication, WEBIST 2009, 23-26 March, Lisboa, Portugal, pp. 479–488.
- Kaila, E., Rajala, T., Laakso, M.-J. & Salakoski, T. (2008). Automatic Assessment of Program Visualization Exercises. In Proceedings of the 8th international Conference on Computing Education Research (Koli, Finland, November 13 - 16, 2008). Koli '08. ACM, New York, NY, 101-104. DOI= http://doi.acm.org/10.1145/1595356.1595376.
- Laakso, M.-J., Malmi, L., Korhonen, A., Rajala, T., Kaila, E., and Salakoski, T. (2008). Using roles of variables to enhance novice's debugging work. Issues in Informing Science and Information Technology, 5, 281-294.
- Laakso, M-J., Kaila, E., Rajala, T. & Salakoski, T. (2008). Define and Visualize Your First Programming Language. In Proceedings of ICALT 2008 - the 8th IEEE International Conference on Advanced Learning Technologies. July 1st - July 5th, 2008. Santander, Cantabria, Spain.
- 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

- Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2007). VILLE - A language-independent program visualization tool. Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, Vol. 88, Australian Computer Society. Raymond Lister and Simon, Eds.

- Mäntylä, T., Laakso, M.-J., Innola, E. and Salakoski, T. (2005). Student counselling with the SOPS-tool. Proceedings of the Fifth Koli Calling Conference on Computer Science Education.

1 Introduction

Programming is a complex and a cognitively demanding task. Many students are struggling with the first steps of learning to program and this phenomenon has been reported in many studies all over the world. Students have difficulties with reading, tracing, writing and designing simple code fragments. The following studies will give insights into this phenomenon.

The study conducted by McCracken's working group (McCracken et al., 2001) included four universities with total of 216 students. The average scored points was only about 23 points out of a maximum 110 points, and the results indicated a great number of failures in simple programming tasks where students were asked to produce a short piece of program code.

In addition, Lister et al. (2004) conducted a research study related to tracing and understanding the execution of simple programs. They assessed mostly students who had completed (or nearly completed) their first programming course. The study included 556 students in 12 different institutions. In that study, there were two types of tasks: students were asked to predict the outcome of small program i.e. tracing the code, and to select the right choice from given alternatives to complete a small code fragment. This study concluded that students had serious problems understanding the execution of small programs and students were especially weak in the code-completion task which is a necessary pre-requisite for problem solving.

Tenenberg et al. (2005) examined students' ability to design software with over 300 participants from 21 institutions in four countries. They concluded that students cannot even design small programs after an introductory course.

To summarize this so far, we have a universal problem in that students are struggling to obtain the basic skills of programming. They cannot even create, design and implement simple programs and one clear outcome of this is that the results of programming courses have been substandard i.e. students are dropping out from their introductory programming courses (see Bennedsen and Caspersen, 2007).

During the last few decades, this has lead to development of many different kinds of methods, tools and techniques to assist students' learning processes in programming education and almost a half of the publications in the Computer Science Education (CSE) field have some relation to this issue.

One of the biggest research fields is the use of visualization as a learning aid. There are numerous systems developed that use graphical components in explaining and clarifying the dynamic nature of programs. The challenge is even greater, when we combine the use of tools with the fact that many introductory programming and algorithm courses are overcrowded with hundreds of students. In this kind of course, we cannot provide extensive personal guidance with limited resources, so it becomes necessary that we produce systems that are capable of visualizing, assessing automatically and giving immediate feedback. In other words, these systems support students' independent learning in one way or another.

In addition, there are also many other approaches to assist students in learning to program which have some connections to the topic of this thesis. Interested reader can find more information about approaches like collaborative learning, pair-programming and blended learning, and theories related to programming learning; why it is hard, what kind of skills are needed to master it, what should be taken into account when teaching and assessing introductory programming courses. (Daniels et al., 2004, du Boulay, 1989, Milne and Rowe, 2002, Boyle et al., 2003, Nagappan et al., 2003, Pears et al., 2005, Pears et al., 2007, Robins et al., 2003, Eckerdal et al., 2005, McGettrick et al., 2005, Lahtinen et al., 2005, Jain et al., 2006).

In this thesis, I focus on researching issues relating to the use of visualization tools as learning aids and how we can utilize these tools effectively in the different learning settings. The thesis also investigates what implications these tools have from the student and teacher points of view.

1.1 Research questions

The research reported, and experiments conducted, in this thesis focus on programming learning using visualization-based tools. By programming learning I mean courses ranging from introductory programming courses to data structures and algorithm courses. The main motivation for conducting these experiments was to tackle the problem of teaching the large student population. Since the first programming courses are usually overcrowded, with many students and with limited resources we need to develop methods and tools to support the provision of resource efficient ways of teaching.

The first research goal is to understand the effect of the introduction of a new tool for programming learning from the student perspective.

RQ1: *What kinds of effects there are when we adopt web-based tools in our teaching and learning?*

This question is quite large and it can be divided into four sub-questions in order to fully understand the changes in the student's learning process; what are the students' opinions and attitudes towards a new tool, what is the effect of a new tool on the students' performance, and what is the role of automatic assessment and immediate feedback in achieving changes in students' learning performance.

RQ1.1: *What are the students' perceptions and attitudes towards web-based learning?*

RQ1.2: *How does the usage of web-based tools affects students' performance?*

RQ1.3: *Are web-based tools TRAKLA2 and ViLLE effective?*

RQ1.4: *What are the roles of automatic assessment, user engagement and immediate feedback in enhancing students' learning of programming?*

The second main research question is connected to the first one. The first main research question and its sub-questions covered issues relating to the student point of view, and the second research question tackles the process from the teacher point of view. In addition, we wanted to find out what kind of issues there are in relation to

conducting the evaluation of a new tool or a new method from a methodological stance.

RQ2: *How can we create and evaluate visualization-enhanced web-based tools that support programming learning?*

RQ2.1: *What implications are there from the teachers' point of view?*

RQ2.2: *What kind of issues should be taken into account when we evaluate any new tools and methods?*

In the papers **P1**, **P2**, **P3, P4, P5** and **P6** we study the effect of visualization tools from students' point of view. In addition, papers **P4** and **P6** present studies in relation to the research questions R2.2. All of the papers **P1-P6** contribute to the research question R2.1. The literature review is used to build the framework for each of these research questions in all of the papers and in this thesis as well.

1.2 The structure of this thesis

Section 2 presents a research framework related the topic of this thesis. In addition, section 3 presents an overview of the visualization tools used in this thesis. Section 4 summarizes the conducted experiments, and the result are revisited and discussed in section 5. Finally, the conclusions are drawn and future work is presented in section 6.

2 Software visualization

Ben-Ari (2001) has stated that visualization includes everything from indentation of program code to complex animations. In addition, he concluded that the effective use of visualizations requires that textual and graphical notations need to be connected and synchronized to each other. The goal of any visualization is to help the learners grasp the essential elements of a topic and to reveal any underlying relationships. Visualization can also be used to provide learning models for the learner to link new knowledge with old knowledge (Hyrskykari, 1993).

However, for novice learners it is usually hard to determine which parts of visualization are important and relevant in different states of the visualization (Ben-Ari, 2001, Petre and Green, 1993). The relevant parts should be highlighted by the visualization itself, but that still doesn't ensure that learner knows how to interpret these visualizations. In addition, Petre (1995) has stated that "the question is not 'Is a picture worth a thousand words?', but 'Does a picture convey the same thousand words to all viewers?". It can be easily seen, that this simple statement sets the requirements and challenges for visualizations. Different learners' needs must be fulfilled, and the level of visualization must be adapted to the level of knowledge of the learner.

One of the first visualizations is the movie about list manipulation using the L6 programming language (Knowlton, 1966). Still, perhaps the first widely recognized visualization is Ronald Baecker's (1981

and 1998) video *Sorting out Sorting* in which the goal is to illustrate the difference between nine basic sorting algorithms and especially emphasize the difference in the execution (complexity) time between different samples. After that many different forms of visualization have been created and the basis for the research field was set in late 1980's and 1990's.

Software visualization (SV) is the field of computer science which focuses on visualization of different aspects of software (programs, algorithms, classes, data flows etc). SV is defined by Price et al. (1993) and in Stasko et al. (1997):

> *"The use of the crafts of typography, graphic design, animation, and cinematography with modern human computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software."*

Moreover, the goal of SV is to aid a learner's understanding of issues related to software engineering and its processes. The definition of SV itself is very broad and therefore many different subfields have emerged, the two major subfields being the *program visualization* (PV) and the *algorithm visualization* (AV). In PV the goal is to promote understanding of dynamic behavior of program code by highlighting changes in the state of the program, and AV visualization can be seen as a higher-level representation of that same piece of code. Ben-Ari (2001) has stated that visualization can be low-level or high-level. In the low-level, the focus of the visualization is on displaying aspects like values of variables, evaluation of statements

and changes in program state in general. In the high-level, the visualization is focused on visualizing whole program modules, data structures and operations on those. Naturally, PV is usually counted as a low-level visualization and AV is counted as a high-level visualization. Although, it must be noted that the difference between algorithm visualization and program visualization systems is nowadays very narrow and there are systems which are capable of visualizing both the low-level and the high-level aspects of an algorithm.

There are many studies that have focused on aspects like the engagement of the learner and the relevance of each component which should be taken into account when we design and implement new visualizations-based tools (e.g. Stasko et al., 1993, Naps et al., 2002, Rößling and Naps, 2002a, Rößling and Naps, 2002b, 2002, Rößling and Häussage, 2004, Rößling et al., 2006, Brusilovsky et al., 2006).

2.1 Program visualization

As Wiggins (1998) has stated, the purpose of visualizations is to help the learner to understand what a program does, why it does it, how it works, and what the outcome is. Program visualization can be defined as the visualization of program or algorithm execution with graphical components. The goal of PV is to assist the learner in understanding the effects of program execution on variable state and the dynamic behaviour of a program. There are two main types of PV visualization systems: *static* and *dynamic*.

On the one hand, dynamic PV tools can illustrate the flow of control by highlighting the current program code line under execution and

illustrate the effects of the program execution on the program state (values of variables) by using visual components. On the other hand, static PV-tools focus on depicting program structure and its' relations with pictures and diagrams.

Still, there are numerous PV tools developed for learning purposes. BlueJ is a static program visualization tool (Kölling et al., 2003). JavaVis (Oechsle and Schmitt, 2002) visualizes program execution through object and sequence diagrams. JIVE (Gestwicki and Jayaraman, 2002) is a program visualization tool that in addition to code highlighting, visualizes object structure and method calling sequences. Jeliot 3 (Moreno et al., 2004) is a program visualization tool that illustrates program execution in Java with graphical symbols. JGRAPS combines the development environment with visualizations and can automatically identify some data structures (Cross and Hendrix, 2007 and Cross et al., 2007). Raptor (Carlisle et al., 2005) is a programming environment that uses dataflow diagrams for visualizations.

Many PV systems have been developed over the last years, but there are a just few quantitative studies which are focusing on the effects of the tool to the learning outcome (see e.g. Ben-Bassat Levy et al., 2002, Van Haaster and Hagan 2004). In addition, there are many qualitative research studies about programming education in PV (see Oudshoorn et al., 1996, Gestwicki and Jayaraman, 2002, Ben-Bassat Levy et al., 2002, Kannusmäki et al., 2004, Moreno and Joy, 2007). The qualitative research studies have shown that students felt that the PV system usually aided them in the learning process, especially in the early stages (see Kannusmäki et al., 2004).

2.2 Algoritm visualization

As mentioned earlier, the focus of AV is on a more abstract level than PV. Whilst PV visualizes changes in variable state, in AV the focus is on illustrating the effects of algorithm execution to the state of the data structures in hand. For example, in PV we would describe a binary tree with an array while in AV we would present the binary tree with a graphical tree-representation with nodes and edges. The latter gives much better mental model of the binary tree to the learner. This graphical representation gives the learner a better chance to learn the basics of a data structure or an algorithm. In AV, the learning goal is in a more conceptual level and it is important to learn the key elements of the data structure or the algorithm than to learn the low-level implementation of the data structure or algorithm.

AV can be divided further into two subcategories: *static algorithm visualization* and *algorithm animation*. The difference is that static AV uses different static components like diagrams, graphs, and pictures to visualize the relevant aspect of an algorithm, and algorithm animation includes everything related the dynamic visualization. The simplest form of dynamic visualization is the use of series of static pictures, and on the other end are the systems which are capable of interacting with the learner and direct manipulation of the visualization.

In the field of computer science education (CSE), many AV tools have been developed to ease the learning process. Some notable AV systems include ALVIS LIVE (Hundhausen and Brown, 2007), Animal (Rößling and Freisleben, 2002, BALSA-II (Brown, 1988),

JHAVÉ (Naps, 2005), TRAKLA2 (Malmi et al., 2004, Korhonen et al., 2004), XTANGO (Stasko, 1992), and ZEUS (Brown, 1991).

Visual algorithm simulation (VAS) was introduced by Korhonen (2003). This is a dynamic form of AV in which the learner uses a method enabling direct manipulation of the data structure in hand. This is accomplished by performing drag-and-drop operations while simulating (i.e. executing) a given algorithm with given data. One example of this is where the learner has to remove the minimum value from a minimum binary heap. In this exercise, the heap is visualized both in array- and/or in tree-representation and the learner can apply direct manipulations to one of the visualizations. Firstly, the learner swaps with a drag-and-drop operation the root of the heap and the last element of the heap. Secondly, the learner simulates a specific algorithm by performing a number of drag-and-drop operations (i.e. swaps) until the heap property is satisfied and the value has floated down to the correct position in the heap.

2.3 Use of visualization in computer science education

Although there have been many success stories in adapting visualization systems in teaching and learning (e.g. Korhonen et al., 2002, Hundhausen et al., 2002), there has not been widespread adoption of visualization tools by educators and instructors in their teaching. This phenomenon is reported in the ITiCSE working group report (Naps et al., 2002) and since then the situation has been remained quite the same. The report summarizes results from three different surveys related to usage of visualization-based tools in teaching and learning and attitudes of instructors in computer science

education. The main focus in that report was the instructor and the attitudes of the instructor whether towards the use of a new visualization in his teaching. In the academic world, the instructor of a course, in almost all cases, is the only person who decides whether or not to use new methods in his teaching. The report included almost 200 answers from about 20 – 30 countries.

In the first survey (namely pre-conference survey) there were total of 29 respondents, and 59% of respondents strongly agreed and 41% of agreed to the statement: "Using visualization can help students learn computing concepts". The same question was asked in the second survey (namely index card survey), and 43% strongly agreed, 49% agreed and 8 % didn't have opinion or were neutral. The total number of respondents was 66 in this second survey. There were almost 100 computer science educators who answered to this statement, and none of those disagreed with this statement. In addition, the report collected also opinions about benefits of using visualizations for the instructor, and the top benefits were:

- 90 %: the teaching experience is more enjoyable
- 86 %: improved level of student participation
- 83 %: anecdotal evidence that the class was more fun for students
- 76 %: visualization provide a powerful basis for discussing conceptual foundations of algorithms
- 76 %: anecdotal evidence of improved students learning

To conclude this section thus far, the report showed that instructors have very positive attitudes towards visualization, and its benefits for students and for themselves.

These attitudes are very positive, so how does this fact reflect in the usage of visualization in teaching? In the first survey, 97% of respondents did use visualizations in lectures occasionally and two-thirds made the visualizations available outside the classroom. Another survey (the third survey; namely Grissom survey) was conducted in 2000 and it was also reported in this WG report. This survey included a total of 91 answers, and there was question about the usage and frequencies of static and dynamic visualization inside and outside the classroom.

Almost three out of four respondents answered that they have frequently used static visualizations in teaching frequently. This is not surprising since a showing picture of program / algorithm state is counted as using a static visualization. An interesting finding was that 54% of respondents had used dynamic visualization a few times per term and 53% had used dynamic visualizations outside classroom also a few times per term. In addition, 13% had never used dynamic visualizations, and 23% had never used dynamic visualization outside the classroom. We can conclude that even if the instructor has positive attitudes towards visualizations, they are not utilizing visualizations frequently in teaching, since 67% of respondents did not use dynamic visualization frequently inside the classroom and even 76% did not use dynamic visualization frequently outside the classroom.

The surveys also collected reasons for this lack of use, which were:

- 93 %: time required to search for good examples
- 90 %: time it takes to learn the new tool
- 90 %: time it takes to develop visualizations
- 83 %: lack of effective development tools
- 79 %: time it takes to adapt visualizations to teaching

The most common factor for these reasons is the word *time* or more generally that the effort of introducing new visualization-based tools is too high. There are other studies which support these findings. For example, Hundhausen et al. (2002) studied the reasons why instructors are not using visualizations in teaching finding similar results.

In addition, Shaffer et al. (2007) found that most of the existing algorithm visualizations are of low quality, and the content of those visualizations are covering only the easier topics. They also noted that there are no repositories or collections of algorithm visualizations, although nowadays we have some portals like Algorithm Visualization Portal (AlgoViz, 2010).

2.4 Effectiveness of visualization

These studies indicate that intuitively people believe that visualizations will help the learner in the learning tasks. Even though there have been some quantitative and qualitative studies which have investigated the effectiveness of visualizations, there are still many open questions related to the effects of visualizations on learning outcomes and what are the relevant aspects of these.

One of the most recognized studies in the field of visualization in computer science education is the meta-study by Hundhausen et al. (2002). They conducted a study which analysed 24 educational experiments using visualizations. Only 46% (11 out of 24) of the studies analyzed reported statistically significant differences favouring the treatment group. In those cases the treatment group (utilizing some form of visualization) outperformed the control group (learning without the visualization) in learning performance. On the other hand, over the half of those studies did not find any difference in learning performance between the treatment and control group, and one of those experiments even reported an opposite result against the use of visualization. Hundhausen et al. (2002) reported that in many of those evaluated experiments the focus was on the number of visualized components instead of how those visualizations benefited students' learning. Hundhausen et al. (2002) concluded that the passive usage (viewing) of visualization does not guarantee better learning performance and it is really important to engage and activate the learner with visualization during the learning activity.

The importance of interaction between the learner and the material is reported also in the educational literature (see e.g. Evans and Gibbons, 2007, Mayer and Chandler, 2001, Mayer et al., 2003). In particular, this is reported in the research field *multimedia learning* (ML) which is defined by the use of multimedia accompanied with a learning target (Mayer, 2001). In addition, Mayer (2001) characterises ML learning from pictures and words or, more specifically, as *dual-channel learning* or *dual-code learning* (Paivio, 1986). Dual-channel learning means that we use both visual and auditive channels (eyes

and ears) in the learning process. The assumption of dual-code learning is that learning is more effective if the multimedia material is presented both in visual and verbal form. Dual-channel learning includes the *limited-capacity assumption* (Baddeley, 1992, Chandler and Sweller, 1991) which means that the learner can handle limited amounts of pieces of knowledge in both channels. Moreover, one key factor in this theory is the *cognitive load* of the material. The cognitive load is the number of element that we need to combine and integrate in order to learn a new element of knowledge (Lehtinen 2006, Mayer 2001, Sweller and Chandler 1994). More specifically, Lehtinen (2006) has identified a special form of cognitive load related to the learning environments. In other words, this is concerned with how the learning material is organized and represented. In addition, there are many more articles related to the ML and interaction (see Evans and Sabry, 2002, Moore, 1989, Schar and Krueger, 2000).

2.5 Engagement taxonomy

Findings in the meta-study by Hundhausen et al. (2002) revealed that we need to engage the learner with visualization to promote learning outcomes. This was one of the facts that lead to the development of the *Engagement Taxonomy* (ET, Naps et al., 2003). The taxonomy is a result of an ITiCSE workgroup in 2003 which was lead by Thomas Naps and Guido Rößling.

ET describes six levels of engagement between the learner and the visualization. In addition, it provides set of hypotheses about how the engagement affects learning outcomes. The ET gained acceptance in the CSE field very rapidly, and many different studies have been

carried out to prove its hypothesis. The central idea of the ET is the following: the higher the engagement between the learner and the visualization, the better the learning outcome. The six levels of the ET are presented in the Table 1.

Engagement level	**Description**
No-viewing	There is no visualization in use.
Viewing	The visualization is only viewed.
Responding	The learner interacts with the visualization by responding to the visualization related questions.
Changing	Visualization or state of visualization can be altered.
Constructing	The learner can create own visualizations.
Presenting	The learner presents visualizations for discussion and feedback.

Table 1 the levels of engagement taxonomy

No viewing is the first level of ET and it simply means that there is no visualization in use. Still, in this level it is possible to use textual based learning material without graphics such as pictures, etc.

Viewing is the next level of ET and it is the core form of engagement. This level of engagement exists also in every above level of this

taxonomy since all visualization systems include some kind of viewing. In this level, the learner can view the visualization passively or actively. The simplest example of *passive viewing* is watching an algorithm animation without controls like movie Sorting out Sorting (Baecker, 1981). The same animation could be counted as an *active viewing* if it is provided with VCR-like controls.

In the *responding* level the learner is asked questions about the visualization. The questions can be, for example, about predicting the next state of the visualization, asking the code behind the visualization, etc. There are many systems developed supporting this level of engagement like Animal (Rößling and Freisleben, 2002), JHAVÉ (Naps et al., 2000), ViLLE (Rajala et al., 2007) and TRAKLA2 (Malmi et al., 2004). In addition, there is a prototype version of question generator in Jeliot 3 (Myller, 2007).

In the *changing* level, the learner modifies the visualization. The system can allow the learner to vary the input data of the algorithm under visualization. This feature is found, for example, in ALVIE (Crescenzi and Nocentini, 2007), Alvis (Hundhausen and Douglas, 2002) and JHAVÉ (Naps et al., 2000). In addition, ViLLE (Rajala et al., 2007) and Jeliot 3 (Moreno et al., 2004) can be counted in this level because a learner can change the source code from which the visualization is generated.

The *Constructing* level includes activities in which the learner can create or construct his own visualizations of the algorithm. This can be done for example by direct manipulation, hand constructing or by visual algorithm simulation (Korhonen, 2003).

In TRAKLA2 (Malmi et al., 2004), PILOT (Bridgeman et al., 2000) and in MA&DA (Krebs et al., 2005) the learner is required to apply the visual algorithm simulation process to existing visualization or to create a visualization from scratch. It should be noted, that all of these aforementioned systems are capable of automatic evaluation and providing, to some extent, feedback of students' answers.

Other systems, in which there is a possibility to construct new visualization or animations, include ALVIE (Crescenzi and Nocentini, 2007), JHAVÉ (Naps et al., 2000) and WinHIPE (Pareja-Flores et al., 2007, Urquiza-Fuentes and Velázquez-Iturbide, 2007).

The highest level of the ET is *presenting*. In this level, the learner presents visualization to an audience for feedback or for discussion. All visualization systems can be utilized in this level and this can be accomplished by showing the user interface of the system through a data projector to the audience. However, there are systems which have integrated features which promote the use of the *presenting* level like on-the-fly features in Animal (Rößling and Ackermann, 2007) and in MatrixPro (Karavirta et al., 2002).

2.5.1 Evaluation of the engagement taxonomy

The engagement taxonomy has been tested in many studies over the last decade and a half. This section will present some of the studies in which the learning performance has been evaluated.

Lawrence et al. (1994) conducted a study related to the effects between *viewing*- and *changing*-level with 62 participants. The result indicated that students in the changing level got higher accuracy on examination compared to students in the viewing level, though the

difference was not statistically significant. In addition, there was also comparison between groups who attended the laboratory session (in *changing* or *viewing* –level) and those who did not attend. The students in the *changing* –level accompanied with the laboratory session-group outperformed the students in the no-laboratory-group with statistically significant difference.

In 1999, Byrne et al. (1999) reported two studies related to *no-viewing*, *viewing*, and *responding* –levels (*viewing* with prediction) with a total of 150 students. The results were not statistically significant, but they suggest that prediction was the key factor over the animation in the learning process, and they found a trend towards prediction i.e. *responding level*.

Hansen et al. (2000) presented a study which reported a summary from eight different experiments with over 230 participants. They compared the learning outcome between *viewing* and *changing* –levels. In their study, the *changing* –group outperformed the *viewing* group with statistically significant difference.

Jarc et al. (2000) carried out a study related to two experiments comparing the ET levels *viewing* and *responding* with 52 participants. There were no statistically significant differences in learning between the groups.

Kehoe et al. (2001) conducted a study in which they compared the learning outcome between *no viewing* and *viewing* –levels in a homework scenario with a total of 12 participants. There were some trends towards the *viewing* group, but students from both groups performed similarly in most of the questions.

Hübscher-Younger and Narayanan (2003) reported outcomes from three related experiments comparing *constructing* and *viewing*-levels with over 100 students. In addition, there is a collaboration aspect as well since the learners were able to evaluate and discuss the visualizations. The students improved their learning by creating and evaluating visualizations. In addition, the students working in the *constructing* level outperformed the others with statistically significant difference.

In 2003, Grissom et al. (2003) conducted a study comparing three different levels of ET (namely *no-viewing*, *viewing* and *responding)* with over 150 participants. The *responding* group outperformed the *viewing* (no statistical difference) and *no-viewing* group (with statistical significance), and viewing group outperformed no-viewing group (no statistical difference). These trends supported the hypotheses of the ET.

Rhodes et al. (2006) conducted a study in which they compared *viewing* and *responding* levels. They did not find statistically significant differences in learning. In addition, they found that questions with feedback were more successful than questions without the feedback (with statistical significance). However, the number of participants was quite low (29 participants divided into six groups), which suggest that these finding needs to be repeated with a larger number of participants.

Lauer (2006) carried out a study with 96 students comparing *viewing*, *changing* and *constructing* –levels of engagement. They did not find statistically significant differences in the learning performance

between the groups. The authors indicated that one possible reason for the outcome was the influence of an introductory lecture of the same topic before the experiment.

A study by Urquiza-Fuentes and Velázquez-Iturbide (2007) compared the engagement levels of *viewing* and *constructing* with a total of 15 participants. They found some evidence favoring the *constructing* group with statistically significant difference.

A study by Taylor et al. (2009) studied the difference between the learners using passive and predictive animations. In other words, they compared the ET levels *viewing* and *responding*. The results indicated that the students working in the *responding* group outperformed the *viewing* group. It must be noted, that the authors did not state if the difference has a statistical meaning or not.

In addition, there have been a couple more wide scale studies related to the effectiveness of visualizations in general. The first is the meta-study by Hundhausen et al. (2002) with mixed results, and the second is the study by Urquiza-Fuentes and Velázquez-Iturbide (2009). The latter included a survey about successful experiments about the use of visualization.

The next few paragraphs will conclude findings based on the existing research related to learning outcomes between the different engagement levels.

No-Viewing vs. Viewing.

At least, the passive *viewing* does not seem to predict higher learning performance than the *no-viewing* level (Hundhausen et al., 2002, Naps et al., 2002, Naps, 2005).

No-viewing vs. responding, changing, and constructing.

The research suggests, that visualization accompanied by some active form of engagement (levels: *responding*, *changing* and *constructing*) produces better learning outcomes when comparing active levels to the *no-viewing* level (Grissom et al., 2003, Hansen et al., 2000, and Hübscher-Younger and Narayanan, 2003) In addition, Lawrence et al. (1994) found also a trend favouring the changing group.

Viewing vs. responding.

The one of the most interesting situation is the difference between levels of *viewing* and *responding*. There are a couple of studies which do not show any differences in learning between these levels (see Jarc et al., 2000, Rhodes et al., 2006). However, there are also studies which are showing at least a trend towards the *responding* level (Byrne et al., 1999, Grissom et al., 2003, Taylor et al., 2009). In order to clarify whether there is a difference in the learning performance or not, further studies should be carried out which compare the levels *viewing* and *responding* with a decent number of participants.

Viewing vs. changing and *constructing*.

There are a number of large studies which have shown that there is a difference in learning performance between the *viewing* and *changing/ constructing* –level (Hansen et al., 2000, Hübscher-Younger and Narayanan, 2003). In addition, the study by Lauer (2006) did not find any differences in learning between these levels, but there might be some disturbing factors in that study as stated by the authors.

In addition, Naps et a. 2005 have stated that the *active engagement* include *responding*, *changing*, *constructing* and *presenting* –levels, so

the implications are that *passive engagement* includes only the *viewing* level, and no-viewing belongs to *no-engagement*.

At least, one interesting question remains between *active* and *passive engagement*, and where is the line between those? For example, if we provide visualization with VCR-controls, does it belong to active or passive engagement*?*

From the mixed result of existing research and unanswered questions, at least couple of changes and extensions have been proposed (Myller et al., 2009, Lauer, 2008a, Lauer, 2008b) to ET. Myller et al. (2009) have presented an extension to the ET which is called *Extended Engagement Taxonomy*, EET. The idea behind the EET is that it provides finer granularity of engagement levels to the researchers and to the visualization tool designers. They present four additional levels to the engagement taxonomy: *controlled viewing*, *providing input*, *modifying*, and *reviewing* (see Table 2).

New levels in EET	Description
Controlled-viewing	The visualization is viewed with controls.
Providing input	The learner provides input to the visualization.
Modifying	Modification of the visualization before viewing.
Reviewing	Visualizations are viewed for the purpose of suggestion and comments.

Table 2 the new levels introduced in the extended engagement taxonomy

In the *controlled viewing*, the viewing is more active than in viewing levels (active vs. passive viewing). The learner can control the execution speed of the animation or changes the views in the animation. The most common example is where the learner is able to control the visualization by VCR-controls, though backward movement is still an absent feature in many of visualization systems. The *controlled viewing* is after the level of *viewing*. In other words, the controlled viewing is the next higher level of engagement from the *viewing* level.

The *providing input* level is positioned between the *controlled viewing* and *changing*. In this level, the learner can change the input to a program or to a method during or before the execution of it.

In the *modifying* level, the learner modifies the visualization by altering the source code or input data. This level of EET is located between ET's originals levels *changing* and *constructing*.

Reviewing is the highest level of engagement after ET's *presenting* level. This level is different from presenting in that the presenter of the visualization is not especially the author of it, and the visualization is presented to audience for suggestions and feedback.

Lauer (2008a, 2008b) has also presented changes to the ET. He suggests that the *constructing* level should be divided into levels of *simulating*, *hand-constructing* and *code-based constructing* in presented order. In addition, viewing is divided into *viewing* and *controlled viewing*. More information about these extensions can be found in PhD-thesis of Myller (2009) in which Myller compares these two extensions and describe some joint work between Myller and Lauer.

Another important aspect in EET (Myller et al., 2009) is that the hypotheses of the ET are extended to collaborative learning as well. They have created a hypothesis "*increasing the level of engagement between learners and the visualization tool results in a higher positive impact of the visualization on the collaboration process*" and they have provided some partial and empirical evidence to support that hypothesis. In general, the collaboration factor is becoming more widely used method in computer science and education, and nowadays we use visualization in collaborative situations. Further reading about this collaborative aspect of learning can be found in Lehtinen et al. (1999), Hundhausen (2005), or Myller (2009).

3 Visualization tools considered in this thesis

In this section I present the visualization tools used in the articles included in this thesis. Firstly, an introduction to the TRAKLA2-tool and its previous work is presented and secondly the introduction to the ViLLE-tool is given in the same manner.

3.1 TRAKLA2

TRAKLA2 (Malmi et al., 2004) is a learning environment which is based on the visual algorithm simulation and visualization framework called *Matrix* (Korhonen et al., 2004).

In the TRAKLA2 system the learner can do visual algorithm simulation exercises related to data structures and algorithms. In the VAS-exercise the learner is supposed to simulate the execution of a given algorithm to given data by performing drag-and-drop operations inside the graphical user interface. The system is capable of providing automatic assessment and immediate feedback to the students. In addition, the input data is randomized for every instance of an exercise and the learner can practise the algorithm simulation with varying sets of data. In addition, the learner can look at the model answer with given data and compare it to his own answer. Tailored initial data prevents the use of brute force or trial-and-error techniques, and provides the learner with the possibility to make multiple re-submissions. In practise, after viewing the model answer, the learner

has to start the exercise from the beginning with new initial data to be able to submit again.

The task in the TRAKLA2-system can be for example: ”Find the 4^{th} smallest element using a minimum binary heap. Insert a given elements in a given order to the initial empty minimum binary heap by applying MIN-HEAP-INSERT-algorithm. After that, remove four times the smallest element from the heap by simulating the MIN-HEAP-DELETE algorithm.”

The exercise is performed with a graphical user interface (a java applet), and the learner changes the state of the given data structure by using VAS. For example, a binary heap can be presented in the array representation and/or in the tree-representation. The learner can use either of those representations to change the state of a data structure and the system reflects the change automatically to the other representation.

In the example exercise (see Figure 1), the learner should add the given elements (Stream of keys) to an initially empty binary heap by using Min-Heap-Insert algorithm. In the Figure 1, the learner has already inserted seven elements to the heap, and in the model answer the next value (77) is also inserted to the correct position in the heap.

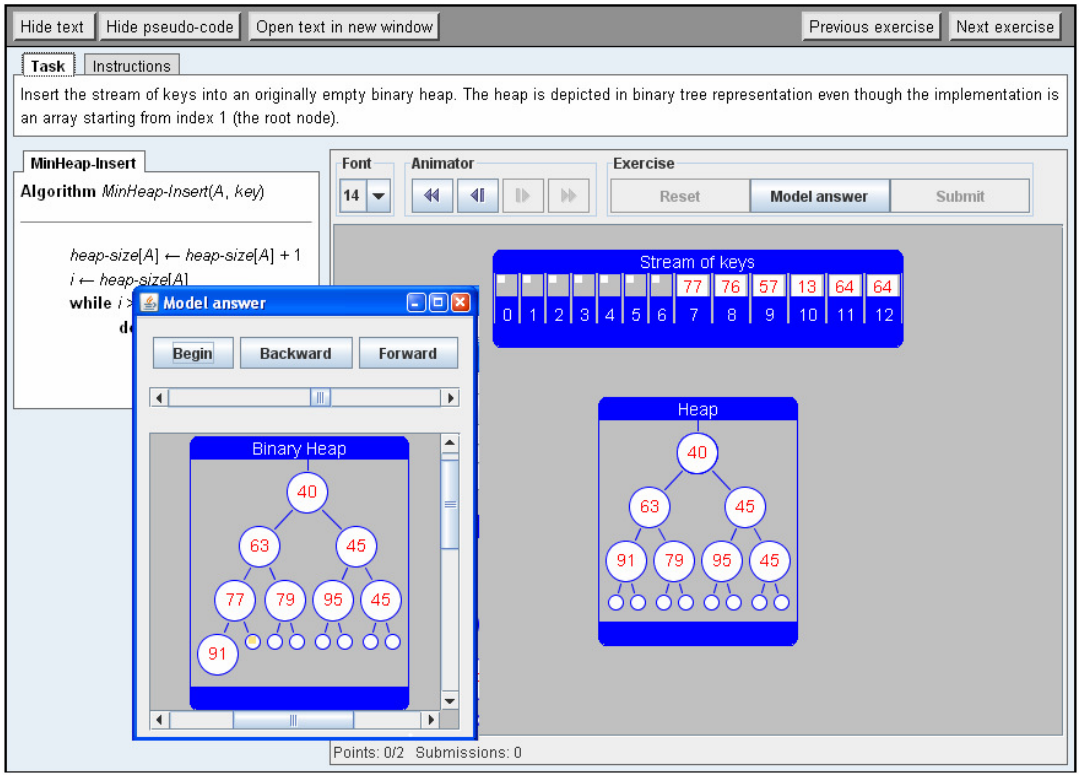


Figure 1 the user interface of the TRAKLA2 -tool

The learner's answer is recorded as a series of states which can be compared to the model solution's states. The model solution is created by the system by the actual implementation of algorithm in hand. In other words, the formal nature of algorithms makes it possible to utilize automatic assessment and immediate feedback. On the one hand, that the learner does not need to code anything. On the other hand, the learner works in the conceptual level of the algorithm and the visualization can help the learner to build the mental model of the data structure. From the ET point of view, we can use the TRAKLA2 tool in the levels of *viewing (in EET controlled-viewing)*, *changing / constructing* and *presenting*.

3.1.1 Previous work

The first intervention by Korhonen et al. (2002) study was carried out at Helsinki University of Technology (HUT) in 2001. The participants were divided randomly to three different groups

$$N_A = 372, N_B = 77, N_C = 101.$$

The study was conducted in a 12-week data structure and algorithms course (DSA). The students' performance utilizing TRAKLA learning environment, which is predecessor of TRAKLA2, was compared to other groups which used traditional classroom sessions. The main difference between TRAKLA and TRAKLA2 was that the earlier version of the system did not have a graphical user interface. The study concluded that, if the exercises are the same, there is no difference in learning between students exercising on the web (group A) or in the classroom (group B). In addition, the drop-out rates were almost equal in both of groups A and B. Meanwhile, the group C utilized more challenging exercises in the classroom exercises resulting significant increase in learning performance compared to other groups. However, the drop-out rate was also significantly higher as well.

Karavirta et al. (2005) studied the behaviour of the students using TRAKLA2 based on their submissions count and achieved points. They clustered different types of learners and concluded that there is only a small number of learners who use the resubmission-feature inefficiently like with trial-and-error tactic.

In fall 2006, Myller et al. (2007) conducted a quantitative study focusing on the ET at University of Turku (UTU). The learning

outcomes of the students were compared using the TRAKLA2-tool in different engagement levels in collaborative setting (i.e. in pairs). The treatment group utilized the tool on *changing*-level while the control group utilized the tool in EET of *controlled viewing*. There were a total of 105 participants, 52 students in the treatment group and 53 students in the control group. The setup was a typical pre-test, treatment, post-test design. The results concluded that the level of engagement had an effect on students' learning results favouring 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.

Seppälä et al. (2006) conducted a study related to the misconception of students. They analysed student's answers to a simulation exercise related to binary heap. They suggested that many students recognize the goal of the exercise, but they do not study the algorithm enough. They identified many misconceptions which can be modelled to the TRAKLA2 and identified automatically by the system to give more detailed feedback.

3.2 VILLE –visual learning tool

We have developed a program visualization tool called ViLLE at the University of Turku. Its main goal is to illustrate the dynamic behaviour of program code, the changes in the program state during the execution with various graphical and textual means. In addition, the tool is primarily designed for the teacher. The tool has built-in editors for creating and modifying syntax, examples handling and editing questions and it supports multiple programming languages.

For example, the parallel view makes it possible to compare the syntax of different programming languages side by side and emphasizes the similarity of basics of imperative programming languages. And despite of the small differences in syntax the basic functionalities of constructs are quite similar in all (imperative) programming languages. In addition, ViLLE's automatically assessed exercises can be easily integrated as a part of a programming course by using the TRAKLA server (Malmi et al. 2004) or with the export function the example set can be distributed as an independent collection on the web or in other media like usb-memory.

The ViLLE-tool has multiple exercise types. In the *tracing* exercise, the student is asked pop-up questions while the execution is in the progress. In the *programming* exercise, the student is required to code a short piece of code. It must be noted, that the tool can provide automated assessment and visualization of the code inside tool's limitations. In the *programming code sorting* exercise, the lines of a given fragment of code are randomly shuffled. The goal of this type exercise is to rearrange the given code lines into order to satisfy the requirements.

3.2.1 Teacher point of view

From the design point of view, the most important user of the ViLLE system is the teacher. The teacher is the person who decides if a student uses the system. In the academic world the most challenging thing is to convince the teacher to use a tool, the students will always follow the teacher's decision. Hence, the ViLLE-tool (see Figure 2) includes many features for the teacher including an easy to use

graphical interface to built-in editors for the programming language, examples and questions. In addition, the system includes predefined and easily customizable set of exercises with exporting and automated assessment features covering most of the topics in the typical first programming course.

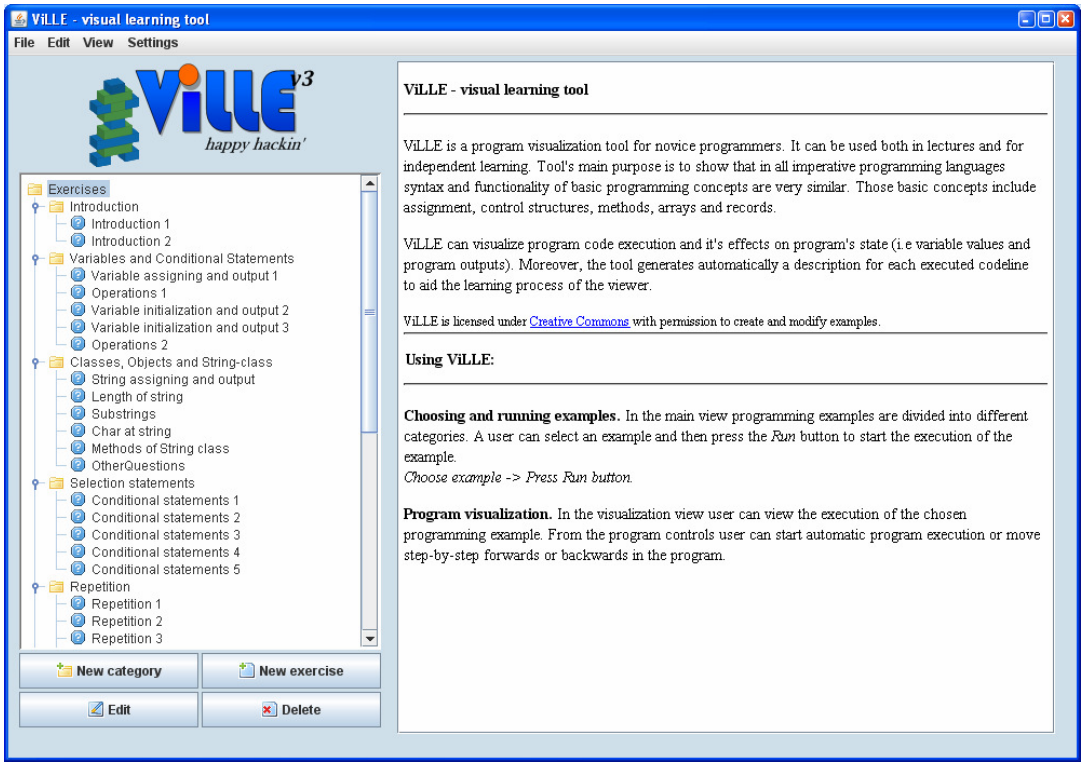


Figure 2 The main view of the ViLLE -tool

To conclude, the teacher can use his own teaching philosophy without a need to adjust it to a tool's constraints, and the tool can be integrated easily to any course without investing a great amount of time to installing, exploring, and maintaining the tool.

3.2.2 Student point of view

The following section presents ViLLE's features from the student point of view (see figure 3).

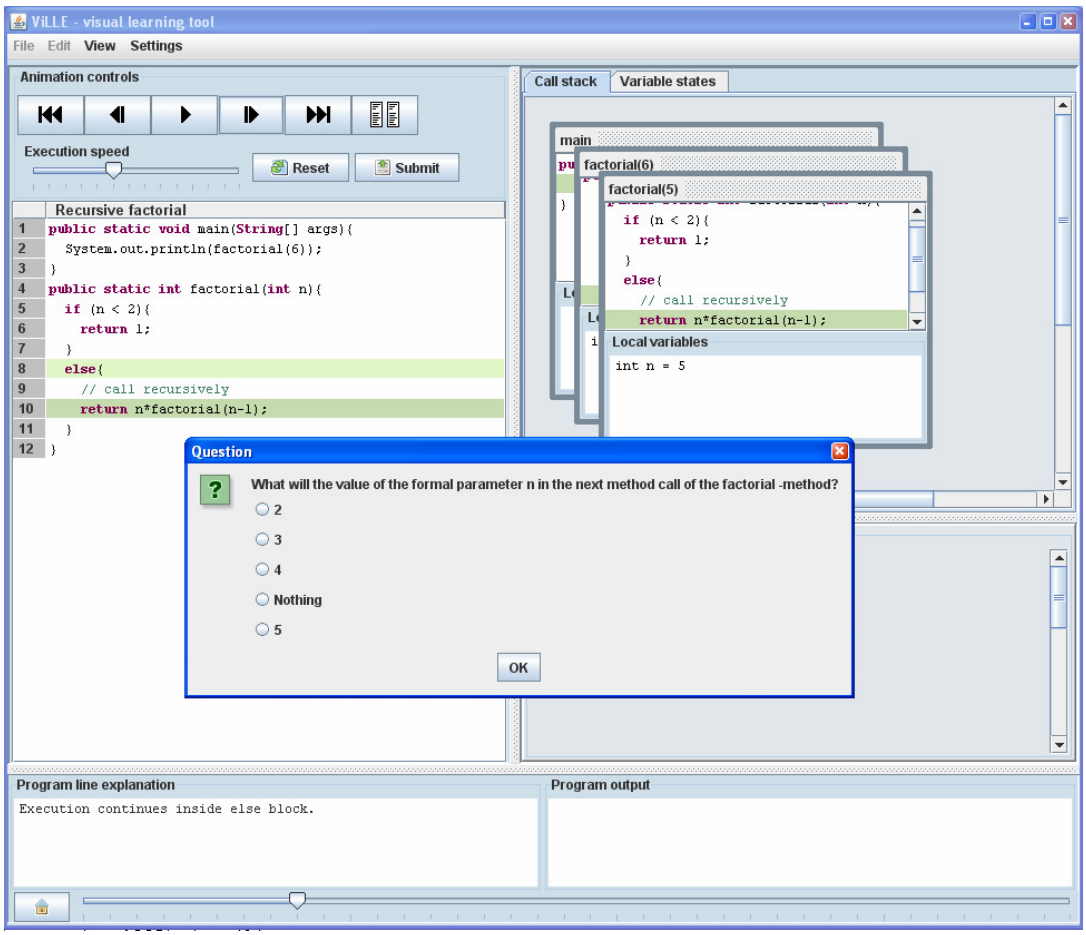


Figure 3 The visualization view of the ViLLE -tool

The features are grouped under the following topics:

Visualization of the program execution. The execution of the program code is visualized line by line. Currently and previously executed program code lines are highlighted in different colours. In addition, the variable values and changes in them, program line explanation and the output of the program are presented in their own area. The *call stack view* presents subprograms, local variables and the return values. Moreover, arrays are presented graphically in their own area called *shared memory*.

Language independency. The program code execution is visualized similarly regardless of the chosen programming language, and the language can be changed anytime during the visualization. In addition, the program code execution can be viewed in two different languages simultaneously in parallel view.

Visualization controls: Controls are flexible – the user can move one step at a time, both forwards and especially backwards in the program execution. The user can view the execution as an animation with adjustable speed. The user can use the execution slider to move directly to any state of the program execution. The execution slider located at the bottom of the visualization view also has a secondary function: the number of steps can be used to determine the complexity of the program and to compare the complexities of different algorithms.

Engagement and interaction. The system supports multiple views of engagement and interaction. The plain form in engagement is the possibility to view the execution of the program code. In addition, the students can answer presented questions while tracing the program code. They can also modify the program code and those changes can be visualized immediately. However, since the editing must be done in Java, this feature cannot naturally be utilized in all courses with other programming languages. From the ET point of view, we can use the ViLLE tool in the levels of *viewing (EET controlled viewing)*, *responding*, *changing* and *presenting*. More information about the system can be found in (Rajala et al., 2007, Kaila et al., 2008, Kaila et al., 2009).

4 Summary of publications

The first three papers (**P1**, **P2**, and **P3**) present a set of experiments in which the TRAKLA2 tool was introduced and evaluated at two different universities.

The **P1** (Laakso et al., 2004) presents the first results of the study of the introduction of TRAKLA2 system into the course of data structures and algorithms at the University of Turku in 2004. We compared students' learning results with the results of the previous instance of the same course. The students' performance statistics were clearly better than in 2003 when only pen-and-paper types of exercises were used in classroom sessions. In addition, 100 students were surveyed about their attitudes (and changes in those) towards web-based learning environments while getting involved with a wholly new system providing automatic feedback and the chance to resubmit their solutions.

Our results show that such an on-line learning environment considerably increases positive attitudes towards web-based learning, and according to students' self-evaluations, the best learning results are achieved by combining traditional teaching and web-based learning.

The **P2** (Laakso et al., 2005a) is the second publication of the study conducted in University of Turku in 2004 accompanied with results from a usability study. It presents results from three interrelated studies focusing on introducing the TRAKLA2 learning environment in a data structure and algorithms course (DSA) at two different

universities. The students used a new system capable which was capable of providing automatic assessment and immediate feedback of visual algorithm simulation exercises. Students learning performance was compared to a previous year's course in which TRAKLA2 was not used. In addition, the students' attitudes, opinions and especially changes in them were gathered with surveys. We also conducted a usability study at Åbo Akademi in which the students used a TRAKLA2-exercise and their actions were monitored in a usability lab.

The study concluded that TRAKLA2 had a positive effect on students' learning when looking the learning performance, and the tool activated students' behaviour in other areas of the DSA course. Moreover, the number of passed attendants rose from 49 to 81; TRAKLA2 was extra handy to less talented students, supporting them to get over the edge and pass the course. A large questionnaire study at UTU has shown that students' attitudes strengthened positively towards web-based exercises. According to students' self-evaluations, the best learning results are achieved by combining traditional exercises with web-based ones. The results from the usability study at Åbo Akademi showed that the TRAKLA2 system is easy to use and it takes a short time to learn to use the system.

The results from the **P1** and **P2** encouraged us to integrate the TRAKLA2 system more heavily in our DSA course at UTU. The **P3** (Laakso et al., 2005b) presents results from this intervention study which was carried out at UTU and HUT. In that study, the studies from 2001 (Korhonen et al., 2002) and 2004 (**P1**) were repeated. The

students (*N = 133 + 134*) were divided randomly into two groups in both institutions. The research setup is described in Figure 4.

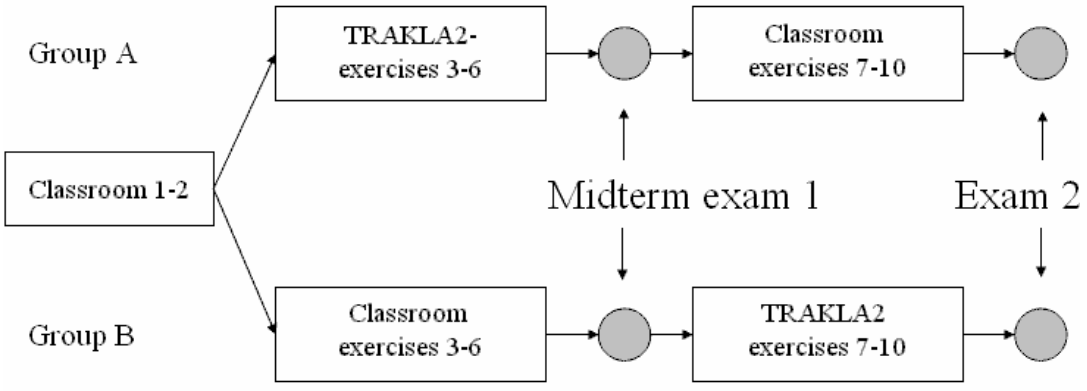


Figure 4 the setup of the study (Laakso et al., 2005b).

In this study, the group A started the solving procedure (after joint classroom exercises) with web-based exercises while the group B practised the same topics in a classroom. In the midpoint of the course, the treatment was changed: the group A continued in the classroom while the group B utilized TRAKLA2 exercises on the web. The students' performance was measured with two exams, one in the middle of the course and one in the end of the course.

The study found that the there were no statistically significant differences in learning performance if the exercises are the same. In addition, the results suggest that it is beneficial to introduce easy and human guided exercises at the very beginning of the course and that there is an emerging need for both type exercises in a DSA course. Based on the students' self assessment, the recommended way to introduce the web-based exercises in DSA is by combining these approaches. There are certain types of exercises which are suitable to be solved and automatically assessed on the web while other exercises

(i.e. proofing and analysing) are more suitable for the traditional classroom sessions. It seems that in these kinds of more challenging exercises, human guidance is needed in order for students to fully understand the exercise's multifaceted nature like to grasp underlying reasons of the complexity time of a given algorithm.

The **P4** (Laakso et al., 2009) reports outcomes from a repetition study, in which the design of the study was based on the experiment by Myller et al. (2007) with some design flaws improved. The goal of this repetition study was to find the effect of collaboration on learning performance when students are learning in different levels of engagement of EET. There were a total of 75 students in this study, and they were randomly divided into two groups. The pairs in the control group utilized TRAKLA2 exercises in the *controlled viewing* level, and the treatment group utilized the TRAKLA2 exercises in the *changing/constructing* level of EET. The study itself was a typical pre-test, treatment, post-test design. In addition, the screen activity with sound was captured for each pair.

The study concluded that both groups learned with statistically significance difference. The students gained a statistically significant improvement in performance in shared questions from the pre- to the post-test in both groups. Still there was a trend favouring the treatment group in almost all of the questions, so we wanted to find out the underlying reason(s) for it. It must be noted, that in the original setup of this study, there was no statistically significant difference in learning between the groups so we could not reject the null hypotheses. However, when the screen recordings were analysed there was a major finding, many students in the treatment group performed

only in the *controlled viewing* level (i.e. condition of the control group). In this observational study (see Figure 5), the students (Viewing T), who utilized TRAKLA2 exercises only in the *controlled viewing* level, were moved to the new control group (Viewing A). After the rearrangement of the groups, the treatment group (Changing T) outperformed the new control group (Viewing A) with statistically significant difference.

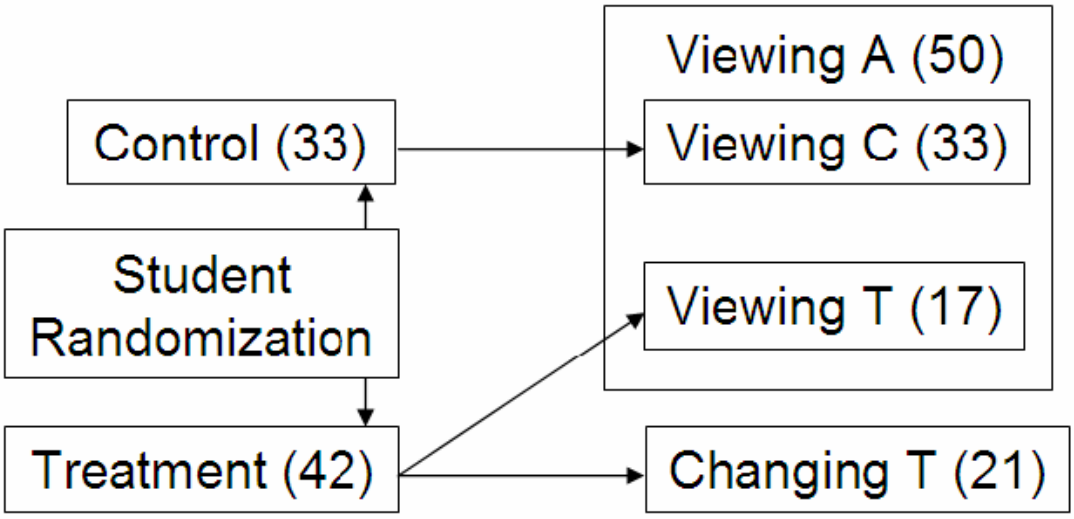


Figure 5 the setup of the study and the rearrangement (Laakso et al., 2009).

From the findings of this observational study, it can be said that the higher engagement level enhances learning in collaborative settings with visualization. This fact is supported by analysis by Myller (2009) in his PhD thesis. He has shown that by combining the results from these two consecutive experiments (Myller et al. 2007 and the **P4**) and by analysing the post-test results with the binomial test ($F(16,21,0.5) = 0.013$, two tailed, $p < 0.05$) we can actually reject the null hypothesis. To conclude, there is a difference between the students' learning performance in EET's levels *controlled viewing* and *changing* when learning is done in collaboration.

In addition, the **P4** concluded that the use of screen capturing software and voice recording should be a standard procedure in this type of research setup because then we can verify that the participants really do what we expect them to do. By doing this, we can avoid making false research conclusions and implications in our studies.

The **P5** describes (Rajala et al., 2008) and discusses the results of a study on the effectiveness of ViLLE. The research was carried out at the University of Turku, and it included students in their first programming course. Students were divided randomly into two groups, and the setup of the study was a typical pre-test, treatment, post-test design lasting two hours (see Figure 6). The control group used only traditional textual material during the session, whereas for the treatment group, the same material was extended with interactive examples using ViLLE.

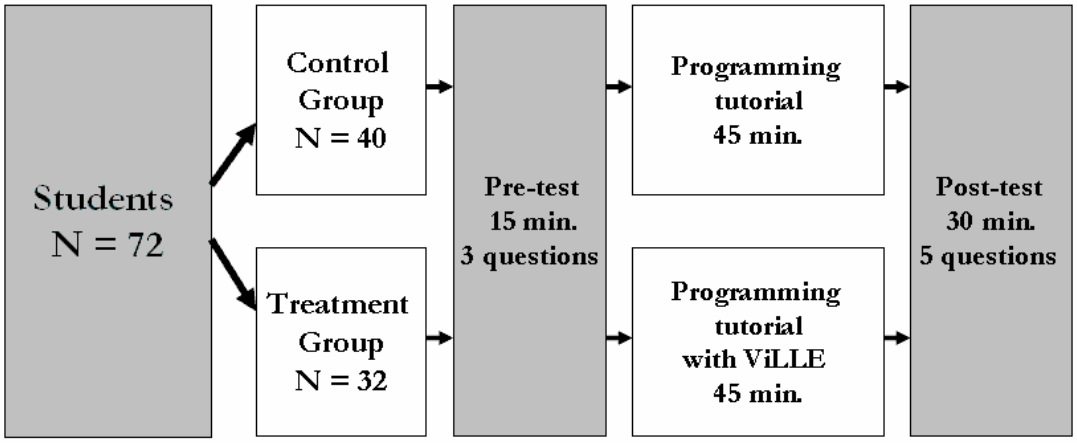


Figure 6 the setup of the study Rajala et al. (2008).

With this research setting, we formed two research questions: "Does ViLLE help students in learning to program?", and "Is there any difference in learning when previous programming experience is taken

into account?" We did not find any evidence to reject the first null hypothesis, because the differences were not statistically significant. For the latter question, we obtained evidence that ViLLE enhances the learning of students with no prior programming experience, so that the statistically significant differences between the novices and the more experienced learners disappeared as a result of a single training session. This indicates that program visualization indeed improves novice students' learning, and this phenomenon has been supported by latter research reported in Kaila et al. (2010).

In the **P6** (Laakso et al., 2008), we investigated the influence of prior experience of VILLE on the students' performance. Prior experience of the tool and especially the meaning of the graphical notation used by the tool can be also seen as a form of cognitive load of the learning environment (Lehtinen et al., 2006). There were 17 students in two sessions in the control group, and there were 7 students in one session in the treatment group. The condition was randomized between the group, and the control group used the ViLLE-tool only in the experiment while the treatment group utilized the tool in the course before the experiment. The difference between the groups was the in knowledge of the tool; its user interface, and the meaning of the graphical notation used by the tool before the study. The setup of the study was a two hour pre-test, treatment, post-test design (see Figure 7).

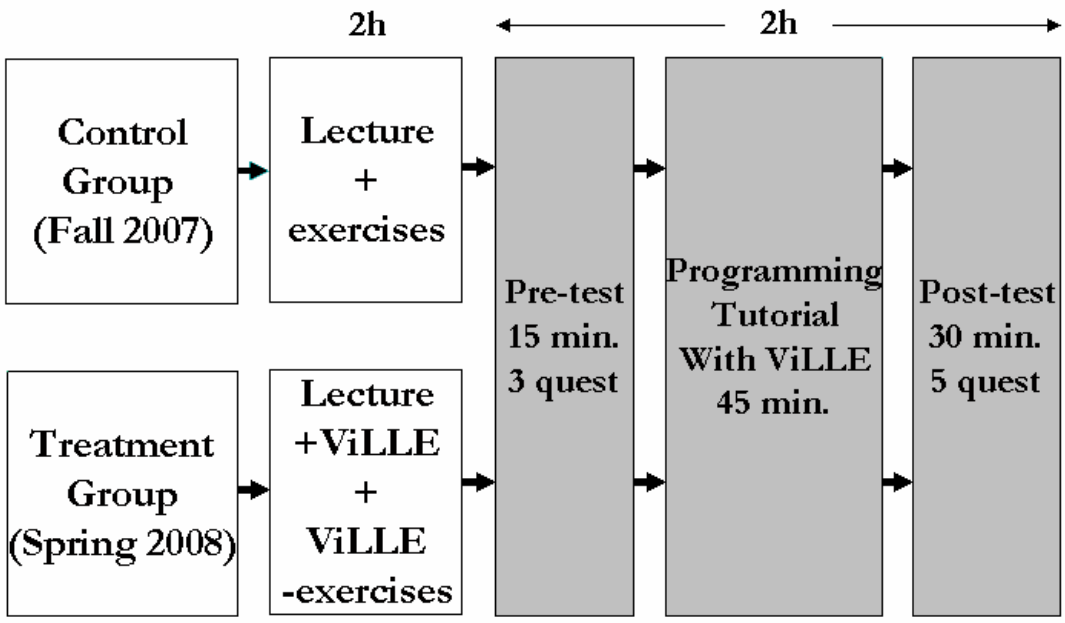


Figure 7 the research setup of Laakso et al. (2008).

The treatment group outperformed the control group with statistically significant difference in the post-test, and there was no difference between the groups in the pre-test. The reason behind this was that the students in the control group did not know how to fully interpret the graphical notation of the tool and some part of their learning effort was spent on the tool itself. On the other hand, the students in the treatment group were able to focus their learning effort solely on the topic in hand. One implication of the results is that the participants need to have a good introduction to a tool when evaluating the effectiveness. In other words, the participants should know how to handle the graphical user interface of the tool and to know how to interpret the graphical notation used by it. Moreover, many studies seemed to ignore this or decided not to report this important factor. It may be concluded that teachers should give a proper introduction to the tool to students in order to maximize the learning benefits from it.

4.1 Contributions of the Author

This thesis includes six research papers which report studies of the effectiveness of visualization-based tools on students' learning and factors which influence this. The **P1** is the first report of the study in which TRAKLA2 tool was introduced in DSA-course at University of Turku. In that paper, I was the main author of the paper and I mainly designed, carried out and analysed the data of the experiment. The TRAKLA2 course was designed mostly by me in collaboration with the course's teacher Jouni Järvinen from the existing TRAKLA2 -exercises. In addition, the maintenance of a TRAKLA2-environment for that course was done by SVG-group at Aalto University. The reporting was done in collaboration with the authors of the paper.

The **P2** is the second report of this study and is accompanied with a usability study of TRAKLA2. The usability study was designed and analyzed by Mrs Mannila (ex. Grandell) and Ms Qiu. I was the main author of the paper, and I was main responsible person for the paper, and writing was done with the help from other authors (Tapio Salakoski, Ari Korhonen, Lauri Malmi, and Linda Mannila).

The main author of the paper **P3** is me. The setup of the study was designed by me and Ari Korhonen. I mainly analysed the data, and the reporting was done in collaboration with the authors of the paper.

The **P4** is a joint effort by me, Niko Myller from University of Eastern Finland and Ari Korhonen from the Aalto University. All authors designed this study, carried out the experiment, analysed the data, and reported the results in collaboration.

We have designed and created a program visualization tool called ViLLE at the University of Turku jointly with Teemu Rajala, Erkki Kaila and Ilkka Sillanpää. The **P5** presents the effectiveness study of the ViLLE tool. The design of the study was designed and the data was analysed mainly by me and Teemu Rajala, and the study was carried out in collaboration with Teemu Rajala. The reporting was done in collaboration between all authors of this paper.

The paper **P6** was conducted at high school of Kupittaa. The study was designed and the results were analysed mainly by me. The study was carried out jointly with all authors, and the reporting was done in the same manner.

5 Results revisited

In this section, I analyse the results in this thesis through the research questions presented in Section 1.1. In general, there are two different types of research questions in this thesis. Firstly, are the web-based tools TRAKLA2 and ViLLE effective from the student's point of view? Secondly, from the teacher's point of view, what are the implications of these results for the creation and evaluation of web-based visualization tools?

5.1 Effectiveness of the tools

The effectiveness of the TRAKLA2 and ViLLE has been pointed out with quantitative and qualitative results in several papers in this thesis. The foci in the papers **P1**, **P2**, and **P3** were the introduction and use of TRAKLA2-exercises in the DSA course at UTU. In addition, the effect of the ViLLE system is reported in **P5**.

First, I will answer the research question R1.1: "What are the students' perceptions and attitudes towards web-based learning?" Based on the self-assessment study in **P1**, **P2** and **P3**, the possibility to do TRAKLA2 exercises with automatic assessment and immediate feedback was well approved and preferred by the students. However, the students' responses indicated that there is a need for classroom and web-based exercises. It can be concluded, that the simulation exercises are more suitable for the TRAKLA2 environment and more challenging exercises like proofing and analysing are more suitable for classroom sessions, where human guidance is more often needed. In addition, over 80% of students responded in the post-course survey

that TRAKLA2-exercises should be a compulsory part of the DSA course, supporting the student acceptance. Overall, the open ended questions, course evaluation and anecdotal evidence from students revealed that the quality of the DSA course got better after the introduction of TRAKLA2.

In addition, when looking at the students' learning performance in the course after the introduction of TRAKLA2, the results are encouraging (R1.2: "How the usage of web-based tools affects students' performance?"). We have studied the effects of TRAKLA2 on the overall course statistics and the drop-out rates in the **P1**, **P2** and **P3**. The learning was at least the same in the TRAKLA2 courses when comparing overall course points and grades – there were no statistically significant differences in learning either in 2004 or 2005. The positive effect was shown also in other areas of the course. The average student was doing more work in the course overall and TRAKLA2 exercises aided especially the less talented students in passing the DSA course. The number of passed students rose by 65.4 % (from 49 to 81) from 2003 to 2004. In the 2005 study, it seems that the introduction of easy and human guided exercises in the beginning of a course is an effective measure to increase engagement and motivation of students and thus help reduce the drop-out rate.

The third sub-question to the first research question was: "Are web-based tools TRAKLA2 and ViLLE effective?"

It is now quite clear that the TRAKLA2 exercises are effective from the students' point of view, and that they are approved and preferred by the students. What makes these aforementioned results even

stronger is that the number of human-guided sessions was reduced from 13 classroom sessions in 2003 to 10 classroom sessions in 2004. We went even further in 2005, when the total number of classroom sessions was reduced to 6 (the number is still the same in 2009). In other words, we were able to increase the quality of the course by reducing the resources and keeping the learning outcomes at least in the same level. At the same time, students still preferred the web-based exercises and thought that those exercises can contribute to their learning, so the learning experience was also increased.

We have created a program visualization tool called ViLLE to assist students in the early steps of learning to program. In **P5**, we evaluated the tool with a typical pre-test, treatment and post-test design. Based on the result, it can be concluded that the ViLLE-tool has proven to be effective from the students' point of view. The learning was enhanced for the novices, and we have also shown the same phenomenon in a course long experiment (see Rajala et al., 2010). Moreover, students like to do ViLLE-exercises and it is well accepted by the students (see Kaila et al., 2009).

We can conclude that both of these tools are resource efficient as well. Students prefer to do web-based exercises and they think that that those exercises contribute to their learning.

The research question RQ1.4 was "What are the roles of automatic assessment, user engagement and immediate feedback?"

The main reason behind the learning benefits from a theoretical point of view is that the TRAKLA2 and the ViLLE tools can activate and engage the student in the learning process. This is in line with the

hypothesis presented by Naps et al. (2002). The tools' exercises are done at the active levels of the engagement taxonomy (Naps et al., 2005). Especially in TRAKLA2, the higher engagement accompanied with automatic assessment, a possibility to view model answer, and immediate feedback helps students to complete the given exercises more often.

As stated in **P3**: "It seems that automatic feedback can be adequate enough to compensate its drawbacks compared with human guidance due to the fact that it is available all the time during the learning session, thus allowing the students to study at their own pace".

In addition, both of these systems are suitable for collaborative learning (**P4**, Rajala et al., 2009). This is important because collaborative learning methods are gaining more ground in computer science education. There are also other studies in which we have shown that more interactive visualizations can increase the quality of collaboration (Korhonen et al., 2009) and produce better learning results (**P4**).

So far in this section we have described the benefits of the tools for the students, but what do they offer to the teacher? The second research questions tackled teacher related issues. For the teacher, there should be concrete and direct benefits, since the teacher is the main person to decide if a student uses a tool. Based on my experience and experiments presented in this thesis, the most important aspect for a teacher is that the tool saves time and effort in designing and creating course's teaching resources. For example, seven classroom exercises were replaced with TRAKLA2 exercises at UTU. This is the situation

every single year, so we save a decent amount of resources compared to previous situation. Of course, every year we need to set up the course in TRAKLA2 and give an introduction to students about the usage of the tool but that is only a fraction of the resources that we save every year. In addition, we can require students to do a number of exercises and monitor this process really easily with the server environment.

Still, there are many courses where the benefits of TRAKLA2-exercises are not that obvious to the instructor. For example, when the course's curriculum does not have exercise types that can be replaced by TRAKLA2 exercises, the material includes different versions of a specific concept or algorithm (e.g. the equal elements are positioned into different branches in a binary tree), the fact that the introduction of the tool takes time, learning approach is different, the pseudo code of the exercises uses different notation, etc. Sadly, in these kinds of situations, the outcome is often that the instructor does not utilize a new tool or method. In other words, there are not enough immediate gains for the instructor and the benefits for the students are not enough to offset these difficulties.

The above is valid for the ViLLE-tool as well, except that we can overcome a couple of those aforementioned problems. In TRAKLA2, the creation of a new exercise is quite hard and requires practical coding, e.g. you cannot change even a letter in the pseudo code in the exercise unless you rewrite, compile and deploy it.

The design of the ViLLE-system was centralized around the teacher from day one to overcome some of these problems, and to support

multifaceted teaching approaches. The teacher can easily handle the pre-defined sets of examples, add new exercises, add or modify the existing programming languages, add questions to examples, add short coding exercises, export the collection to the web, etc. All of these features are designed to lower the barrier to utilize the tool, and to ensure that a minimum amount of effort is needed by the instructor in the creation of the content.

To conclude, TRAKLA2 and VILLE systems are well approved by students and they can enhance learning in a resource efficient way. For example, there were approximately 60,000 automatically assessed exercise submissions in the fall 2008 by these tools in several institutions in Finland.

Personally, I think that in many cases we underestimate the role of the instructor when we design and implement new tools and systems. Furthermore, the development should be done as collaboration between many institutions, since quite often we have a common and universal problem that should be tackled together.

5.2 Methodological results

5.2.1 Evaluation of a tool

There are a couple of things to remember when evaluating the effectiveness of a new tool or a new method: first, as concluded in **P6**, the participants should be properly familiarized with the tool to ensure that they actually know how to use it. In other words, a brief active or passive introduction in the context of the treatment session is not enough, as the participants need to have an adequate knowledge of the graphical notation and its interpretation. This requires “calendar” time

and hands-on time with the tool before the experiment. Moreover, as stated in **P4**, the standard procedure should include monitoring, such as screen capture and/or voice recording, in order to check that the tool is used as expected in the treatment.

By keeping the two aforementioned aspects in mind while designing empirical experiments, we can avoid drawing false conclusions from our experiments.

5.2.2 Effectiveness of web-based tutorial in a computer lab

There are three papers in this thesis that include a typical two hour pre-test - treatment - post-test design (**P4**, **P5**, and **P6**). In addition, there are three additional experiments utilizing the same research setup (Laakso et al., 2009, Myller et al., 2007, Rajala et al., 2009).

In all of these experiments, a statistically significant increase in learning occurred in all groups during the session. In total, there were over 300 students participating in six different treatments with two different visualization tools.

All these experiments were carried out in the computer lab with practically no human-guidance. The instructor of the session was informed not to assist the participants in the actual learning, but only give guidance related to the usage of the tool like assistance in the technical problems.

There was a web-based tutorial with or without interactive exercises (ViLLE or TRAKLA) included in each of the sessions. Before the learning session, participants' knowledge of the topic was measured with a pre-test. After that, instructions for the session were given and

goals were set. One of the most important factors in these instructions was that the students were informed that they could get a small fraction of their grade based on the performance in the post-test. Based on the empirical and anecdotal evidence from the instructors, these kinds of learning sessions can be said to be successful. An average student worked really hard and achieved a statistically significant increase in learning, which – though irrelevant of the treatment – can hence be enhanced with proper usage of the tools (**P4, P5** and **P6**).

I call the above learning setting "the janitor philosophy", as the instructor doesn't need to know anything about the content. The instructor's role is solely to guide the participants through the session and assist in technical problems. Of course, in "real life" learning situation, the instructor should act like as a teacher to further enhance students' learning performance. The most effective way to utilize this kind of learning method is to provide web-based learning material with interactive exercises. Moreover, the material should be utilized in collaboration mode (i.e. with students in pairs) in an active level of engagement. In addition, it has been shown that the discussion and quality of collaborative processes can be promoted by providing interactive exercise in the active ET-levels like *responding* (Rajala et al., 2009) and *changing* (Korhonen et al., 2009)

5.2.3 The usage of new tools/methods in teaching

Chapter 5.1 can be simplified from teacher's point of view into one question: "Does the tool save time or effort for the instructor in a short time interval?" Note, that there is no mention of the student in this

process. The benefits for the student are important and those generally exist only if there is something for the teacher as well.

In addition, as lecturers we need to remember the fact that we are only humans that inherit the behaviour of our ancestors (previous lecturers). The usual process for a new lecturer is that he acquires the material from the previous lecturer – for example, a national survey in Finland revealed that 86% of materials in introductory programming courses are inherited from the previous lecturer and possibly modified in some extent to suit the new lecturer's needs. (Kaila, 2008).

The reason behind this is the well known fact, that the time spend on development of course is the time away from the research. Quite often we feel that we can do those changes more easily than create a totally new approach to our teaching. Another implication is that we live in the academic world – a scientific community in which we do know how to assess publications, but we do not know how to evaluate teaching skills and we do not have a universal standard for defining a good teacher. In practice, we quite often select our lecturers based on their research productivity, while usually there is a limited correlation between the research and teaching skills. In addition, the lecturers are not necessarily interested in following the related research of new teaching methods or learning gains. The main reasons are that they don't have enough time for it, and the initial effort to introduce a new tool or technique is too big. More importantly, it does not support their academic career as their research track record does since we value publication over teaching skills in the current academic world.

In order to promote (programming) learning in general, we need to convince lecturers nowadays to start doing more research on their own teaching. In this process, the CSE-field researchers are important players, as they can promote this kind of research approach in their own institutions, since they have the practical knowledge of conducting teaching related experiments. Still, the first step to start to share knowledge of new improvements in teaching and learning is by giving periodic seminars to our colleagues in our own institutions.

We still need to remember, that one of the main tasks of the universities is to conduct research, and that the foundations of these institutions are based on scientifically proven facts. Still, we typically do not utilize scientifically proven tools or methods in our teaching, mainly because we have not established a decent culture and process of research-based improvement of teaching or international scholarship of teaching. Typical situation is that, the university sets the boundaries to a course, but the instructor can still implement a course in his own style due to very autocratic nature of the position.

To conclude, we need to create a connection between the teaching and research, and as CSE-educators we have a very important role to deliver this message of research-based improvement of learning to all our colleagues in our own institutions and all over the world.

6 Conclusions and future work

In this thesis I have studied the effects of the web-based visualization from the student and teacher point of view. There are three types of contributions in this thesis.

Firstly, we understand much more clearly what effects there are to students' perceptions and performance when we adapt new tools in out teaching. In addition, we have shown that both the presented tools, TRAKLA2 and ViLLE, are effective for students learning in a resource efficient way, helping us to cope with the problem of teaching large course with limited resources. The first set of experiments (**P1-P3**) concluded that students' learning is at least in the same level as it is without the use of TRAKLA2, and in the same time, we managed to reduce the cost of the course. Also from anecdotal evidence we can conclude that the quality of the course was better after the introduction of a new tool. In addition, we have created the ViLLE-tool that supports automatic assessment and immediate feedback, and we have shown that it enhanced students' learning (**P5** and Kaila et al., 2010) and the learning experience (Kaila et al., 2009).

To conclude, both of these tools are well accepted and approved by the students, and they also feel that the tools contribute to their learning. The most crucial feature in that phenomenon is the automated assessment combined with immediate feedback. These features help students grasp the dynamic behaviour of programs, and can more activately engage students in the learning task. As these tools have been proven to be effective, we should start using them

more in programming education. Automated assessment is one of the most important features from the teachers' point of view since it provides some time savings for the teacher, allowing him to free some resources for better use.

It must be noted that from my experience, the success of a tool depends quite often whether the tool can provide some reasonable savings for the teacher in return for their invested time. In practice, this means that the tools should support student independent learning in some form, for instance, by providing systems that are capable of automated assessment of students' exercises with immediate feedback. In addition, those tools should be suitable for collaborative learning and we should create active engagement tools to support multifaceted learning settings (**P4**).

To conclude, we should start to develop tools and further develop existing ones for the teacher, and we should create tools which support active form of engagement, automatic assessment and immediate feedback. Moreover, we need to remember the limitations of automated assessment and that there are also exercises (i.e. proofing, analysing) in which the human-guidance is still needed.

Secondly, even though we live in a scientific world and we have scientifically proven tools out there, we adopt those tools poorly in our teaching. This is due to the fact, that we do not have enough time to keep track of latest innovations in teaching and learning. Still, we are facing the same problems in programming education all over the world, and yet we do not know how to share the knowledge of the best practices and how to utilize them effectively in our teaching. As CSE

educators, we need to start to deliver the message of new innovations to our colleagues in other research fields, and to promote research-based improvement of our teaching.

Thirdly, we have also shown some methodological implications to our research practice in CSE. When we evaluate a new tool or a method, especially one accompanied with visualization, we need to give a proper introduction to the tool and to explain the meaning of the graphical notation used by tool (**P6**). The standard procedure should include capturing the screen with audio to confirm that the participants did what they were supposed to do (**P4**). By following these guidelines, we can avoid drawing false conclusion from our studies.

In future, we need to establish international joint projects to create a more supportive tool set for programming teachers. The crucial question for that environment is "How can these tools support carrying out research projects in multi-national and multi-institutional ways?"

We need to create a community through which we can share these best practices and at the same time conduct multi-national research projects more easily. In practise, this means that we need to be able to start to share our teaching resources, like exercises and exams through a "facebook for teachers" type of system. However, it is not enough for instructor to follow social life of fellow instructor(s), he must have some concrete benefits from the platform which can be provided in the form of automatically assessed exercises. The collaborative aspect of the platform should increase the quality of the courses, exams, and exercises in the platform in the long run. The instructor can look for

new teaching resources in the platform based on instructor/student ratings. This collaborative creation process of educational resources will, with enough time, generate top-of-the-line resources for various topics in computer science education. (see ViLLE – the collaborative education tool, http://ville.utu.fi)

References

Ahoniemi, T. and Lahtinen, E. (2007). Visualizations in Preparing for Programming Exercise Sessions. Electron. Notes Theor. Comput. Sci. 178, 137-144. DOI= http://dx.doi.org/10.1016/j.entcs.2007.01.043

AlgoViz (2010). Algoviz[Algorithm Visualization Portal], Visited 30 September 2010 from http://www.algoviz.org

Baddeley, A. D. (1992). Working memory. Science, 255, 556-559.

Baecker, R. M., (1981).Sorting out of sorting. Narrated videotape, 30 minutes.

Baecker, R. M., (1998). Sorting Out Sorting: A Case Study of Software Visualization for Teaching Computer Science, chapter 24, pages 369-381. The MIT Press, Cambridge, MA, 1998.

Ben-Ari, M. (2001). Program Visualization in Theory and Practice. Informatik/Informatique 2:8-11.

Ben-Bassat Levy, R., Ben-Ari, M., Uronen, P. A. (2002). The Jeliot 2000 program animation system. Computers and Education, 40(1), 15–21.

Ben-Bassat Levy, R., Ben-Ari, M., Uronen, P. A. (2002). The Jeliot 2000 program animation system. Computers and Education, 40(1), 15–21.

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

Boyle, T., Bradley, C., Chalk, P., Jones, R. and Pickard P. (2003). Using blended learning to improve student success rates in learning to program. Journal of Educational Media, special edition on Blended Learning, 28(2-3), 165-178

Bridgeman, S., Goodrich, M. T., Kobourov, S. G., and Tamassia, R. (2000). PILOT: an interactive tool for learning and grading. SIGCSE Bull. 32(1), 139-143. DOI= http://doi.acm.org/10.1145/331795.331843

Brown, M.H. (1988). Exploring Algorithms Using Balsa II. IEEE Computer, 21(5), 14-36.

Brown, M.H. (1991). Zeus: A System for Algorithm Animation and Multi-View Editing. In the Proceedings of IEEE Workshop on Visual Languages, 4-9. New York: IEEE Computer Society Press.

Brusilovsky, P., Grady, J., Spring, M., and Lee, C.-H. (2006). What should be visualized?: faculty perception of priority topics for program visualization. SIGCSE Bulletin, 38(2):44–48.

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

Carlisle, M.C., Wilson, T.A., Humphries, J.W. and Hadfield, S.M. (2005). RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving. In the Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, USA, 176-180.

Chandler, P. and Sweller, J. (1991). Cognitive load theory and the format of instruction. Cognition and Instruction, 8, 293-332.

Chinn, D., Sheard, J., Carbone, A., Laakso M.-J. (2010) Study habits of CS1 students: what do they do outside the classroom?. In Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103 (Brisbane, Australia, January 01 - 01, 2010). T. Clear and J. Hamer, Eds. Conferences in Research and Practice in Information Technology Series. Australian Computer Society, Darlinghurst, Australia, 53-62.

Crescenzi, P. and Nocentini, C. (2007). Fully integrating algorithm visualization into a cs2 course.: a two-year experience. In Proceedings of the 12th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (Dundee, Scotland, June 25 - 27, 2007). ITiCSE '07. ACM, New York, NY, 296-300. DOI= http://doi.acm.org/10.1145/1268784.1268869.

Cross, J.H., and Hendrix, D. (2007). JGRASP: An integrated Development Environment with Visualizations for Teaching Java in CS1, CS2, and Beyond. Journal of Computing Sciences in Colleges, 23 (1), 5–7.

Cross, J.H., Hendrix, D, Jain J., and Barowski L.A. (2007). Dynamic object viewers for data structures. In proceedings of the 38th SIGCSE technical symposium on Computer science education, New York, NY, USA, 4-8.

Daniels, M., Berglund, A., Pears, A., and Fincher, S. (2004). Five myths of assessment. In Proceedings of the Sixth Conference on Australasian Computing Education - Volume 30 (Dunedin, New Zealand). R. Lister and A. Young, Eds. ACM International Conference Proceeding Series, vol. 57. Australian Computer Society, Darlinghurst, Australia, 57-61.

du Boulay, D. Some difficulties of learning to program. In E. Soloway and J. Spohrer, editors, Studying the Novice Programmer, pages 283–299. Lawrence Erlbaum, 1989

Eckerdal, A., Thuné, M. and Berglund, A. (2005). What does it take to learn 'programming thinking'? Proceedings of the 2005 international workshop on Computing education research, Seattle, WA, USA, 135-142.

Evans, C. and Gibbons, N. J., (2007). The Interactivity Effect in Multimedia Learning, Computers and Education, 49(4), 1147-1160.

Evans, C. and Sabry, K. (2002). Evaluation of the interactivity of web-based learning systems: principles and process. Innovations in Education and Teaching International, 40(1), 89–99

Gestwicki, P. and Jayaraman, B. (2002). Interactive Visualization of Java Programs. IEEE Symposia on Human-Centric Computing Languages and Environments, Arlington, 226-235.

Grissom, S., McNally, M. and Naps, T. (2003). Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. In Proceedings of the ACM Symposium on Soft-ware Visualization, San Diego, California, 87–94.

Hansen, S. R., Narayanan, N. H., and Schrimpsher, D. (2000). Helping learners visualize and comprehend algorithms. Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, 2(1).

Hübscher-Younger, T. and Narayanan, N. H. (2003). Dancing hamsters and marble statues: characterizing student visualizations of algorithms. In proceedings of the 2003 ACM symposium on Software Visualization, SoftVis'03, pages 95–104, New York, NY, USA, ACM.

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 Douglas, S.A. (2002). Low fidelity algorithm visualization. Journal of Visual Languages and Computing 13(5), 449-470.

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

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. Journal of Visual Languages and Computing, 18(1), 22-47.

Hyrskykari, A. (1993). Development of Program Visualization Systems, Report, Department of Computer Science, University of Tampere. Finland. Presented at the 2nd Czech British Symposium of Visual Aspects of Man-M6achine Systems, Praha, 1-21

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, 377–381, Austin, Texas, USA.

Kaila, E., Rajala, T., Laakso M.-J., Salakoski, T. (2010). Effects of course-long use of a program visualization tool. In Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103 (Brisbane, Australia, January 01 - 01, 2010). T. Clear and J. Hamer, Eds. Conferences in Research and Practice in Information Technology Series. Australian Computer Society, Darlinghurst, Australia, 97-106.

Kaila, E. (2008). Ohjelmoinnin opetus ja opettajien suhtautuminen opetusta kehittäviin välineisiin. Ohjelmoinnin perusopetuksen verkostohanke. Retrieved 21 January 2010 from http://www.cs.hut.fi/Research/COMPSER/Verkostohanke/julkaisut.shtml

Kaila, E., Rajala, T., Laakso, M.-J. and Salakoski, T. (2009). Effects, Experiences and Feedback from Studies of a Program Visualization Tool. Informatics in Education, 8(1):17–34.

Kaila, E., Rajala, T., Laakso, M.-J. & Salakoski, T. (2008). Automatic Assessment of Program Visualization Exercises. *In Proceedings of the 8th international Conference on Computing Education Research* (Koli, Finland, November 13 - 16, 2008). Koli '08. ACM, New York, NY, 101-104. DOI= http://doi.acm.org/10.1145/1595356.1595376

Kannusmäki, O., Moreno, A., Myller, N. and Sutinen, E. (2004). What a Novice Wants: Students Using Program Visualization in Distance Programming Course. In Proceedings of the Third Program Visualization Workshop (PVW'04), Warwick, UK.

Karavirta, V. Korhonen, A., Malmi, L. and Stålnacke, K. (2004). MatrixPro - A tool for on-the-fly demonstration of data structures and algorithms. In Proceedings of the Third Program Visualization Workshop, pages 26--33, The University of Warwick, UK.

Karavirta, V., Korhonen, A., Malmi, L., (2005). Different Learners Need Different Resubmission Policies in Automatic Assessment Systems. In Proceedings of the 5th Annual Finnish / Baltic Sea Conference on Computer Science Education, pp. 95–102.

Kehoe, C., Stasko, J. and 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.

Knowlton, K. C. (1966). L6: Bell telephone laboratories low-level linked list language. 16 mm black and white sound film, 16 minutes.

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. (2003). Visual Algorithm Simulation. PhD thesis, Helsinki University of Technology. Tech Report TKO-A40/03.

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stalnacke, 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.

Korhonen, A., Laakso, M.-J., and Myller, N. (2009). How does algorithm visualization affect collaboration? Video Analysis of Engagement and Discussions. In: Joaquim Filipe and José Cordeiro eds. Proceedings of the 5th International Conference on Web Information Systems and Technologies. INSTICC — Institute for Systems and Technologies of Information, Control and Communication, WEBIST 2009, 23-26 March, Lisboa, Portugal, pp. 479–488.

Krebs, M., Lauer, T., Ottmann, T., and Trahasch, S. (2005). Student-built algorithm visualizations for assessment: flexible generation, feedback and grading. SIGCSE Bull. 37(3), 281-285. DOI= http://doi.acm.org/10.1145/1151954.1067522

Kölling, M., Quig, B., Patterson, A. and Rosenberg, J. (2003). The BlueJ system and its pedagogy. Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13(4).

Laakso, M.-J., Salakoski, T., Korhonen, A., and Malmi, L. (2004). Automatic assessment of exercises for algorithms and data structures - a case study with TRAKLA2. In Proceedings of Kolin Kolistelut/Koli Calling---Fourth Finnish/Baltic Sea Conference on Computer Science Education. Helsinki University of Technology, 28-36.

Laakso, M.-J. , Salakoski, T., Grandell, L., Qiu, X. Korhonen, A. and 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., and Korhonen, A. (2005b). The feasibility of automatic assessment and feedback. Proceedings of Cognition and Exploratory Learning in Digital Age, Lisbon: IADIS Press, 113–122.

Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Journal of Educational Technology and Society. 12(2), pp. 267–282.

Laakso, M.-J., Rajala, T., Kaila, E. and Salakoski, T. (2008). The Impact of Prior Experience in Using a Visualization Tool on Learning to Program. Proceedings of CELDA 2008, Freiburg, Germany, 129-136

Lahtinen, E., Ala-Mutka, K., and Järvinen, H. 2005. A study of the difficulties of novice programmers. In Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (Caparica, Portugal, June 27 - 29, 2005). ITiCSE '05. ACM, New York, NY, 14-18.

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.

Lawrence, A.W., Badre, A.M. and Stasko, J.T., (1994). Empirically evaluating the use of animations to teach algorithms. In: Proceedings of the 1994 IEEE Symposium on Visual Languages, pp. 48–54.

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.

Lehtinen, E. (2006). Teknologian kehitys ja oppimisen utopiat. In book Järvelä, S., Häkkinen, P. and Lehtinen, E (eds). Oppimisen teoria ja teknologian opetuskäyttö, 264-278, WSOY Oppimateriaalit Oy.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Mostrom, 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.

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.

Mayer, R. E. (2001). Multimedia learning. London: Cambridge University Press.

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, 390–397.

Mayer, R. E., Dow, G. T., and Mayer, S. (2003). Multimedia learning in an interactive self-explaining environment: What works in the design of agent-based microworlds? Journal of Educational Psychology, 95, 806–813.

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.

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.

Moore, M. G. (1989). Editorial: three types of interaction. The American Journal of Distance Education, 3, 1–6

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., Sutinen, E. and Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. Proceedings of the Working Conference on Advanced Visual Interfaces, 373-376.

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. (2009). Collaborative Software Visualization for Learning: Theory and Applications. PhD thesis, Department of Computer Science and Statistics, University of Joensuu. University of Joensuu Computer Science and Statistics Dissertations 23.

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

Myller, N., Bednarik, R., Sutinen, E., and Ben-Ari, M. (2009). Extending the engagement taxonomy: Software visualization and collaborative learning. ACM Transactions on Computing Education. 9, 1, Article 7, 27 pages.

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). JHAV´E – 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., Norton, L.L. (2000). JHAVÉ: An environment to actively engage students in web-based algorithm visualizations. In Proceedings of the SIGCSE Session, pages 109–113, Austin, Texas, March. ACM Press, New York.

Naps, T., Cooper, S., Koldehofe, B., Leska, C., Rößling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R. J., Anderson, J., Fleischer, R., Kuittinen, M. and McNally, M. (2003). Evaluating the Educational Impact of Visualization. Working group reports from ITiCSE on Innovation and Technology in Computer Science Education.ACM Press, 124–136.

Naps, T. and Grissom, S. (2002). The effective use of quicksort visualizations in the classroom. Journal of Computing Sciences in Small 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. A. (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.

Oechsle, R. and Schmitt, T. (2002). JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI). Revised Lectures on Software Visualization, International Seminar, 76-190.

Ohjelmoinnin perusopetuksen verkostohanke, loppuraportti. http://www.cs.hut.fi/Research/COMPSER/Verkostohanke/seminaari-TKK/Tulososio.pdf, 11.12.2009.

Oudshoorn, M.J., Widjaja, H. and Ellershaw, S.K. (1996). Aspects and Taxonomy of Program Visualization. In Eades, P., Zhang, K., (Eds.). Software Visualization, Series on Software Engineering and Knowledge Engineering, 7, 3-26.

Paivio, A. (1986). Mental representations. A dual coding approach. Oxford Psychology Series No. 9. New York: Oxford University Press.

Pareja-Flores, C., Urquiza-Fuentes, J., and Velázquez-Iturbide, J. Á. (2007). WinHIPE: an IDE for functional programming based on rewriting and visualization. SIGPLAN Not. 42(33), 14-23. DOI= http://doi.acm.org/10.1145/1273039.1273042

Pears, A., Seidman, S., Eney, C., Kinnunen, P., and Malmi, L. (2005). Constructing a core literature for computing education research. *SIGCSE Bull.* 37, 4 (Dec. 2005), 152-161. DOI= http://doi.acm.org/10.1145/1113847.1113893

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. (2007). A survey of literature on the teaching of introductory programming. SIGCSE Bull. 39, 4 (Dec. 2007), 204-223. DOI= http://doi.acm.org/10.1145/1345375.1345441

Petre, M. (1995). Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. Communications of the ACM, 38(6):33-44.

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.

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

Rajala, T., Laakso, M.-J., Kaila, E. and Salakoski, T. (2007). VILLE – A language-independent program visualization tool. Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, Vol. 88, Australian Computer Society. Raymond Lister and Simon, Eds.

Rajala, T., Laakso, M.-J., Kaila, E. and Salakoski, T. (2008). Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. Journal of Information Technology Education: Innovations in Practice, 7, IIP 15-32.

Rajala, T., Kaila, E., Laakso, M.-J. Salakoski, T. (2009) Effects of Collaboration in Program Visualization. Appeared in the proceedings of the Technology Enhanced Learning Conference, TELearn 2009, Taipei, Taiwan.

Rajala, T., Salakoski, T., Kaila, E., Laakso, M.-J. (2010). How does collaboration affect algorithm learning? A case study using TRAKLA2 algorithm visualization tool. In proceedings of the Education Technology and Computer (ICETC), 2nd International Conference on vol.3, pp.V3-504-V3-508, 22-24, June. doi: 10.1109/ICETC.2010.5529489

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.

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

Rößling, G. and Ackermann, T. (2007). A Framework for Generating AV Content on-the-fly. Electronic Notes in Theoretical Computer Science. 178, 23-31. DOI= http://dx.doi.org/10.1016/j.entcs.2007.01.036

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 Häussage, G. (2004). Towards tool-independent interaction support. In Proceedings of the Third Program Visualization Workshop, PVW'04, pages 110–117, The University of Warwick, UK.

Rößling, G, Naps, T., Hall, M. S., Karavirta, V., Kerren, A., Leska, C., Moreno, A., Oechsle, R., Rodger, S. H. , Urquiza-Fuentes, J. and Velazquez-Iturbide, J. A. (2006). Merging interactive visualizations with hypertextbooks and course management. SIGCSE Bulletin, 38(4):166–181.

Rößling, G. and Naps, T. L. (2002a) A testbed for pedagogical requirements in algorithm visualizations. In Proceedings of the 7th Annual SIGCSE

Conference on Innovation and Technology in Computer Science Education, ITiCSE'02, pages 96–100, Aarhus, Denmark. ACM Press, New York.

Rößling, G. and Naps, T. L. (2002b). Towards intelligent tutoring in algorithm visualization. In Second International Program Visualization Workshop, PVW'02, pages 125–130, Aarhus, Denmark. University of Aarhus, Department of Computer Science.

Schar, S. and Krueger, H. (2000). Using new learning technologies with multimedia. IEEE Multimedia, 7(3), 40–51

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.

Shaffer, C. A., Cooper, M., Edwards, S.H. (2007). Algorithm visualization: a report on the state of the field. In Proceedings of the 38th SIGCSE technical symposium on Computer Science Education, SIGCSE'07, pages 150–154, New York, NY, USA. ACM Press.

Stasko, J. (1992). Animating Algorithms with XTANGO. ACM SIGACT News, 23(2), 67-71.

Stasko, J., Badre, A. and Lewis, C. (1993). Do algorithm animations assist learning? An empirical study and analysis. Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems, ACM Press, New York, 61-66.

Stasko, J. T., Brown , M. H., Price, B. (1997). A. Software Visualization, MIT Press, Cambridge, MA.

Sweller, J., and Chandler, P. (1994). Why is some material difficult to learn? Cognition and Instruction, 12, 185-233.

Taylor, D., Lurie, A., Horstmenn, C., Johnson, M., Sharma, S., and Yin E. (2009). Predictive vs. passive animation learning tools. In Proceedings of the 40th ACM technical symposium on Computer Science Education, SIGCSE'09, pages 494–498, New York, NY, USA.

Tenenberg, J., Fincher, S., Blaha, K., Bouvier, D., Chen, T.-Y., Chinn, D., Cooper, S., Eckerdal, A., John-son, H., McCartney, R. and Monge, A. (2005). Students designing software: a multi-national, multi-institutional study. Informatics in Education, 4(1), 143-162.

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. DOI= http://dx.doi.org/10.1016/j.entcs.2007.01.038

Urquiza-Fuentes, J. and Velázquez-Iturbide, J. Á. (2009). Pedagogical Effectiveness of Engagement Levels -- A Survey of Successful Experiences. Electronic Notes in Theoretical Computer Science. 224, 169-178. DOI= http://dx.doi.org/10.1016/j.entcs.2008.12.061

Van Haaster, K. and Hagan, D. (2004). Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool. Information Science + Information Technology Education Joint Conference, 344 – 345

Wiggins, M. (1998). An overview of program visualization tools and systems. In Proceedings of the 36th Annual Southeast Regional Conference, 194–200.

Publication reprints

Paper 1

Laakso, M.-J., Salakoski, T., Korhonen, A., and Malmi, L. (2004). Automatic assessment of exercises for algorithms and data structures - a case study with TRAKLA2.

In Proceedings of Kolin Kolistelut/Koli Calling---Fourth Finnish/Baltic Sea Conference on Computer Science Education. Helsinki University of Technology, 28-36.

Automatic Assessment of Exercises for Algorithms and Data Structures – a Case Study with TRAKLA2

Mikko-Jussi Laakso and Tapio Salakoski
University of Turku
Department of Information Technology
Turku, Finland

Ari Korhonen and Lauri Malmi
Helsinki University of Technology
Department of Computer Science and Engineering
Espoo, Finland

{milaak,tapio.salakoski}@it.utu.fi,{archie,lma}@cs.hut.fi

Abstract

This paper presents the results of the case study introducing TRAKLA2 system in the course of data structures and algorithms at the University of Turku in 2004. We compared students' learning results with the results of the previous year. The numerical course results were clearly better than in 2003 when only pen-and-paper type exercises in classrooms were used. In addition, a survey was made with over 100 students on the changes in their attitudes towards web-based learning environments while getting acquainted with a wholly new system providing them automatic feedback and the option to resubmit their solutions. Our results show that such an on-line learning environment considerably increases positive attitudes towards web-based learning, and according to students' self-evaluations, the best learning results are achieved by combining traditional teaching and www-based learning.

1 Introduction

Automatic assessment (AA) tools for CS courses are being developed and gaining acceptance more and more widely at university level education. The survey of the ITiCSE working group "How shall we assess this" in 2003 indicated clearly that the experience of using AA tools correlates with a positive attitude towards applying such methods more widely, also when assessing higher order skills (Carter et al., 2003). The field where AA is most widely used is assessing programming exercises (*e.g.* Higgins et al. (2002); Luck and Joy (1999); Saikkonen et al. (2001); Vihtonen and Ageenko (2002)). Other applications include grading algorithm exercises (Bridgeman et al., 2000; Hyvönen and Malmi, 1993; Korhonen and Malmi, 2000) and analyzing object-oriented designs and flowcharts (Higgins et al., 2002).

In this paper, we report the experiences on using the TRAKLA2 system for assessing algorithm exercises in which students simulate working of algorithms on a conceptual level. TRAKLA2 by Malmi et al. (2004); Korhonen et al. (2003) is a visual algorithm simulation exercise system that has been developed at Helsinki University of Technology (HUT). Students solve the exercises using graphical manipulation of conceptual visualizations of data structures on the screen. The system provides automatic formative and summative feedback on their work, and allows for resubmitting the solutions.

TRAKLA2 exercises were used for the first time in the basic data structures and algorithms courses at HUT in spring 2003. The system was used in parallel with the old TRAKLA system so that in total 14 TRAKLA2 exercises and 24 TRAKLA exercises were used in two courses[1]. In 2004, only TRAKLA2 was used and the total number of exercises was 26. During these two years more than 1000 students used the system.

In 2004, the University of Turku (UTU) also adopted TRAKLA2 for their data structure course with over 100 students. Compared with HUT this was a major cultural change on

[1]There were two versions of the course, one for CS majors and one for students of other engineering curricula.

the course. In HUT we have used automatically assessed algorithm simulation exercise since 1991 using the older TRAKLA tool, and thus the type of exercises and the culture of using automatic assessment is well-established both for the students and teachers. In UTU, however, no such exercises have been applied, except occasionally as pen-and-paper exercises without any automatic assessment.

In all these courses, both at HUT and UTU, TRAKLA2 exercises were a compulsory part of the course, and grading points achieved from the exercises had an effect on the final grade of the courses, although in slightly different ways. In HUT, TRAKLA2 exercises have an overall effect of 30% of the final course grade, whereas at UTU the TRAKLA2 exercises increased the number of examination points. In both institutes the minimum requirement was achieving at least 50% of the maximum points of the TRAKLA2 exercises.

The structure of the paper is the following. In the next section we give an overview of the TRAKLA2 system. Section 3 presents how the system was used in UTU, and how students attitudes and opinions were surveyed. Section 4 presents the results of the survey and final conclusions are included in Section 5.

2 Overview of the TRAKLA2 system

TRAKLA2 is a system for automatically assessing *visual algorithm simulation exercises* (Korhonen et al., 2003). It is based on the Matrix algorithm visualization, animation, and simulation framework (Korhonen and Malmi, 2002). TRAKLA2 distributes individually tailored tracing exercises to students and automatically assesses answers to the exercises. In visual algorithm simulation exercises, a learner directly manipulates the visual representation of the underlying data structures to which the algorithm is applied. The learner manipulates these real data structures through GUI operations with the purpose of performing the same changes on the data structures that the real algorithm would do. The answer to an exercise is a sequence of discrete states of data structures resulting from application of the algorithm, and the task is to determine the correct operations that will cause the transitions between each two consecutive states.

Let us consider the exercise in Figure 1. The learner has started to manipulate the visual representation of the Binary Heap data structure by invoking context-sensitive *drag-and-drop operations*. In the next step, for example, he or she can drag the key `C` from a *Stream of keys* into the left subtree of `R` in the binary heap. After that, the new key is sifted up via a *swap* with its parent until the parental dominance requirement is satisfied (the key at each node is smaller than or equal to the keys of its children). The swap operation is performed by dragging and dropping a key in the heap on top of another key. In addition, the exercise applet includes a push button for activating the Delete operation. The `Delete` button is applied in the second phase of the exercise to simulate the deleteMin operation. The student selects a node to be deleted and thereafter uses swap operations to heapify the tree again.

An exercise applet is initialized with proper *randomized input data*. The binary heap exercise, for example, is initialized with 15 alphabetic keys (Stream of keys), that do not contain duplicates. This means that the exercise can be initialized in more than 10^{19} different ways. The learner can *reset the exercise* by pressing the `Reset` button at any time. As a result the exercise is reinitialized with new random keys.

After attempting to solve the exercise, the learner can *review the answer* step by step using the `Backward` and `Forward` buttons. Moreover, the learner can *ask feedback* on his or her solution by pressing the `Grade` button in which case the learner's answer is checked and immediate feedback is delivered. The feedback reports the number of correct steps out of the total number of required steps in the exercise. Finally, it is possible for the student to *submit the answer* to the course database using the `Submit` button. By default an answer to an exercise can be submitted unlimited times; however, a solution for a specific instance of exercise with certain input data can be submitted only once. In order to resubmit a solution

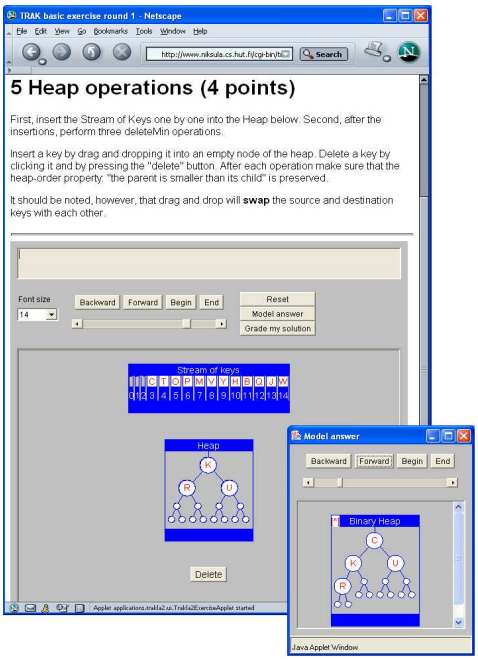


Figure 1: TRAKLA2 applet page and the model solution window.

to the exercise, the learner has to reset the exercise and start over with new randomized input data. Thus, it is not possible to grade the same solution and improve it arbitrarily before submitting it.

A learner can also *examine the model solution* of an exercise. It is represented as an algorithm animation so that the execution of the algorithm is visualized step by step. In Figure 1, the model solution window is opened in the front. The states of the model solution can be browsed using the `Backward` and `Forward` buttons. For obvious reasons, after opening the model solution for given input data, a student cannot submit a solution until the exercise has been reset and resolved with new random data.

Each TRAKLA2 exercise page (*e.g.*, Fig. 1) consists of a description of the exercise, an interactive Java applet, and links to other pages that introduce the theory and examples of the algorithm in question. The current exercise set covers almost 30 assignments on basic data structures, sorting, searching, hashing, and graph algorithms. Appendix A lists the current exercises in TRAKLA2.

3 Algorithms and data structures course at University of Turku

Algorithms and data structures (DSA-UTU) course at University of Turku included 56 lecture hours, 10 classroom exercises (each 2 hours) and 22 TRAKLA2 exercises in spring 2004. Previous courses were held with 56 lecture hours and 13 classroom exercises (2 hours each). The classroom exercises consist of five single exercises like illustrating exercises, proofing exercises, *etc.* TRAKLA2 exercises, however, are most effective to represent exercises in which the task is to illustrate how a specific algorithm works with given input values. Thus, the number of classroom exercises was cut down after TRAKLA2 was taken in use. In numbers, classroom exercises decreased from 65 to 50. Each TRAKLA2 exercise was given points from one to four. There was a possibility to get in total of 47 TRAKLA2 points in DSA-UTU course. The TRAKLA2 exercises were divided into three rounds by synchronizing the exercises to topics in hand in the DSA-UTU course.

3.1 Grading and requirements of the DSA-UTU course

There were two ways of passing the course. By taking the final examination (0-32 course points) or by taking two midterm-examinations (both 0-16 course points). In either way, student must still fulfill the minimum requirements, which are: i) students must do at least 20 of the 50 classroom exercises, ii) students must get at least 50% of the TRAKLA2 points (maximum 47 points), and iii) students must get at least 20 course points out of the total of 40 course points in share.

It was possible to get 32 course points from the examination(s) and eight course points from TRAKLA2 exercises. Conversion of TRAKLA2 points to course points was linear between the minimum requirements 50% (pass with zero course points) and 100% TRAKLA2 points (8 course points that is 20% of the maximum).

In comparison with earlier DSA-UTU courses, TRAKLA2 exercises replaced one question in the examination or a half of a question in both midterm-examinations. Traditionally one of the five questions in examination has been such an illustrative type of assignment, and this was the very question now replaced by TRAKLA2 exercises.

The final grading of the DSA-UTU course was in scale from one to three with 0.25 steps. By getting 20 course points the student will get lowest grade, which is one. In addition, by doing 60% or 80% of classroom exercises, any student can get an additional + or $\frac{1}{2}$ to his grade, respectively (still requires the student to fulfill the course minimum requirements).

3.2 The setting of the study

The attitudes of the students in UTU where studied using questionaries. Three sets of questionaries where filled by the students during the course. The first questionary at the beginning of the course, the second (Mid) at the first midterm-examination (after the first round of TRAKLA2 exercises), and the third one at the second midterm-examination (after the courses).

The first questionary was aimed to gather information about students' attitudes towards and experiences of www-based materials and tools in earlier courses. Questions also covered students' opinions about how well www-based exercises are suitable in DSA-UTU course (scale in numbers 1-5, 5 is the best). It was also asked how students prefer to do DSA-UTU courses exercises (by www-exercises, by classroom-exercises, or mixed). Students ranked different ways of doing exercises in order from one to three (one is the best, three is the worst) by their own interest. In the same way, the students also self assessed the level of their learning.

There were two main questions of yes-no type in the second questionary. The first question was about the contribution of TRAKLA2 system in the learning of course topics. The second question was about usability of TRAKLA2 and about any problems of using it. Both questions included also possibility of free text comments.

On the third questionary, the questions on the first and second questionaries were repeated. In addition, further comments and suggestions were asked for.

4 Results and discussion

As a whole, the TRAKLA2 system has worked well with surprisingly good results both at HUT and at UTU. In 2004, 30% of the students at HUT achieved the maximum number of points for the 26 exercises, and over 55% achieved 90% of the maximum. Only 15% of the students failed to get the required minimum of 50% of the points; in practice these were students who dropped the whole course early. At UTU the results were even better. The average number of points achieved was 7.34 points out of maximum 8 points.

Students' opinions of the system were determined through a web-based survey at the end of the HUT course in 2003. 364 students answered. 80% of them gave an overall grade of 4 or 5 to the system in scale 0–5, where 5 was the best grade. The system was almost unanimously

considered to aid learning and easy to use. In UTU, free feedback from the system was well in line with these observations. In addition, a different questionnaire was carried out which surveyed how students' attitudes towards on-line learning environments was changed when they had used TRAKLA2. This pointed out clearly that the attitudes became more positive.

In the following, we present a more detailed analysis of the results of the survey on the UTU students' opinions and attitudes towards www-based learning. Moreover, the learning results are presented based on students' self evaluation. After that, results derived from course statistics are presented, including the numbers of students failed/passed in total, average grades, attendances in classroom and TRAKLA2 exercises, *etc.* The data is compared with the data from DSA-UTU course in spring 2003, when the course was given by the same lecturer and the classroom exercises were very similar to those in spring 2004.

4.1 The survey results

There were 96 students answers to the first questionary ('Start'), 103 to the second ('Mid'), and 81 to the third questionary ('End'). At the Start and End the students were asked about their opinion on the suitability of www-based exercises for learning data structures and algorithms. The answering alternatives were well (5), quite well (4), neutral (3), quite bad (2), and bad (1). The Start average were quite high, 3.94, and the End average were even higher, 4.84. These results indicate that www-based exercises are very suitable for learning data structures and algorithms. Also the increase of the average during the course is large and therefore it seems that www-based exercises were well accepted and approved even by students without strong positive prejudice.

As to the qualitative analysis, also the free text comments were analyzed. There were a number of answers in which students said that it is much more elegant to do this kind of illustrative type of exercises with TRAKLA2 rather than doing the same in a piece of paper step by step. Also, it was often mentioned that TRAKLA2 exercises concretized the actions and operations of an algorithm. It was also confirmed that the immediate feedback by the TRAKLA2 system helped the students to find the point where they made a mistake and encouraged them to further deepen their understanding of the subject. This is also reflected by course statistics.

In the Mid and End questionaries, the students were asked how TRAKLA2 exercises contributed to their learning. In the Mid, the question was formulated as yes/no-type, and 94% of students answered that TRAKLA2 exercises did aid their learning process. At the End, the students were asked to describe the contribution on a scale from 1 to 5 (5 is the best). The average of the answers was 4.10, and 84% of the students selected 4 or 5, while there were only two answers below 3. This result is well in line with previous results from the study at HUT.

We also asked the students to give their preference on the three ways of doing exercises: traditional classroom exercises, web-based exercises, or mixed (see Figure 2). In the same manner, the students were asked to assess the level of their learning (Figure 3). It can be seen from the answers that the students' attitudes changed positively towards www-based exercises during the course. Students prefer the most to do exercises by combining traditional and www-based exercises even in the starting questionary, and their opinion strengthened during the course so that at the end, nearly three out of four students considered mixed exercises the best. The same happened to the students' self assessment of their learning. The mixed alternative is clearly the most suitable way to learn data structures and algorithms. Furthermore, if the students' were to choose only between traditional and web-based exercises, they would prefer traditional over www-based exercises due to their better contribution to learning. This is very interesting result suggesting that although web-based exercises complement very well traditional classroom exercises, the former can hardly replace the latter in general.

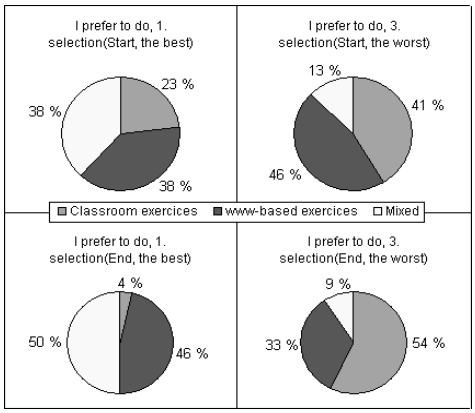


Figure 2: I prefer to do

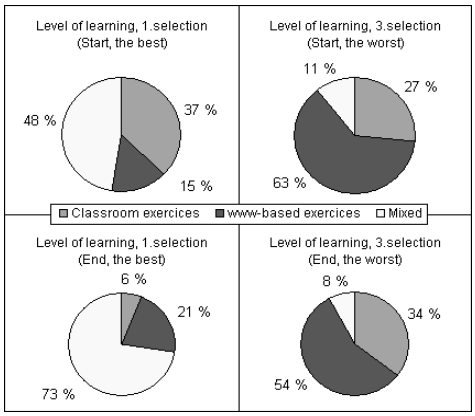


Figure 3: The level of learning

Table 1: Students' activity in classroom exercises

	Spring 2003	Spring 2004
Number of (#) attendants	186	165
Average % of classroom exercises (only who did at least 40%)	54.5	60.3
Number of (#) attendants who did 0% - 40% of classroom exercises(failed)	76	43
# attendants who did 40% - 60% of classroom exercises(no bonus)	80	79
# attendants who did 60% - 80% and received + from classroom exercises to their final grade	18	21
# attendants who did 80% - 100% and received $\frac{1}{2}$ from classroom exercises to their final grade	12	22

4.2 The course statistics

Table 1 shows statistics about students' activity in classroom exercises from DSA-UTU courses in spring 2003 and spring 2004. In addition, students got as an average of 7.34 course points from TRAKLA2 exercises, and 69,2 % of students did 100 % of TRAKLA2 exercises.

As we can see from the statistics, in spring 2004, the students were more active not only in using TRAKLA2 but also in other part of the course compared with 2003; especially, the average performance in classroom exercises raised from 54,5% to 60,3%. There is also a statistically significant difference (χ^2-test, $p < 0.01$) between the two years in the statistics in Table 1. Thus, a larger number of students received additional + / $\frac{1}{2}$ to their final grade in 2004 than in 2003. These observations confirm that the introduction of TRAKLA2 system enhanced the students' motivation and performance on the DSA-UTU course.

In Table 2, there are shown the basic statistics from DSA-UTU courses in 2003 and 2004,

Table 2: Course statistics

	Spring 2003	Spring 2004
Number of (#) attendants	186	165
Average course points	26.15	27.51
Average of the final grades	2.01	1.97
# attendants who were in second midterm-examination	58	82
# passed attendants	49	81
% of attendants who were in second midterm-examination and passed the course	84.5	98.7

which were of about the same size. There was a major increase in number of passed attendants. On the other hand, when looking at the average of course points (t-test, $p = 0.19$) and the average of final grades (χ^2-test, $p = 0.12$), there is no statistically significant difference between those two courses. Combining these two observations it can be concluded that TRAKLA2 aided many students to get over the edge and pass DSA-UTU course. Hence, it seems that TRAKLA2 is truly useful for those students who have difficulties learning data structures and algorithms by classroom exercises.

5 Conclusions

The study showed that students' attitudes strengthened positively towards www-based exercises. Moreover, the mixed alternative is far the most appropriate way to learn topics of DSA course, and it's well approved and preferred by students. Furthermore, the results suggest that www-based exercises constitute a good amendment to DSA course. However, it seems also that there exits a certain desire for more traditional exercises. Whether these students' exceptations can be fulfilled by a future version of TRAKLA2 or similar web based tools, remains an interesting challenge.

Interface of the TRAKLA2 system was easy to use, and features like possibility to get immediate feedback and the resubmit alternative aided students to complete given exercises, and by that they enhanced their learning. In addition, the study pointed out that the TRAKLA2 system affected positively on students' behaviour on other areas of DSA-UTU course, and an average student did more work for learning the course's topics. In the same time, the number of passed attendants raised from 49 to 81, thus the TRAKLA2 system aided especially less talented students to get over the edge and pass the course.

At this time, the only existing type of TRAKLA2 exercise is to illustrate how a specific algorithm works on given input. Basically, this calls for tracing the execution of the algorithm, whereas the system currently offers no support for constructive exercises, such as in which a problem is described, example input and output values are given, and the task is to construct the algorithm.

In conclusion, the TRAKLA2 system was well accepted and approved by students, and it will be used in forthcoming DSA courses also at UTU. A key task of the future is to develop novel types of TRAKLA2 exercises in collaboration between Helsinki University of Technology and University of Turku.

References

Bridgeman, S., Goodrich, M. T., Kobourov, S. G., Tamassia, R., 2000. PILOT: An interactive tool for learning and grading. In: The proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education. ACM, pp. 139–143.
URL `citeseer.nj.nec.com/bridgeman00pilot.html`

Carter, J., English, J., Ala-Mutka, K., Dick, M., Fone, W., Fuller, U., Sheard, J., 2003. ITICSE working group report: How shall we assess this? SIGCSE Bulletin 35 (4), 107–123.

Higgins, C., Symeonidis, P., Tsintsifas, A., 2002. The marking system for CourseMaster. In: Proceedings of the 7th annual conference on Innovation and technology in computer science education. ACM Press, pp. 46–50.

Hyvönen, J., Malmi, L., 1993. TRAKLA – a system for teaching algorithms using email and a graphical editor. In: Proceedings of HYPERMEDIA in Vaasa. pp. 141–147.

Korhonen, A., Malmi, L., 2000. Algorithm simulation with automatic assessment. In: Proceedings of The 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education. ACM, Helsinki, Finland, pp. 160–163.

Korhonen, A., Malmi, L., May 2002. Matrix — Concept animation and algorithm simulation system. In: Proceedings of the Working Conference on Advanced Visual Interfaces. ACM, Trento, Italy, pp. 109–114.

Korhonen, A., Malmi, L., Silvasti, P., Nikander, J., Tenhunen, P., Mård, P., Salonen, H., Karavirta, V., 2003. TRAKLA2. URL: http://www.cs.hut.fi/Research/TRAKLA2/ (27.9.2003).

Luck, M., Joy, M., 1999. A secure on-line submission system. Software - Practice and Experience 29 (8), 721–740.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppl, O., Silvasti, P., 2004. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. Informatics in Education 3 (2), 267 – 288.

Saikkonen, R., Malmi, L., Korhonen, A., 2001. Fully automatic assessment of programming exercises. In: Proceedings of The 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'01. ACM, Canterbury, United Kingdom, pp. 133–136.

Vihtonen, E., Ageenko, E., 2002. Viope-computer supported environment for learning programming languages. In: The Proceedings of Int. Symposium on Technologies of Information and Communication in Education for Engineering and Industry (TICE2002). Lyon, France, pp. 371–372.

A TRAKLA2 Exercises

Table 3: The visual algorithm simulation exercises in TRAKLA2 system. The column *name* describes the topic and the *description* characterizes the exercise. The roman numbers (i-iv) indicate the separate exercises and the number of sub-topics.

Name	Description
Insertion into (i) Binary search tree, (ii) Digital search tree, and (iii) Radix search trie	The learner is to insert random keys into an initially empty search tree by dragging and dropping them into the correct positions.
Binary search tree deletion	The learner is to remove 4 keys from a binary search tree of 15 to 20 keys. Pointer operations are simulated by directly manipulating the edges that connect the nodes of the tree in the visualization.
Faulty Binary Search Tree	The learner is to show how to bring the following binary search tree in an inconsistent state: duplicates are allowed and inserted into the left branch of an equal key, but the deletion of a non-leaf node relabels the node as its successor.
AVL tree (i) insertion, (ii) single rotation, and (iii) double rotation	The learner is to (i) insert 13 random keys into an initially empty AVL-tree. The tree (i-iii) has to be balanced by rotations. The rotation exercises (ii-iii) require pointer manipulation, while the insertion exercise (i) provides push buttons to perform the proper rotation at the selected node.
Red-black-tree insertion	The learner is to insert 10 random keys into an initially empty Red-Black-tree. The color of the nodes must be updated and the tree must be balanced by rotations.
BuildHeap algorithm	The learner is to simulate the linear time buildheap algorithm on 15 random keys.
Binary heap insertion and delete min	The learner is to a) insert 15 random keys into a binary heap and b) perform three deleteMin operations while preserving the heap order property (see Fig. 1).
Sequential search: (i) Binary search, and (ii) Interpolation search	The task is to show which indices the algorithm visits in the given array of 30 keys by indicating the corresponding keys.
Tree traversal algorithms: (i) preorder, (ii) inorder, (iii) postorder, and (iv) level order	The learner is to show which keys in a tree the algorithm visits by indicating the visited keys in the required order.
Preorder tree traversal with stack	The learner is to simulate how the stack grows and shrinks during the execution of the preorder algorithm on a given binary tree.
Fundamental Graph algorithms: (i) Depth First Search, and (ii) Breadth First Search	The learner is to visit the nodes in the given graph in DFS, and BFS order.
Minimum spanning tree algorithms: Prim's algorithm	The learner is to add the edges into the minimum spanning tree in the order that Prim's algorithm would do.
Shortest path algorithms: Dijkstra's algorithm	The learner is to add the edged to the shortest paths tree in the order that Dijkstra's algorithm would do.
Open addressing methods for hash tables: (i) linear probing, (ii) quadratic probing, and (iii) double hashing	The learner is to hash a set of keys (10-17) into the hash table of size 19.
Sorting algorithms: (i) Quicksort, and (ii) Radix Exchange sort	The learner is to sort the target array using the given algorithm.

Paper 2

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.

Informatics in Education, 2005, Vol. 4, No. 1, 49–68

Multi-Perspective Study of Novice Learners Adopting the Visual Algorithm Simulation Exercise System TRAKLA2

Mikko-Jussi LAAKSO, Tapio SALAKOSKI
Department of Information Technology, University of Turku
Lemminkäisenkatu 14 A, FIN-20520 Turku
e-mail: {milaak,tapio.salakoski}@it.utu.fi

Linda GRANDELL, Xuemei QIU
Åbo Akademi University
Department of Computer Science, Turku Centre for Computer Science
Lemminkäisenkatu 14 A, FIN-20520 Turku
e-mail: {linda.grandell, xuemei.qiu}@abo.fi

Ari KORHONEN, Lauri MALMI
Department of Computer Science and Engineering, Helsinki University of Technology
PL 5400, FIN-02015 HUT
e-mail: {archie,lma}@cs.hut.fi

Received: December 2004

Abstract. This paper presents results from three interrelated studies focusing on introducing TRAKLA2 to students taking courses on data structures and algorithms at the University of Turku and Åbo Akademi University in 2004. Using TRAKLA2 they got acquainted with a completely new system for solving exercises that provided them with automatic feedback and the possibility to resubmit their solutions. Besides comparing the students' learning results, a survey was made with 100 students on the changes in their attitudes towards web-based learning environments. In addition, a usability evaluation was conducted in a human-computer interaction laboratory.

Our results show that TRAKLA2 considerably increased the positive attitudes towards web-based learning. According to students' self-evaluations, the best learning results are achieved by combining traditional exercises with web-based ones. In addition, the numerical course statistics were clearly better than in 2003 when only pen-and-paper exercises in class were used. The results from the usability test were also very positive: no severe usability problems were revealed; in fact, the results indicate that the system is very easy to learn and user-friendly as a whole.

Key words: automatic assessment, feedback, computer science education, usability, evaluation.

1. Introduction

Automatic assessment (AA) tools for courses in computer science (CS) are being developed and gaining acceptance more and more widely in education at university level. The

survey of the ITiCSE working group 'How shall we assess this' in 2003 clearly indicated that the experience of using AA tools correlates with positive attitudes towards applying such methods more widely, also when assessing higher order skills (Carter *et al.*, 2003). The most common area in which AA is extensively used is assessing programming exercises (e.g., (Higgins *et al.*, 2002; Luck and Joy, 1999; Saikkonen *et al.*, 2001; Vihtonen and Ageenko, 2002)). Other applications include grading algorithm exercises (Bridgeman *et al.*, 2000; Hyvönen and Malmi, 1993; Korhonen and Malmi, 2000) and analyzing object-oriented designs and flowcharts (Higgins *et al.*, 2002).

TRAKLA2 by Korhonen *et al.* (2003a,b) is a visual algorithm simulation exercise system that has been developed at Helsinki University of Technology (HUT). Students simulate how algorithms work on a conceptual level solving exercises using graphical manipulation of visualizations of data structures on the screen. The system provides automatic summative feedback on their work, and allows for resubmission of the solutions.

In 2004, TRAKLA2 was introduced at two universities in Turku, and in this paper we report the experiences from three interrelated studies focusing on using TRAKLA2: the first study compares the students' learning results between using and not using TRAKLA2 in a course on data structures and algorithms (DSA). The second study consisted of a survey made with 100 students on the changes in their attitudes towards web-based learning environments. Finally, a usability evaluation was conducted in order to assess various usability aspects. This was the first time the usability of TRAKLA2 was studied by observing users interacting with the system in a human-computer interaction laboratory; the study can therefore be regarded as a pilot evaluation in this aspect. Before presenting the studies and the results in detail, we will start by giving an overview of the TRAKLA2 system. In the end of the paper, we will present the main conclusions and some suggestions for future work.

2. Overview of the TRAKLA2 System

TRAKLA2 is a system for automatically assessing *visual algorithm simulation exercises* (Korhonen *et al.*, 2003b). It is based on the Matrix algorithm visualization, animation, and simulation framework by Korhonen and Malmi (2002). TRAKLA2 distributes individually tailored tracing exercises to students and automatically assesses answers to the exercises. In visual algorithm simulation exercises, a learner directly manipulates the visual representation of the underlying data structures to which the algorithm is applied. The learner manipulates these real data structures through GUI operations with the purpose of performing the same changes on the data structures that the actual algorithm would do. The answer to an exercise is a sequence of discrete states of data structures resulting from application of the algorithm, and the task is to determine the correct operations that will cause the transitions between each two consecutive states.

Let us consider the exercise in Fig. 1. The learner has started to manipulate the visual representation of the Binary Heap data structure by invoking context-sensitive *drag-and-drop operations*. In the next step, for example, he or she can drag the key C from a *Stream*

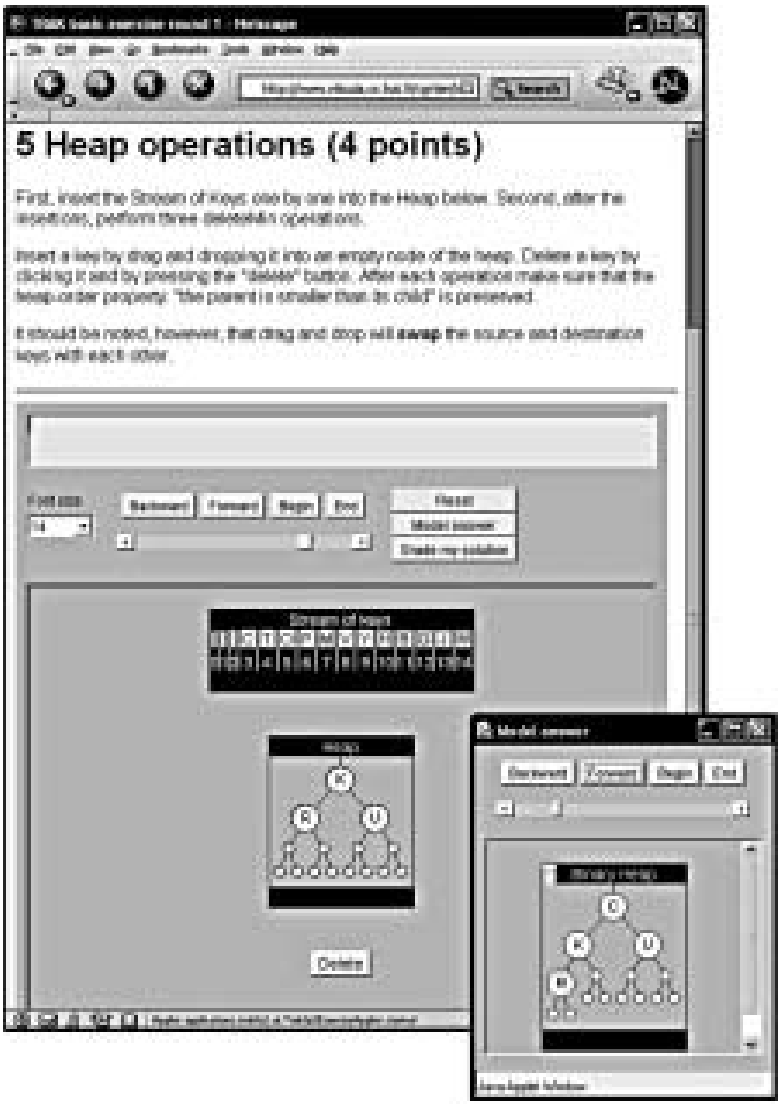


Fig. 1. TRAKLA2 applet page and the model solution window.

of keys into the left subtree of `R` in the binary heap. After that, the new key is sifted up via a *swap* with its parent until the parental dominance requirement is satisfied (the key at each node is smaller than or equal to the keys of its children). The swap operation is performed by dragging and dropping a key in the heap on top of another key. In addition, the exercise applet includes a push button for activating the Delete operation. The `Delete` button is applied in the second phase of the exercise to simulate the deleteMin operation. The student selects a node to be deleted and thereafter uses swap operations to heapify the tree again.

An exercise applet is initialized with proper *randomized input data*. The binary heap exercise, for example, is initialized with 15 alphabetic keys (Stream of keys) that do not contain duplicates. This means that the exercise can be initialized in more than 10^{19} different ways. The learner can *reset the exercise* by pressing the `Reset` button at any time. As a result the exercise is reinitialized with new random keys.

After attempting to solve the exercise, the learner can *review the answer* step by step using the `Backward` and `Forward` buttons. Moreover, the learner can *ask for feedback*

on his or her solution by pressing the `Grade` button in which case the learner's answer is checked and immediate feedback is delivered. The feedback reports the number of correct steps out of the total number of required steps in the exercise. Finally, it is possible for the student to *submit the answer* to the course database using the `Submit` button. By default an answer to an exercise can be submitted unlimited times; however, a solution for a specific instance of an exercise with certain input data can be submitted only once. In order to resubmit a solution to the exercise, the learner has to reset the exercise and start over with new randomized input data. Thus, it is not possible to grade the same solution and improve it arbitrarily before submitting it.

A learner can also *examine the model solution* of an exercise. It is represented as an algorithm animation so that the execution of the algorithm is visualized step by step. In Fig. 1, the model solution window is opened in the front. The states of the model solution can be browsed using the `Backward` and `Forward` buttons. For obvious reasons, after opening the model solution for given input data, a student cannot submit a solution until the exercise has been reset and resolved with new random data.

Each TRAKLA2 exercise page (e.g., Fig. 1) consists of a description of the exercise, an interactive Java applet, and links to other pages that introduce the theory and examples of the algorithm in question. The current exercise set covers almost 30 assignments on basic data structures, sorting, searching, hashing, and graph algorithms. Appendix 7 lists the current exercises in TRAKLA2.

3. Study 1. Effect on Learning

3.1. *Background*

TRAKLA2 exercises were used for the first time in the basic DSA courses at HUT in spring 2003. The system was used in parallel with the older TRAKLA system so that a total of 14 TRAKLA2 exercises and 24 TRAKLA exercises were used in two courses[1]. In 2004, only TRAKLA2 was used and the total number of exercises was 26. During these two years more than 1000 students used the system.

In 2004, TRAKLA2 was also adopted in the DSA course at the University of Turku (UTU); a course attended by more than 100 students. Compared to the situation at HUT this was a major cultural change: at HUT automatically assessed algorithm simulation exercises have been used since 1991 using the older TRAKLA tool, and this type of exercises and the culture of using automatic assessment are thus well-established both among students and teachers. At UTU, however, no such exercises had been applied previously, except occasionally as pen-and-paper exercises without any automatic assessment.

In all these courses, both at HUT and UTU, the TRAKLA2 exercises constituted a compulsory part of the course, and grading points achieved from the exercises had an effect on the final grade of the courses, although in slightly different ways. At HUT,

[1] There were two versions of the course, one for CS majors and one for students of other engineering curricula.

TRAKLA2 exercises had an overall effect of 30% on the final course grade, whereas at UTU the TRAKLA2 exercises increased the points in the examination. The minimum requirements were the same at both universities: the students had to achieve at least 50% of the maximum points for the TRAKLA2 exercises.

3.2. *Settings*

The Course at UTU

The DSA course at UTU (DSA-UTU) in spring 2004 included 56 lecture hours, 10 classroom sessions (each 2 hours) and 22 TRAKLA2 exercises. Previous courses were given with 56 lecture hours and 13 classroom sessions (2 hours each). The classroom exercises consist of five single exercises, such as illustrating exercises, proof exercises, *etc.* TRAKLA2 exercises, however, are most effective to represent exercises in which the task is to illustrate how a specific algorithm works with given input data. Thus, the number of classroom exercises were cut down after TRAKLA2 was taken into use. In figures, the number of classroom exercises decreased from 65 to 50. Each TRAKLA2 exercise was given points from one to four. It was possible to get a maximum of 47 TRAKLA2 points in the DSA-UTU course. The TRAKLA2 exercises were divided into three rounds by synchronizing the exercises as different topics were covered in the course.

Grading and Requirements

It was possible to get 32 course points from the examination(s) and eight course points from the TRAKLA2 exercises. The conversion of TRAKLA2 points to course points was linear between the minimum requirements 50% (pass with zero course points) and 100% TRAKLA2 points (8 course points, i.e., 20% of the maximum).

There were two ways of passing the course: taking the final examination (0–32 course points) or taking two midterm examinations (both 0–16 course points). In either case, the students still had to fulfill the minimum requirements: i) do at least 20 of the 50 classroom exercises, ii) get at least 50% of the TRAKLA2 points (maximum 47 points), and iii) get at least 20 course points out of the total of 40.

In comparison with earlier DSA-UTU courses, TRAKLA2 exercises replaced one question in the examination or half a question in each midterm examination respectively. Traditionally one of the five questions in the examination has been of illustrative type, and this was the very question now replaced by TRAKLA2 exercises.

DSA-UTU course's final grading was on a scale from one to three with 0.25 steps. By getting 20 course points the student got the lowest grade (one). In addition, by doing 60% or 80% of the classroom exercises, any student could get an additional $+$ or $\frac{1}{2}$ to his/her grade, respectively (the students were still required to meet the minimum requirements of the course).

In the following, the data are compared to the data from the DSA-UTU course in spring 2003, when the course was given by the same lecturer and the classroom exercises were very similar to those in spring 2004.

3.3. *Results and Discussion*

As a whole, the TRAKLA2 system has worked well with surprisingly good results both at HUT and at UTU. In 2004, 30% of the students at HUT achieved the maximum number of points for the 26 exercises, and over 55% achieved 90% of the maximum. Only 15% of the students failed to get the required minimum of 50% of the points; in practice these were students who dropped out of the course at an early stage. At UTU the results were even better. The average number of points achieved was 7.34 points out of a maximum 8 points.

We will now present results derived from the course statistics, including the number of students failed/passed, average grades, attendances in classroom and TRAKLA2 exercises, and so on.

Table 1 shows statistics about student activity during classroom exercises from DSA-UTU courses in spring 2003 and spring 2004. In addition, students got an average of 7.34 course points from TRAKLA2 exercises, and 69.2 % of students did 100 % of TRAKLA2 exercises.

As we can see from the statistics, the spring 2004 students were more active not only in using TRAKLA2 but also in other parts of the course: in particular, the average performance in classroom exercises increased from 54.5% to 60.3%. There is also a significant statistical difference (χ^2-test, $p = 0.01$) between the two courses when considering the number of students in each class in Table 1. Moreover, a larger number of students received an additional $+$ / $\frac{1}{2}$ to their final grade in 2004 than in 2003. These observations confirm that the introduction of TRAKLA2 enhanced the students' motivation and work input in the DSA-UTU course.

Table 2 presents the basic statistics from DSA-UTU courses in 2003 and 2004. The statistics cover only results from the second midterm examination that was the first chance to pass the course. There was a major increase in the number of passed attendants. On the other hand, when looking at the average of course points (t-test, $p = 0.19$) and the average of final grades (χ^2-test, $p = 0.12$), there is no statistically significant difference between the two courses. Combining these two observations one can conclude that

Table 1

Student activity in classroom exercises

	Spring 2003	Spring 2004
Average % of classroom exercises done (only who did at least 40%)	54.5%	60.3%
Number of (#) attendants who did 0% - 40% of classroom exercises (failed)	76 (41%)	43 (26%)
# attendants who did 40% - 60% of classroom exercises (no bonus)	80 (43%)	79 (48%)
# attendants who did 60% - 80% and received + from classroom exercises to their final grade	18 (10%)	21 (13%)
# attendants who did 80% - 100% and received $\frac{1}{2}$ from classroom exercises to their final grade	12 (6%)	22 (13%)

Table 2
Course statistics of students taking the midterm examinations

	Spring 2003	Spring 2004
Number of (#) attendants in the course	186	165
# attendants who took the second midterm examination	58 (31%)	82 (50%)
# passed attendants (in midterm examinations only)	49 (26%)	81 (49%)
# attendants who took and failed the second midterm examination	9 (15.5%)	1 (1.2%)
Average course points	26.15	27.51
Average final grades	2.01	1.97

TRAKLA2 aided many students to get over the edge and pass the DSA-UTU course. Hence, TRAKLA2 seems to be truly useful for students who have difficulties in learning DSA by completing classroom exercises. The trend is very similar in examinations later in the course (actually, the average final grades were a little bit better in 2004 compared with 2003 if we look at the final statistics including midterm examinations and all the other examinations).

4. Study 2. Attitudes and Opinions

4.1. *Background*

Students' opinions about the system were gathered through a web-based survey at the end of the HUT course in 2003. 364 students answered. 80% of them gave an overall grade of 4 or 5 to the system on the scale 0–5, where 5 was the best grade. The system was almost unanimously considered to aid learning and to be easy to use. At UTU, free feedback about the system was well in line with the observations from the HUT survey. As a continuation on this survey we decided to also study the attitudes of the UTU students using questionnaires.

4.2. *Settings*

Three sets of questionnaires were filled out by the students during the DSA-UTU course. The first questionnaire was given at the beginning of the course, the second together with the first midterm examination (after the first round of TRAKLA2 exercises), and the third one with the second midterm examination (after the end of the course).

The purpose of the first questionnaire was to gather information about students' attitudes towards and experiences of web-based materials and tools in earlier courses. The students were also asked to state how well they thought web-based exercises could suite

the DSA-UTU course (scale from 1 to 5, 5 being the best). The questionnaire also included a question on what kind of exercises the students would prefer to do in DSA-UTU courses as well as a question on how they assessed their own learning.

The second questionnaire had two main questions of yes/no type: the first question considered the contribution of TRAKLA2 to the students' learning of the course topics, and the second one covered the usability of TRAKLA2 and any potential usability problems. The students were also given the possibility to write free text comments related to these questions. In the third questionnaire, the questions from the first and second questionnaires were repeated. In addition, the students were asked for further comments and suggestions.

4.3. *Results and Discussion*

In addition, a different questionnaire was carried out which surveyed how students' attitudes towards on-line learning environments changed when they had used TRAKLA2: the results clearly indicated that the attitudes became more positive.

In the following, we present a more detailed analysis of the results of the survey on the UTU students' opinions and attitudes towards web-based learning. Moreover, the learning results based on students' self evaluation are presented.

There were 96 answers to the first questionnaire ('Start'), 103 to the second ('Mid'), and 81 to the third questionary ('End'). At the Start and End the students were asked about their opinion on the suitability of www-based exercises for learning DSA. The answer alternatives were *well* (5), *quite well* (4), *neutral* (3), *quite bad* (2), and *bad* (1). The Start average was quite high, 3.94, and the End average was even higher, 4.84. These results indicate that web-based exercises are perceived as very suitable for learning DSA. The large increase in the average during the course implies that web-based exercises were well accepted and appreciated even by students who had not demonstrated positive attitudes in advance.

The free text comments were also analyzed giving qualitative data. There was a number of answers in which students said that it is much more elegant to do this kind of illustrative exercises using TRAKLA2 instead of doing the same thing step-by-step using pen and paper. Moreover, many students mentioned that TRAKLA2 exercises concretized the actions and operations of an algorithm. It was also confirmed that the immediate feedback provided by TRAKLA2 helped the students find the point where they made a mistake, at the same time encouraging them to further deepen their understanding of the subject. This is also reflected by the course statistics.

In the Mid and End questionnaires, the students were asked how the TRAKLA2 exercises contributed to their learning. In the Mid, the question was formulated with only a yes/no-answer, whereby 94% of students answered that the TRAKLA2 exercises did aid their learning process. In the End, the students were asked to grade the contribution on a scale from 1 to 5 (5 being the best). The average of the answers was 4.10, and 84% of the students graded the contribution as 4 or 5, while there were only two answers below 3. This result corresponds well with previous results from studies at HUT.

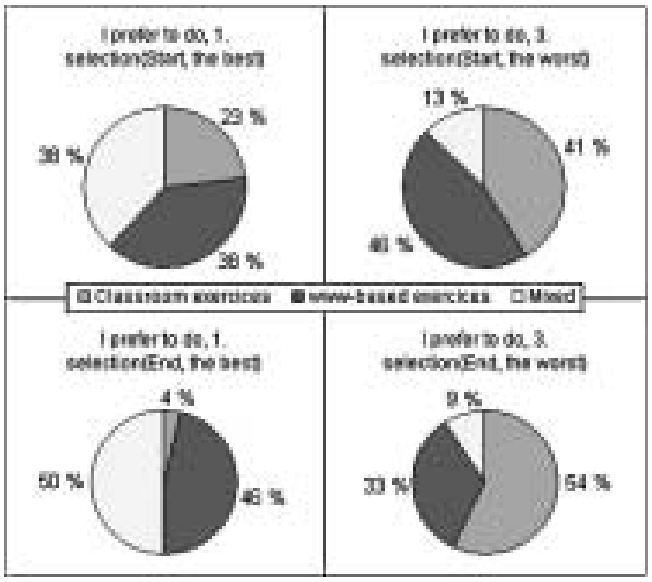


Fig. 2. I prefer to do.

We also asked the students to give their preference on three alternative ways of doing exercises: traditional classroom exercises, web-based exercises or a mix of these two approaches (see Fig. 2). In the same manner, the students were asked to assess the level of their learning (Fig. 3). The answers illustrate that the students' attitudes changed positively towards web-based exercises during the course. Even the results from the first questionnaire (Start) show that students prefer to do exercises by combining the traditional and web-based alternatives, and this opinion was strengthened during the course so that at the end, nearly three out of four students considered mixed exercises the best approach. Similar observations can be made about how the students assessed their own learning.

The mixed alternative is clearly the most appropriate approach to learn DSA. Fur-

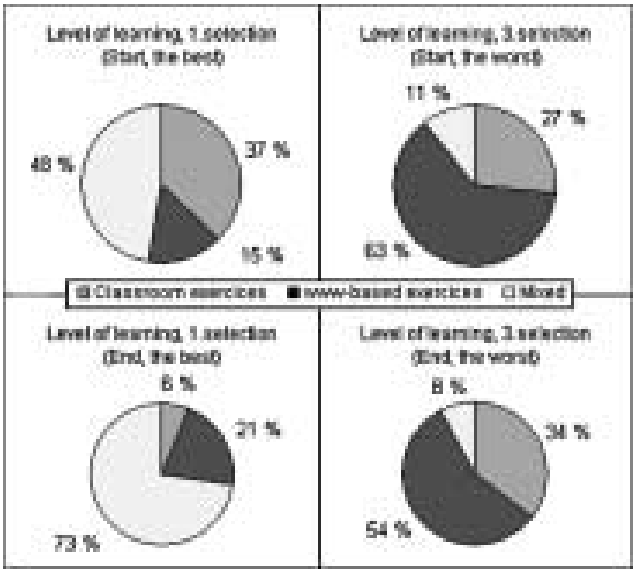


Fig. 3. The level of learning.

thermore, if the students were to choose between traditional and web-based exercises, they would prefer traditional over web-based ones due to the larger role of traditional exercises in facilitating learning. This is a very interesting result suggesting that although web-based exercises complement traditional ones done in class very well, the former can hardly replace the latter (in the current form).

5. Study 3. Usability Test

5.1. *Background*

Many studies have shown the effect of visualization software on learning outcomes in CS education (Hundhausen *et al.*, 2002; Naps *et al.*, 2003), but in order to be truly excellent, software systems should also be user-friendly. Whereas the usability of TRAKLA2 had been tested using questionnaires, this study was the first true usability evaluation including analysis of observations made in a usability laboratory.

When conducting usability tests, there are general guidelines and heuristics that can be used as a base for the evaluation. Naturally, all these general standards also apply to educational software. There are, however, specific issues that are especially important to consider when developing educational systems, such as focusing on learning efficiency, ensuring short response times, generating valuable feedback and motivating the learners. In this study we focused on the following usability aspects:

- learnability, intuitiveness and ease of use,
- usefulness and appropriateness in the curriculum,
- subjective satisfaction and motivation,
- efficiency for promoting learning.

5.2. *Settings*

The test was conducted with five students from the Department of Computer Science at Åbo Akademi University (ÅA) taking their first course on DSA, i.e., the kind of users the system has been designed for. The number of test participants might be considered low, but taking into account the limited resources available for this pilot evaluation such a small group sufficed. In addition, according to Nielsen (2000), five users are usually enough when conducting a usability test:

> As you add more and more users, you learn less and less because you will keep seeing the same things again and again. [...] After the fifth user, you are wasting your time by observing the same findings repeatedly but not learning much new.

To ensure that the students were on the same level concerning previous usage of TRAKLA2 and that their results could be considered comparable, none of the test participants had any prior experience in using the system.

Restrictions

We had many reasons for restricting the test to include only one of the exercises available in TRAKLA2: since the test participants were taking their first course on data structures, they were not yet familiar with all of the structures covered in TRAKLA2. In addition, the interface and functionality are quite similar for all exercises, wherefore testing one of these was assumed to have potential to reveal usability problems of the system as a whole. Finally, testing the entire system would be a laborious task, beyond the scope of this pilot test.

The exercise was decided upon in collaboration with the course lecturer in order to ensure that the chosen data structure and algorithm had been covered in the course up to the date of the test. The chosen exercise dealt with postorder traversal (POT) of binary trees.

Three-Part Study

The usability test consisted of three parts: observations, a pre- and post-test as well as a pre- and post-questionnaire.

The *observation sessions* took place one-by-one in a usability laboratory. Each individual session lasted approximately 30 minutes, of which the subjects spent 15 minutes interacting with the system completing the following scenarios: 1) Enroll, 2) Start the exercise on POT, 3) Use the system to solve exercises on POT and record the result from each attempt, and 4) Logout from the system.

The scenarios were put in logical order in order for the test to correspond with the authentic way of using the system. The two first scenarios can be regarded as rather trivial, giving the test participants a soft start to help them 'forget' that they were in a test situation; anxiety and nervous feelings may affect the results. The third scenario was the main task, during which we were able to observe the participants using the system. The last scenario terminated the interaction, marking the end of the test session.

All observations were recorded as video and audio material for later analysis. In order to be consistent about the information given to the participants, the same test instructions were distributed on paper to each participant at the beginning of the observation session. The subjects were encouraged to think aloud throughout the session in order to reveal the thinking process while interacting with the system.

In order to evaluate the system's effect on the students' understanding of the specific data structure, *two pen-and-paper tests* were conducted; a pre-test one week before using the system and a post-test one week after. We decided not to give the post-test immediately after using the system in order to evaluate the system's effect on the participants long-term learning.

In addition, *two questionnaires* were used: a brief pre-questionnaire was included in the pre-test in order to acquire background data about the participants. The post-questionnaire was given to the participants directly after they had finished interacting with the system and gathered information on their attitudes towards and experiences with TRAKLA2.

Table 3 illustrates how these three parts made it possible to analyze the usability aspects in focus.

Table 3
List of usability aspects in focus

Usability aspect	Observation	Pre/post-test	Post-questionnaire
Learnability & ease of use	x		x
Usefulness & appropriateness in curriculum			x
Subjective satisfaction & motivation	x		x
Effiency for promoting learning		x	

5.3. *Results and Discussion*

Observations

The data gathered from the observations were coded and analyzed using Noldus Observer software[2]. The observational material clearly indicates that the longer a subject worked with the system, the shorter the time required for solving an instance of the exercise. Whereas the mean time for solving the first exercise instance was 78.3 seconds, the corresponding time for solving the last one was 54.7 seconds. For one subject the difference between the longest and shortest duration for solving an exercise was as high as 110.0 seconds, the average difference being 60.7 seconds. The number of exercises solved during the 15 minutes was also high, the mean being 13.6 exercises.

The test participants spent most of the time during scenario three on solving exercises (80% in average), whereas 14% was spent on exploring, i.e., getting familiar with the interface, scrolling the screen and so on. The high percentage for solving actual tasks indicates that the subjects did not find it difficult to use the system in the intended way.

Unfortunately only a few of the test persons thought aloud actively, but those who did made good comments. The positive comments considered for instance the ease of dragging and dropping the keys and the possibility to change and review one's solution. There were no actual negative comments, only some recommendations. One of the subjects mentioned that the interface should be designed so that the exercise window would not require any scrollbars. This issue becomes clearer when the font size is increased. Another suggestion was to display the model answer window below the user's answer in order to facilitate comparisons and error detection. Automatically marking incorrect keys with another color was also expected to make it easier to find errors.

The analysis of the recorded material did not reveal any severe usability problems. When the users interacted with the system in an unexpected way, a 'Sorry'-dialogue was displayed on the top of the screen not giving any valuable information about what had gone wrong. The only option was to close the window, but since it was not modal some subjects did not close it; this made it pop up again every time the exercise was reset. In addition, one subject repeatedly pointed the mouse to the solution review area when attempting to reset the exercise; the majority of the subjects, however, did not have any problems with the navigation.

[2] The Observer is a system that can be used, e.g., to collect and analyse observational data. More information about the software can be found on http://www.noldus.com/site/doc200401012

Pre- and Post-tests

The pre- and post-tests both contained three assignments of corresponding nature. In addition to some POT problems, both tests included a part where the subjects were to describe how the algorithm on POT works. By this we were hoping to get information about how using the system had affected the subjects understanding of the algorithm and whether the system had helped the subjects create a mental model of how POT works (Ben-Ari, 2001), being able to explain it in own words could indicate that the subjects had reached a level of higher understanding. Unfortunately comparing the pre- and post-tests did not provide any valuable information on this point. The subjects' explanations were much shorter on the post- than on the pre-test. One can speculate that the algorithm seemed straightforward to the students after using TRAKLA2, and that they therefore did not feel any need for long explanations.

The two other assignments did, however, indicate that the participants performed better on the post-tests (see Table 4). On the first assignment the students were asked to give the order in which the keys in two binary trees would be visited using POT. On the pre-test the majority of the subjects got no points on this assignment, whereas almost everybody received a full score on the post-test. In the second assignment the subjects were asked to draw the binary tree corresponding to a given POT of keys. The difference between the post- and pre-test on this assignment was not as remarkable, but whereas three of the subjects did not receive any points at all on this assignment on the pre-test, everybody got at least two points on the post-test.

The score for each subject on the first two assignments on the pre- and post-test are given in Fig. 4. It is interesting to note that four out of five participants follow the same pattern, whereas one demonstrates an opposite behavior. Naturally, one cannot give any clear explanations for this phenomenon, but the common trend is still evident: for most participants the scores improved on the later test.

Post-questionnaire

In the post-questionnaire, the subjects were asked to grade TRAKLA2 on a five point Likert-scale (from 1 (disagree) to 5 (agree)) based on how they experienced the system. In Table 5 we give the median for the participants answers to questions that we found to be of importance in order to evaluate the first three usability aspects listed in Table 3.

As Table 5 shows, the system was regarded enjoyable and motivating. The data also imply that the participants were positive to using the system and most of them would

Table 4

Comparisons of the participants' performance on the pre- and post-tests

		Average	Std.dev
Assignment 1:	Pre-test	7	9.75
Give POT (max 20 p)	Post-test	18	4.47
Assignment 2:	Pre-test	4.2	5.76
Draw tree (max 11 p)	Post-test	4.4	3.71

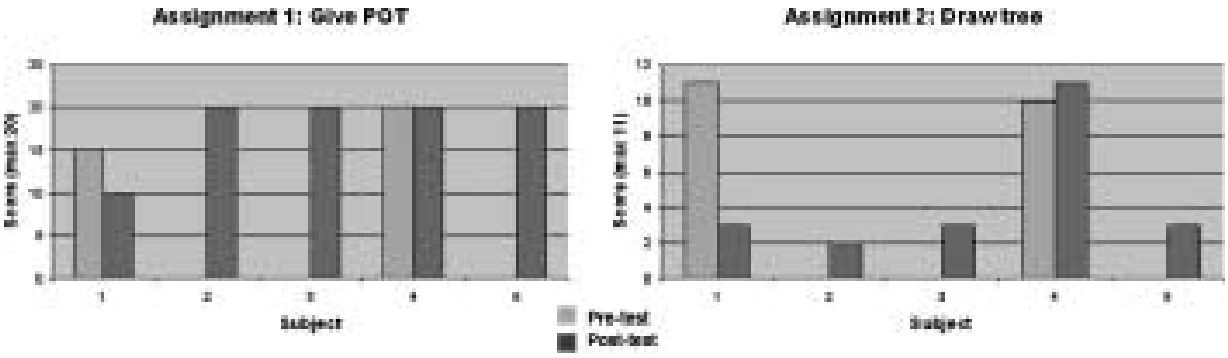


Fig. 4. Comparison of each subjects score on the two first assignments in the pre- and post-tests.

like to use it more extensively – in fact, we were asked to give them more information about the online version of the system. All except one of the participants reported that the system had helped them understand the algorithm on POT. The data indicates that the students managed to start using the system in a short time; learning how to use it was not considered difficult.

As a whole the usability evaluation has indicated that TRAKLA2 is easy to learn and use, thus fulfilling the first usability aspect: learnability and ease of use. The data from the post-questionnaire pointed out that the subjects would like to use TRAKLA2 in their studies and that they thought it would facilitate their learning (aspects 2 and 4: "Usefulness and appropriateness in the curriculum" and "Efficiency for promoting learning"). In addition, the results between the students varied more on the pre-test than on the post-test. This indicates that the system promoted the subjects' learning (aspect 4). Finally, the results from the post-questionnaire also imply that the subjects enjoyed working with the system and also wanted to have further usage. Such attitudes are in general very positive (aspect 3: "Subjective satisfaction and motivation").

Table 5

The median of the test participants' opinions after using the system (1 = disagree, 5 = agree)

	Median
Using the system was	
– frustrating	1
– enjoyable	4
– boring	2
– motivating	4
The system helped me understand the algorithm on POT	5
The system uses terms understandable and familiar to me	5
It took me a long time to start using the system	2
I would like to use the system in my studies	4
It would be easier to learn the topics in the course on data structures if I could use this system	4
Using the system was difficult	1
I solved the tasks quickly compared to the traditional pen and paper way	5

6. Conclusions

In this paper we have reported three studies, all focusing on the same system, TRAKLA2. It is interesting to see that the results from the studies support each other very well. In our opinion, testing a system in this way, from various perspectives, renders it possible to assess its true value, bringing out positive and negative aspects from different points of view.

Comparing the learning results from courses at UTU has shown that the introduction of TRAKLA2 has had a positive effect on students' learning. In addition, the first study pointed out that the TRAKLA2 system had a positive effect on students' behavior in other areas of the DSA-UTU course, for instance, the average student worked harder in order to learn the course topics. The possibility to get immediate feedback and to resubmit answers helped students to complete given exercises, thereby enhancing their learning. In addition to the positive trend in learning results, the number of passed attendants rose from 49 to 81; TRAKLA2 was extra helpful to less talented students in particular, supporting them to get over the edge and pass the course.

The vast questionnaire study at UTU has shown that students' attitudes strengthened positively towards web-based exercises. Moreover, the mixed alternative is far the most appropriate way to learn topics of courses on DSA, and it's well approved and preferred by students. Furthermore, the results suggest that web-based exercises constitute a good amendment to courses on DSA. However, there also appears to be a certain desire for more traditional exercises. Whether these students' expectations can be fulfilled by a future version of TRAKLA2 or similar web based tools, remains an interesting challenge. The results from the usability study indicate that it takes novices a very short time to learn to use TRAKLA2, and that even persons with no previous experience using the system could start working actively with it in only a few minutes.

7. Future Work

In our opinion, the studies have shown that TRAKLA2 is a usable system for enhancing learning of DSA. TRAKLA2 was well accepted and approved by students, and it will be used in forthcoming DSA courses in Turku.

At this time, the only type of exercises available in TRAKLA2 focuses on illustrating how a specific algorithm works on given input. Basically, this calls for tracing the execution of the algorithm. The system does not offer any support for constructive exercises, such as exercises in which a problem is described, example input and output values are given, and the task is to construct the algorithm. A key task of the future is to develop novel types of TRAKLA2 exercises as a collaboration between HUT, UTU and ÅA. In addition, now that a pilot evaluation in a usability laboratory has been conducted, TRAKLA2 could benefit from a larger-scale usability test, covering more exercises and letting the test persons interact more freely with the system.

A TRAKLA2 Exercises

Table 6

The visual algorithm simulation exercises in TRAKLA2 system. The column *name* describes the topic and the *description* characterizes the exercise. The roman numbers (i–iv) indicate the separate exercises and the number of sub-topics

Name	Description
Insertion into (i) Binary search tree, (ii) Digital search tree, and (iii) Radix search trie	The learner is to insert random keys into an initially empty search tree by dragging and dropping them into the correct positions.
Binary search tree deletion	The learner is to remove 4 keys from a binary search tree of 15 to 20 keys. Pointer operations are simulated by directly manipulating the edges that connect the nodes of the tree in the visualization.
Faulty Binary Search Tree	The learner is to show how to bring the following binary search tree in an inconsistent state: duplicates are allowed and inserted into the left branch of an equal key, but the deletion of a non-leaf node relabels the node as its successor.
AVL tree (i) insertion, (ii) single rotation, and (iii) double rotation	The learner is to (i) insert 13 random keys into an initially empty AVL-tree. The tree (i–iii) has to be balanced by rotations. The rotation exercises (ii–iii) require pointer manipulation, while the insertion exercise (i) provides push buttons to perform the proper rotation at the selected node.
Red-black-tree insertion	The learner is to insert 10 random keys into an initially empty Red-Black-tree. The color of the nodes must be updated and the tree must be balanced by rotations.
BuildHeap algorithm	The learner is to simulate the linear time buildheap algorithm on 15 random keys.
Binary heap insertion and delete min	The learner is to a) insert 15 random keys into a binary heap and b) perform three deleteMin operations while preserving the heap order property (see Fig. 1).
Sequential search: (i) Binary search, and (ii) Interpolation search	The task is to show which indices the algorithm visits in the given array of 30 keys by indicating the corresponding keys.
Tree traversal algorithms: (i) preorder, (ii) inorder, (iii) postorder, and (iv) level order	The learner is to show which keys in a tree the algorithm visits by indicating the visited keys in the required order.
Preorder tree traversal with stack	The learner is to simulate how the stack grows and shrinks during the execution of the preorder algorithm on a given binary tree.

To be continued

Name	Description
Fundamental Graph algorithms: (i) Depth First Search, and (ii) Breadth First Search	The learner is to visit the nodes in the given graph in DFS, and BFS order.
Minimum spanning tree algorithms: Prim's algorithm	The learner is to add the edges into the minimum spanning tree in the order that Prim's algorithm would do.
Shortest path algorithms: Dijkstra's algorithm	The learner is to add the edged to the shortest paths tree in the order that Dijkstra's algorithm would do.
Open addressing methods for hash tables: (i) linear probing, (ii) quadratic probing, and (iii) double hashing	The learner is to hash a set of keys (10–17) into the hash table of size 19.
Sorting algorithms: (i) Quicksort, and (ii) Radix Exchange sort	The learner is to sort the target array using the given algorithm.

References

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, **20**(1), 45–73.

Bridgeman, S., M.T. Goodrich, S.G. Kobourov and R. Tamassia (2000). PILOT: An interactive tool for learning and grading. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*. ACM Press, New York, pp. 139–143. `citeseer.nj.nec.com/bridgeman00pilot.html`

Carter, J., J. English, K. Ala-Mutka, M. Dick, W. Fone, U. Fuller and J. Sheard (2003). ITICSE working group report: How shall we assess this? *SIGCSE Bulletin*, **35**(4), 107–123.

Higgins, C., P. Symeonidis and A. Tsintsifas (2002). The marking system for CourseMaster. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*. ACM Press, pp. 46–50.

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

Hyvönen, J., and L. Malmi (1993). TRAKLA – a system for teaching algorithms using email and a graphical editor. In *Proceedings of HYPERMEDIA in Vaasa*, pp. 141–147.

Korhonen, A., and L. Malmi (2000). Algorithm simulation with automatic assessment. In *Proceedings of The 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education*. ACM Press, New York, Helsinki, Finland, pp. 160–163.

Korhonen, A., and L. Malmi (2002). Matrix – Concept animation and algorithm simulation system. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM Press, New York, Trento, Italy, pp. 109–114.

Korhonen, A., L. Malmi and P. Silvasti (2003a). 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*. Joensuu, Finland, pp. 48–56.

Korhonen, A., L. Malmi, P. Silvasti, J. Nikander, P. Tenhunen, P. Mård, H. Salonen and V. Karavirta (2003b). *TRAKLA2*. `http://www.cs.hut.fi/Research/TRAKLA2/` (27.9.2003).

Luck, M., and M. Joy (1999). A secure on-line submission system. *Software – Practice and Experience*, **29**(8), 721–740.

Naps, T.L., G. Rößling, J. Anderson, S. Cooper, W. Dann, R. Fleischer, B. Koldehofe, A. Korhonen, M. Kuittinen, C. Leska, L. Malmi, M. McNally, J. Rantakokko and R.J. Ross (2003). Evaluating the educational

impact of visualization. *SIGCSE Bulletin*, **35**(4), 124–136.

Nielsen, J. (2000). *Why you only need to test with 5 users*. `http://www.useit.com/alertbox/20000319.html` (March 19, 2000)

Saikkonen, R., L. Malmi and A. Korhonen (2001). Fully automatic assessment of programming exercises. In *Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education*. ACM Press, New York, Canterbury, UK, pp. 133–136.

Vihtonen, E., and E. Ageenko (2002). Viope-computer supported environment for learning programming languages. In *The Proceedings of Int. Symposium on Technologies of Information and Communication in Education for Engineering and Industry* (*TICE2002*). Lyon, France, pp. 371–372.

M.-J. Laakso is a lecturer and a PhD student at Department of Information Technology, University of Turku. He received his MSc in computer science from University of Turku in 2003. His main interests are in the field of CS education research focusing on teaching programming to novices and visualization of algorithms.

T. Salakoski received his PhD degree in 1997 at University of Turku, Finland, and currently works as a professor of computer sience at Department of Information Technology, University of Turku. In addition to computer science education research, his scientific interests include developing intelligent data-analysis methods, resources, and tools for bioinformatics and linguistics.

L. Grandell is a PhD student at Turku Centre for Computer Science and the Department of CS at Åbo Akademi University. She received her MSc in computer science from Åbo Akademi University in 2003. Her main interests are in the field of CS education research focusing on teaching programming to novices both at secondary and university level.

X. Qiu is a researcher at the Turku Center for Computer Science, Institute for Advanced Management Systems Research, Åbo Akademi University. She received her MSc (econ) in information systems in 2003 from Åbo Akademi University. Her main research interests are interactive multi-agent systems in semantic Web environment and agent-enabled knowledge mobilization.

A. Korhonen is a researcher in the Helsinki University of Technology (HUT). He received his MSc (computer science) in 1997 and DSc (computer science) in 2003 in the HUT. He is acting as the scientific leader of the Software Visualization Group in the laboratory of Computer Science and Engineering. His research includes data structures and algorithms in software visualization, automatic assessment in computer science education, and various applications of computer aided learning environments.

L. Malmi is a professor of computer science in Helsinki University of Technology (HUT). He received his doctor of technology diploma in HUT in 1997. His main research area is computer science education including software visualization, automatic assessment, new educational methods, and evaluating how they improve learning.

Vizualios algoritmavimo mokymo simuliacinės sistemos TRAKLA2 multiperspektyvus tyrimas

Mikko-Jussi LAAKSO, Tapio SALAKOSKI, Linda GRANDELL, Xuemei QIU, Ari KORHONEN, Lauri MALMI

Straipsnyje nagrinėjami rezultatai trijų tarpusavyje susijusių tyrimų, skirtų studentų supažindinimo su sistemos TRAKLA2 galimybėmis analizuoti. Atliekant tyrimus remtasi patirtimi, įgyta 2004 metais Turku ir Åbo Akademijos universitetuose dėstant duomenų struktūrų ir algoritmavimo kursus. Naudojant TRAKLA2 studentai buvo susipažindinami su visiškai nauja uždavinių sprendimo sistema, turinčia automatinį grįžtamąjį ryšį bei sudarančia galimybę peržiūrėti savo sprendimus. Norint palyginti besimokiusiųjų pasiektus rezultatus, buvo atlikta 100 studentų apklausa. Ja buvo siekiama išsiaiškinti, kaip keitėsi studentų požiūris į žiniatinkliu pagrįstą mokymosi aplinką. Tiriant kompiuterio ir žmogaus dialogą, sąsają, buvo atliktas ir sistemos praktiškumo įvertinimas.

Gauti rezultatai rodo, jog TRAKLA2 sistema padarė nemažą įtaką, kad atsirastų ir sustiprėtų teigiamas požiūris į žiniatinkliu pagrįstą mokymąsi. Atsižvelgiant į studentų atsakymus, galima teigti, kad geriausi rezultatai pasiekiami derinant tradicines ir žiniatinkliu pagrįstas užduotis. Be to, studentų pasiektų rezultatų statistika buvo kur kas geresnė nei 2003 metais, kai dėstant atitinkamus kursus buvo naudota vien „tušinuko ir popieriaus“ mokymo metodika. Sistemos praktiškumo testo rezultatai buvo taip pat teigiami: neaptikta jokių žymesnių panaudojamumo nesklandumų. Gauti rezultatai iš esmės rodo, jog sistema vartotojui nėra sudėtinga, ją nesunku suprasti, išmokti naudotis ja paprasta ir patogu.

Paper 3

Laakso, M.-J., Salakoski, T., and Korhonen, A. (2005). The feasibility of automatic assessment and feedback.

Proceedings of Cognition and Exploratory Learning in Digital Age (CELDA), Lisbon: IADIS Press, 113–122.

THE FEASIBILITY OF AUTOMATIC ASSESSMENT AND FEEDBACK

Mikko-Jussi Laakso and Tapio Salakoski
University of Turku
Department of Information Technology
Lemminkäisenkatu 14 A, FIN-20520 TURKU, Finland

Ari Korhonen
Helsinki University of Technology
Department of Computer Science and Engineering
Teknillinen Korkeakoulu, Tietojenkäsittelyopin laboratorio
PL 5400, 02015 TKK

ABSTRACT

In this study, we report on the results of studies in which two randomly selected groups of students were monitored while they solved exercises in a Data Structures and Algorithms (DSA) course. The first group did the exercises by using web-based system and the second one in the classroom sessions and the roles of the groups were changed in the middle of the course. A web-based system capable of automatically assessing exercises and giving feedback was employed. The research question was to find out how we should introduce the self study material and automatically assessed exercises to the students in order to maximize their learning experience and to avoid drop outs. In addition, we surveyed the students' attitude towards web-based exercises by using questionnaires. The students were asked what kind of exercises they would prefer to do in DSA courses as well as how they would assess their own learning experience in the three different setups (human guided, web-based or combination of these two). All these studies were carried out simultaneously in two different universities.

The results suggest introducing easy and human guided exercises at the very beginning of the course. However, we conclude that currently there is an emerging need for both web-based and classroom exercises. We claim that 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 the most suitable to be solved and automatically assessed on the web while the rest of the exercises are best suitable for traditional classroom sessions. We believe that the results of this study can be generalized to cover also other similar learning environments than that used in this research to give automated feedback for the students, and thus improve the learning experience.

KEYWORDS

automatic assessment, feedback, computer science education, evaluation.

1. INTRODUCTION

Computers are now being used to ease the learning process in a variety of disciplines, including computer science and engineering. There are several methods to apply. One such method is animation of the behavior of complex systems and the other is simulation exercises in which the learner reproduces the animation trace.

Today, the primary use of algorithm animation has been for teaching and instruction. The instructor executes the animation and the learner has a quite passive role in following the representation. From the pedagogical point of view, however, we believe that this is not enough. A large number of systems and reports written on this (Bridgeman 2000, Brown 1997, Carter 2003, Hansen 2000, Jarc 2000, Naps 2000, Naps 2003, Ross 2002) indicate that we should aim at *activating and engaging the learner* in order to promote the learning process. This, however, requires *feedback on students' performance*. The problem is how to provide feasible feedback in large courses that typically have limited resources.

Fortunately, the formal nature of algorithms and data structures allows us build learning environments in which we can compare the student's solution to the correct model solution easily. This gives an opportunity to produce systems that not only portray a variety of algorithms and data structures, but also distribute tracing exercises to the student and then evaluate the student's answer to the exercises. This is called *automatic assessment and feedback of simulation exercises*. One such a system is TRAKLA2 that we have employed with good results during the past few years (Laakso 2005).

Our previous research showed that there seem to be no significant differences between learner groups that solve the *same* exercises on the web and in the class room (Korhonen 2002). However, not all the exercises are such that they can be automatically assessed. Thus, the question is how far we can count on the automatic assessment today, and what would be the most feasible way to establish such a course organization that can best utilize the current learning environments in practice.

In this paper, we report on the preliminary results of the intervention study carried out in two different universities in Finland during the spring term 2005. In both universities, the TRAKLA2 learning environment was employed in the data structures and algorithms courses. The students (N=133(UTU) +134(HUT) = 267) were divided into two randomized exercise groups in both universities. The first group started web-based exercises with the TRAKLA2 learning environment while the second group did their exercises in the class room sessions. The research aimed at repetition of the intervention study carried out at the Helsinki University of Technology (HUT) in 2001 (Korhonen 2002). However, there were some differences as well. This time, we wanted to provide the same learning experience for all the students in order to prevent the drop out consequent upon the research setup. Thus, at the midpoint of the course, the groups transposed their places, the first group switched to the class room and the second group on the web. Moreover, we also report on the repetition of the attitude survey carried out at the University of Turku (UTU) in 2004 (Laakso 2005). This time, the same survey was carried out in both of the aforementioned universities.

In Section 2, we describe the TRAKLA2 learning environment as well as the course syllabus for both of the university courses and the descriptions of research methods are presented. In Sections 3 and 4, we report the results of the intervention study, and the attitude survey carried out, respectively. Finally, in Section 5 we make the conclusions based on the results achieved.

2. BACKGROUND AND METHOD

2.1 TRAKLA2 Learning Environment

TRAKLA2 is a learning environment providing algorithm simulation exercises in which students simulate the working of algorithms covered in the data structures and algorithms course. In algorithm simulation, the student must show in terms of conceptual diagrams how an algorithm changes given initial data structures during its execution.

The exercise could be, for example, "*Insert the following keys in this order into an initially empty binary search tree*: O, J, C, W, X, K, B, E, L, A, I, Y and R. *Draw the tree after each insertion*. The exercises are solved with a Java applet, i.e., students change the contents (keys and references) of data structures visualized on the screen by performing drag-and-drop operations supported by the TRAKLA2 environment[1]. An example exercise is seen in Figure 1. The student should drag and drop the keys from an array into the appropriate positions in the binary search tree. In the example, the first 7 keys are already inserted into the tree. The model answer in the front window shows the next state where also the key E is inserted.

Each student is given a random initial data structure instance to work with like the stream of keys in the previous example. The sequence of performed operations is recorded and the submitted answer is assessed automatically by comparing it to the model sequence generated by a real implemented algorithm. The student receives immediate feedback on his solution that indicates the number of correct steps out of the maximum number of steps. In addition, the model solution that is represented as an algorithm animation can be reviewed at any time.

[1] http://www.cs.hut.fi/Research/TRAKLA2/

If a student does not provide a correct solution, the exercise can be solved and resubmitted again. However, each time the exercise is reseted, a new instance of it with new initial data set is created. In general, the number of resubmissions can be unlimited as one cannot continue with the same data set. Moreover, the solution space of the exercises is far too large to allow brute force solution with simple trial and error method.

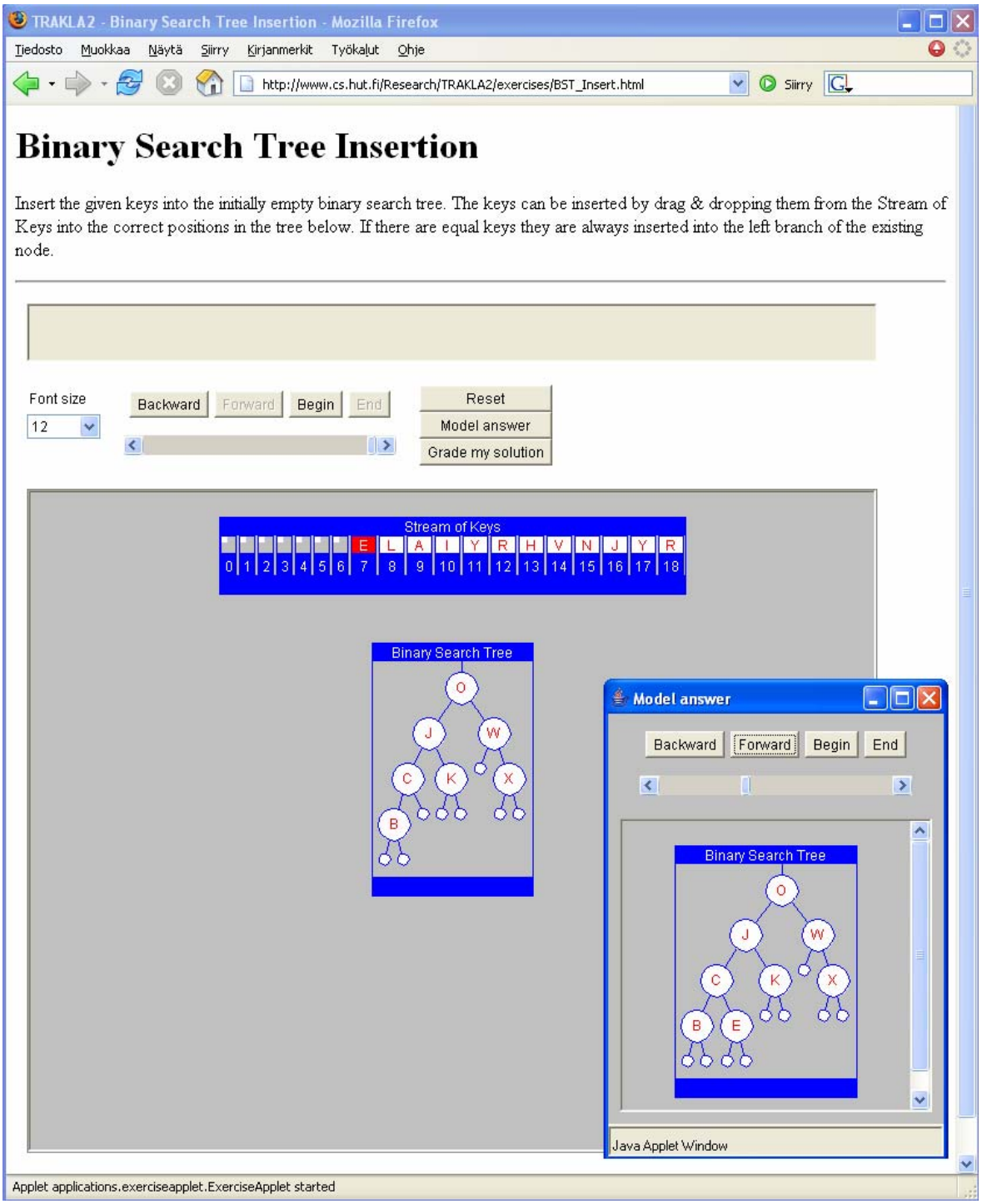


Figure 1. The TRAKLA2 exercise window comprises the data structures and push buttons. The model solution window is open in the front

Currently, approximately 30 difference exercises are included in the TRAKLA2 environment. These cover basic data structures, priority queues (i.e. heaps), sorting algorithms, hashing, dictionaries and graph algorithms. We emphasize here that students do no coding while solving the simulation exercises. They operate purely at a conceptual level. However, to be able to solve these exercises, they have to understand the key principles of the corresponding algorithms.

In the learning environment, the exercises are divided into exercise rounds as it is typically the case in the classroom exercises as well. Each exercise round covers one or more topics in the course. Moreover, the exercise rounds have deadlines as in traditional classroom exercises.

2.2 Course at UTU

In spring 2004, the DSA-UTU course (UTU04) included 56 lecture hours, 10 classroom sessions (each 2 hours), and 22 TRAKLA2 exercises. The classroom sessions consisted of four to six single exercises, such as illustrating exercises, proof exercises, etc.

Our previous study (Laakso 2005) reports very positive results on utilizing TRAKLA2 system in DSA-UTU course. Due to this, it was decided to use TRAKLA2 more extensively in DSA-UTU course in spring 2005 (UTU05). All together, UTU05 course included 56 lecture hours, 10 classroom sessions (each 2 hours), and 14 or 15 TRAKLA2 exercises depending on the group. The TRAKLA2 exercises replaced 4 class-room sessions (out of the 10), thus the number of classroom exercises decreased from 50 to 30 and were replaced by TRAKLA2 exercises. However, TRAKLA2 exercises did not cover all the areas of the replaced classroom exercises such as algorithm analysis, design exercises, proofs, and so on.

There were 2 classroom sessions for both groups at the beginning of the UTU05 course. After that the student were randomly divided into two groups A and B. Group A did 14 TRAKLA2 exercises in three rounds on the web, and at the same time, Group B did 4 classroom sessions (20 single exercises) covering the topics in the first half. The roles of the groups were reversed in the middle of the course after which Group A did 4 classroom sessions, and Group B did 15 TRAKLA2 exercises in two rounds covering the rest of the topics. However, also in this case, the classroom exercises included simulation exercises similar to the TRAKLA2 exercises as well as more challenging exercises such as analysis, design, and proof exercises.

The nature of the classroom sessions is such that every student has to do the given exercises beforehand and at the beginning of the class room session every student informs the demonstrator, which exercises he has solved. Then every exercise is presented sequently by selected students at the blackboard. After that the presented solution is analyzed by the demonstrator which gives feedback on students' performance. Furthermore, the demonstrator presents the correct solutions as well as clarifies the most important aspects of the exercises.

2.2.1 Grading and Requirements at UTU

It was possible to get 40 course points from the examination(s). In addition, any student can get additional 6 course points based on their activity in TRAKLA2 exercises and classroom exercises. These points were summed up to get the total number of course points. The final grading was on a scale from one to three with 0.25 steps. By getting 20 course points the student got the lowest grade.

The 6 extra course points were determined by the TRAKLA2 exercise points (maximum 35) and classroom exercise points (maximum 35) that were summed up. This gives a maximal value of 70 total exercise points (TEP). The conversion of TEP to the course points was linear between the minimum requirements 40% (pass with zero course points) and 100% points (6 course points, i.e. 15% of the maximum). The minimum points were required in order to get the examination.

There were two ways of passing the course: taking the final examination (0-40 course points) or taking two midterm-examinations (both of 0-20 course points). In either case, the students still had to fulfill the minimum requirements: i) to get at least 40% of the classroom exercise points (maximum 35), ii) to get at least 40% of the TRAKLA2 points (maximum 35 points), and iii) to get at least 20 course points out of the total of 46. The first midterm examination was arranged in the middle of the UTU05 course and the second one at the end of the course.

2.3 Course at HUT

The course syllabus at HUT was similar to that at UTU, but there are many differences as well. At HUT, the students have passed only one programming course before attending data structures and algorithm course. At UTU, however, they have a little bit more preliminary courses.

HUT05 course included 40 lecture hours and 7 rounds of exercises. Some 140 students were enrolled in the course including a small number of foreign students that were excluded from the research setup as they could not follow the lectures given in Finnish only. Each student selected an exercise group from the 5 possible alternatives. The first exercise session was common for all of the students, but after that each group was randomly split into two subgroups A and B (based on the last digit of the study book number). Group A did the exercise rounds 2–4 on the web with TRAKLA2 and 5–7 in the class room, and group B vice versa. Each exercise round comprises some 4–6 different simulation exercises. The exercises were the same in both groups, i.e., also the classroom exercise group did simulation exercises and not, for example, analysis or design exercises as they did in UTU.

2.3.1 Grading and Requirement at HUT

The midterm examinations and the compulsory exercises were assessed separately. Both had the 50% influence to the final grade. The conversion of points received from the exercises was linear between the minimum requirements 50% (pass with the grade 1) and 90%-100% points (pass with the maximum 4 grade 5). However, the students did not have to fulfill the minimum requirements before attending the examinations as it was the case in UTU. They had the chance to complete the exercises also after the examination in HUT.

2.4 Drop Out and Performance Evaluation

In this study, we have monitored two randomized groups of students (A and B) while they solved exercises in course of data structures and algorithms. In group A, the solving procedure started with web-based exercises while in B they practiced in a classroom. In the middle of the course, the procedure was changed: group A continued in the classroom while the group B went on the web.

The same evaluation was carried out both in UTU and HUT during the spring term 2005. Both courses started with a pre-test in which questions on the basic data structures and algorithms were asked. There were no differences between the groups A and B in the learning results of this test. In addition, some of the questions were repeated in the first midterm examination in HUT. For example, a pre-test question concerning the visiting order of nodes in tree traversal algorithms shows that only 26% of students was able to connect the algorithms (pre-, post-, inorder as well as level order) to the corresponding list of nodes with the given binary tree representation. However, in the first midterm examination this number was over 90%.

We conclude that actual learning has taken place during the courses. Thus, the research question concerns more the other quality aspects of the learning besides the amount of learning: the overall throughput of the course (i.e., passed students) as well as the fine tuning of the course difficulty. We try to gather evidence to adjust the course difficulty at the level that is the most feasible according to the use of web-based exercises and overall learning results. Moreover, we study the question how we should introduce the self study material and exercises to the students in order to avoid large drop out in courses.

2.5 Attitude Study

Students' opinions about the TRAKLA2 system were gathered through a web-based survey at the end of the HUT course in 2003. 364 students answered, 80% of which gave an overall grade of 4 or 5 to the system on the scale 0 – 5 with 0.5 steps, where 5 was the best grade. The system was almost unanimously considered to aid learning and to be easy to use. In addition, a survey was arranged at UTU in spring 2004. This survey included three sets of questionnaires about students' attitudes and opinions about web-based systems, one at the beginning, one at the middle, and one at the end of UTU04 course.

The purpose of the survey was to gather information about students' attitudes towards and experiences and changes in those during the UTU04 course. The first and the last questionnaire also included a question about what kind of exercises the students would prefer to do in DSA-UTU courses as well as a question on

how they assessed their own learning. There where also possibility to give free comments about the TRAKLA2 system. Indeed, the free feedback from this survey was well in line with the observations from the HUT survey.

The results from study in 2004 (Laakso 2005) shows that the TRAKLA2 system was well-approved by students, and we may say can be said that the students' attitudes became more positive towards web-based systems. Moreover, the results indicated that web-based exercises constitute a good amendment to courses on DSA and suggest that the mixed-alternative is far the most appropriate way to learn topics of courses on DSA.

As a continuation on this survey we wanted to confirm these observed results and it was decided to arrange a repetition study about the attitudes and opinions of the UTU and HUT students using questionnaires in spring 2005. In UTU05 and HUT05 course, a survey was arranged with two sets of questionnaires, one at the beginning and one at the end of each course. Both questionnaires included same questions as in the UTU04 survey. In both questionnaires, the students were asked to state how well they thought web-based exercises could suite to the course, and how much students believe that web-based learning could help them to understand DSA topics better. In addition, in the second questionnaire, there was a question in which students were asked to give their preferration about the role of the TRAKLA2 exercises in DSA courses.

3. RESULTS OF PERFORMANCE EVALUATION

In the following, we focus on the overall throughput of the passed students in the two courses involved both in UTU and HUT. The null hypothesis was that there is no significant difference between the groups A and B. In addition, we have assumed that the learning results must be the same in both of the groups. This was due to the fact that in HUT, the students did the very same exercises regardless of the solving procedure. Even though in UTU the class room exercises were more challenging, the students had a chance to do both kinds of exercises, thus giving them almost an equal learning experience. Performed tests supported this assumption. In UTU, for example, the average points received from the first midterm examination were 9.81 (group A) and 10.36 (group B). The same figures from the second midterm examination were 13.22 and 12.04, respectively. The midterm examinations revealed no statistical differences (t-tests, $p_1 = 0{,}58$, $p_2 = 0{,}33$), respectively.

Table 1. Course statistics of students taking the course exercises (CE) and midterm examinations (MTE). In addition, there is the number of students that passed/took the MTE. Finally, the throughput indicates the percentage of students that passed both CE and MTE.

	UTU A	UTU B	UTU Total	HUT A	HUT B	HUT Total
Number of students(#)	67	66	133	72	62	134
# passed CE	48 (72%)	39 (59%)	87 (65%)	52 (72%)	48 (77%)	100 (75%)
# took MTE	42 (63%)	38 (58%)	80 (60%)	52 (72%)	49 (79%)	101 (75%)
# passed MTE	29 (69%)	26 (68%)	55 (69%)	42 (81%)	42 (86%)	84 (83%)
Throughput	43%	39%	41%	58%	65%	61%

Table 1 summarized the basic data from both of the two courses. The total number of enrolled students in both courses (UTU 133, and HUT 134) was almost equal. In addition, there were no significant differences among the passed students of TRAKLA2 exercises except in UTU-B (t-test, $p = 0.02$). Thus, there was some evidence to reject the hypothesis that the level of difficulty in exercises makes no difference. Actually, it was quite natural that students (in group B) starting with more challenging exercises drop the course early more easily. However, it was interesting to note that there is no similar effect in HUT (t-test, $p = 0{,}80$). Contrary to this, the relative number of passed students was higher in group B. More students drop the course in group A, i.e., this confirms our previous results that in self study, there was a slight tendency to drop the course more easily than if the students were allowed to attend class room sessions (if the exercises are the same) (Korhonen 2002).

In UTU, the students were not allowed to take the 2nd midterm examination until they have passed all the exercises, which was not the case in HUT. Actually, there were 9 students more taking the midterm

examinations than passing the exercises in HUT (the number is not visible in Table 1). It should be noted, however, that in both courses there was an option to complete the performance in exercises by solving more exercises. Thus, the surplus students in examination were possibly planning to do the exercises after the examination. This seemed to be a bad strategy seeing that only 2 out of the 9 students passed the midterm examinations.

The midterm examinations were more difficult in UTU than in HUT. Only 69% passed in UTU while 83% passed in HUT. However, there are no differences between the groups A and B in either of the courses. In addition, it should be noted that midterm examinations was the first possibility to pass the course. There were a couple of more upcoming examinations in both of the courses later this year. Thus, for example in HUT, the overall throughput will be around 80%.

4. RESULTS OF ATTITUDE STUDY

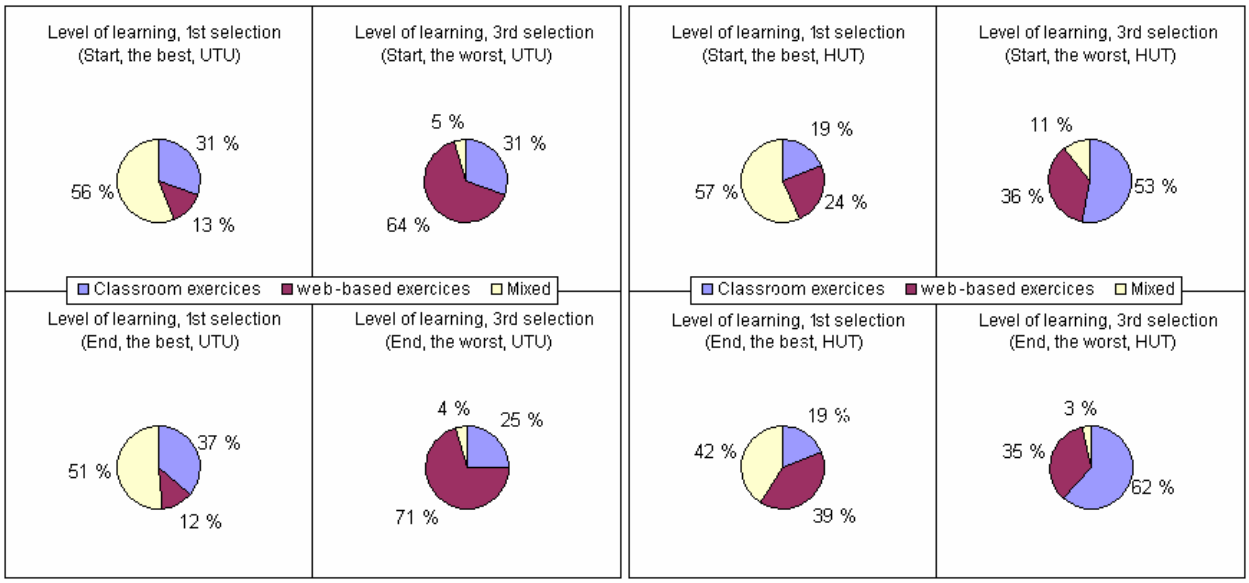
In the following, we present the detailed analysis of the attitude survey results in HUT and UTU in spring 2005. In addition, the opinions on learning results based on students' self evaluation are also presented.

At UTU, there where 108 answers to the first questionnaire ('Start') and 68 to the second questionnaire ('End'). At HUT, the same numbers were 129 and 89, respectively.

At the Start and End, the students were asked about their opinion on the suitability of web-based exercises for learning DSA. The scale was from 1 to 7 (7 being the best). The Start averages were quite high, 5.44 (UTU) and 5,54 (HUT), and the End averages were even higher, 5.59 (UTU) and 6.15 (HUT). The same phenomenon was observed in UTU04 course (Laakso 2005). Furthermore, the students were asked about how much students believe that web-based learning could help them to understand DSA topics easier (same scale as above). The Start averages were also quite high, 4.80 (UTU) and 4.89 (HUT), and the End averages also higher, 4.97 (UTU) and 5.49 (HUT).

These observations clearly indicate that web-based exercises are perceived to be suitable for learning DSA course topics and those exercises aree well accepted and approved by students both at UTU and HUT. Moreover, it can be also said that web-based exercises aids students' learning process of DSA course's topics.

In addition, the students were asked to assess the level of their learning on three alternative ways of doing exercises: traditional classroom exercises, web-based exercises or a mix of these two approaches (see Figure 2)



(a) (b)

Figure 2. This figures presents the first and the third selection in the level of learning (Figure 2 (a) UTU05 and Figure2 (b) HUT05). The upper pie charts illustrate the attitude at the beginning of the course and the lower pie charts at the end.

In Figures 2a and 2b we can see that at the Start, there is approximately the same amount of students in UTU and HUT courses that choose the mixed alternative on the 1st selection when looking the most appropriate way of learning. Furthermore, mixed alternative was considered the most appropriate way to learn DSA course's topics and the least appropriate was doing only web-based exercises. However, there was a quite big difference between UTU and HUT on the 3rd selection, which indicates that HUT students are more open-minded towards web-based exercises or learning. The natural reason for this is the fact that there have been much more usage of web-based exercises at HUT (since 1993) than at UTU (since 2004) in the previous DSA courses. Moreover, there have been only classroom exercises in DSA-UTU courses before UTU04 course that first time introduced TRAKLA2 system.

At the End, changes were quite different at UTU than at HUT. At UTU, mixed alternative loosed some ground to classroom exercises while web-based exercises stood at the same level. And the same statistic from UTU04 course (see Figure 3) shows that mixed alternative actually gained a lot of ground (from 48% to 73%). We suggest that this different behavior can be explained by the fact that in UTU04, the web-based exercises replaced only such exercises that are much more elegantly done in TRAKLA2 system like, for example, all the tracing or illustrative type of exercises (i.e. how a specific algorithm works on given input). In UTU05 course, however, TRAKLA2 exercises did not cover all the areas of replaced classroom exercises. Therefore, web-based exercises gave a little bit narrower conception of the topics compared with the classroom exercises that included also analyzing and proof exercises as well.

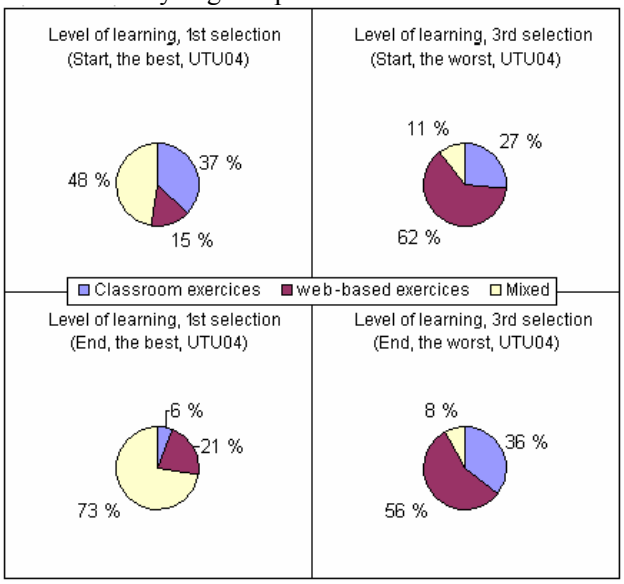


Figure 3. This figure presents the first and the third selection in the level of learning (UTU04). The upper pie charts illustrate the attitude at the beginning of the course and the lower pie charts at the end.

At HUT, the share of the web-based exercises grew from 24% to 39% (1st selection) almost catching up the mixed alternative. The growth is explained by the nature of the classroom exercises at HUT. Students did exactly the same exercises both in TRAKLA2 system and in the classroom with pen and paper. However, the most feasible way to do illustrative and tracing type of exercises is to do them in TRAKLA2 system. This same effect is observed also in (Laakso 2005) and explained why web-based exercises growth its share.

In the same manner as the level of the learning was studied, the students were asked to give their preference on the three alternative ways of doing exercises. The preferred alternative at Start was towards web-based exercises (1st selection: 44% (UTU) and 66% (HUT)). It turns out to be even stronger biased towards web-based exercises at the End (1st selection: 51% (UTU) and 75% (HUT)). In the UTU04 course, the change was about the same order from 38% to 44%. We can argue that the web-based exercises are the most preferred alternative to do the exercises according to the students.

Finally, at the End, the students were asked about the role of TRAKLA2 exercises. At UTU, 85% of students responded that TRAKLA2 exercises should be compulsory part of DSA-UTU course. The same amount at HUT was 96%. In addition, at UTU, 73% of students responded that TRAKLA2 exercises should be compulsory extra exercises or compulsory exercises to replace some of the classroom exercises. The same percentage at HUT was 55%. Still, 40% at HUT responded that they should be compulsory exercises to

replace all of the class room exercises. This observation is in line with other observation in HUT05. Only 15% (UTU) and 4% (HUT) of students responded that TRAKLA2 exercises should be voluntary extra exercises. This concludes that there is a need for both TRAKLA2 exercises and classroom exercises in DSA course. Moreover, while the simulation exercises are well suitable to be performed in the TRAKLA2 system, there are still many exercise types (i.e. analysis and proof exercises) that are better suited to the classroom environment, where human interaction takes place.

5. CONCLUSION

In this study, we have monitored two randomly selected groups of students (A and B) in two different universities while they solved exercises in course on data structures and algorithms. The first group did the exercises by web-based system and the second one in the class room sessions in both universities. The roles of the groups were changed in the middle of the course. In addition, we have surveyed their attitude towards web-based exercises. The students were asked whether they prefer to do the exercises in the class room or by web-based system, and the third alternative was a combination of these two.

There is no difference between groups A and B when looking at the overall throughput, i.e., passed students in the courses. Thus, the final examination (two midterm examinations in this case) seems to be equally difficult for both groups. We conclude that the procedure of doing exercises, i.e., web-based or class room does not have any influence on the learning results. However, if we look at the number of students passed the exercises, there is a statistical difference at UTU. This is due to the fact, that the students starting with more challenging class room exercises drop the course more often in the beginning of the course compared with those starting with the web-based exercises. This contradicts the previous results indicating that self learning causes the students to drop more easily. Thus, the difficulty level and nature of the exercises have more influence in this respect.

Based on the results from the prior attitude study in 2004 together with 2005 study, it can be said that the mixed alternative is the most suitable way to introduce the web-based exercises. The most preferred way to do simulation exercises is on the web, which indicates that the TRAKLA2 system is very easy to use and well-suited for its purpose. In addition, we argue that the web-based exercises should be compulsory in DSA courses and the exercises should replace only those classroom exercises that are suitable for TRAKLA2.

We believe that the results of these studies can be generalized to cover also other similar learning environments that can give automated feedback for the students. It seems that automatic feedback can be adequate enough to compensate its drawbacks compared with human guidance due to the fact that it is available all the time during the exercise session, thus allowing the students to study at their own pace. Moreover, it is not surprising that the results suggest us to introduce easy and human guided exercises first. After this, the students are more engaged into the course, thus they pass the whole course more probably, and can be directed to self-learning environments such as TRAKLA2.

The results also imply that the learning in such a learning environment is as good as in traditional sessions if the exercises are the same. Thus, keeping in mind the limits of automatic assessment, we can recommend the use of the web-based learning environments. In our case, this means that we are going to deliver all the simulation exercises through the web-based learning environment, but parallel to this, we have traditional classroom sessions as well to cover the design and analysis exercises that are beyond the scope of automatic assessment.

We have concluded that currently there is a need for both TRAKLA2 and traditional classroom exercises. There is a set of exercises that are best suitable to be automatically assessed by TRAKLA2. These include not only *tracing exercises*, but also *exploration exercises* (Korhonen 2004) such as coloring a binary tree to a red black tree. Thus, an important challenge of the future is to develop novel types of TRAKLA2 exercises to cover more of the traditional exercises. Our future plans are to do this in collaboration between Helsinki University of Technology and University of Turku.

REFERENCES

Brown, M. H. and Raisamo, R., 1997. JCAT: Collaborative active textbooks using Java. *Computer Networks and ISDN Systems*, Vol *29,* No 14, pp 1577–1586.

Carter, J. et al, 2003. ITICSE working group report: How shall we assess this? *SIGCSE Bulletin* Vol 35, No 4, pp 107–123.

Hansen, S. R. et al, 2000. Helping learners visualize and comprehend algorithms. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning,* Vol *2*, No 1.

Laakso, M.-J. et al, 2005. Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system TRAKLA2. *Informatics in Education,* Vol 4, No 1, pp 49–68.

Naps, T. L. er al, 2003. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin,* Vol 35, No 2, pp 131–152.

Bridgeman, S. et al, 2000. PILOT: An interactive tool for learning and grading. In: *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*. ACM Press, New York, pp. 139–143. URL http://citeseer.ist.psu.edu/bridgeman00pilot.html

Jarc, D. J. et al, 2000. Assessing the benefits of interactive prediction using web-based algorithm animation courseware. In: *The proceedings of the thirty-first SIGCSE technical symposium on Computer science education*. ACM Press, New York, Austin, Texas, pp. 377–381.

Korhonen, A. and Malmi, L., 2004. Taxonomy of visual algorithm simulation exercises. In: Korhonen, A. (Ed.), *Proceedings of the Third Program VisualizationWorkshop.Warwick*, UK, pp. 118–125.

Korhonen, A.et al, 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, New York, Aarhus, Denmark, pp. 121–124.

Naps, T. L. et al, 2000. JHAVÉ: An environment to actively engage students in web-based algorithm visualizations. In: *Proceedings of the SIGCSE Session*. ACM Press, New York, Austin, Texas, pp. 109–113.

Ross, R. J. and Grinder,M. T., 2002. Hypertextbooks: Animated, active learning, comprehensive teaching and learning resource for the web. In: Diehl, S. (Ed.), *Software Visualization: International Seminar. Springer*, Dagstuhl, Germany, pp. 269–283.

Paper 4

Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels.

Journal of Educational Technology and Society. 12(2), 267–282.

Laakso, M.-J., Myller, N., & Korhonen, A. (2009). Comparing Learning Performance of Students Using Algorithm Visualizations Collaboratively on Different Engagement Levels. *Educational Technology & Society*, *12* (2), 267–282.

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

Mikko-Jussi Laakso[1], Niko Myller[2] and Ari Korhonen[3]

[1]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

[2]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

[3]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

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. (in press) 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.

Previous Research

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. (in press) 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).

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. (in press) 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.

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.

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.

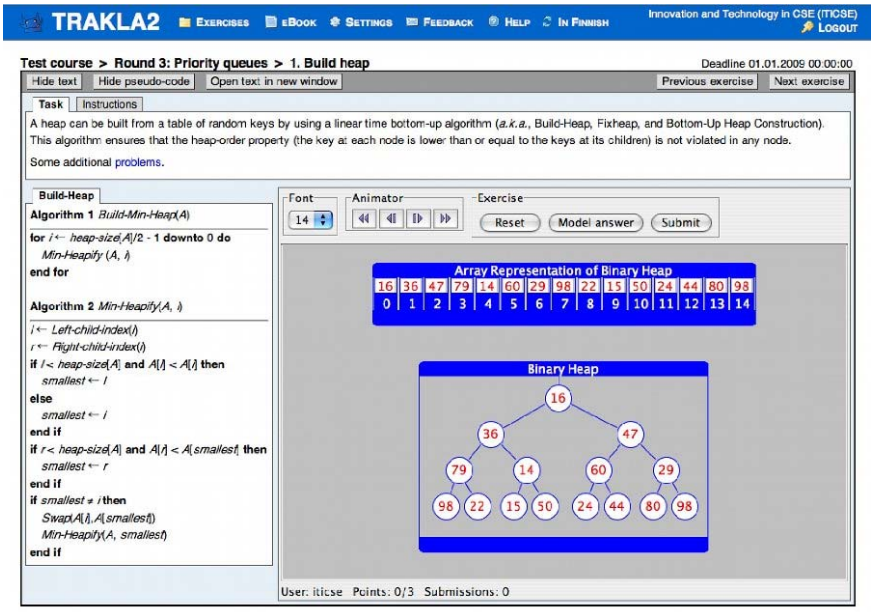


Figure 1: TRAKLA2 exercise page. The student acts in EET level *changing* by solving the exercise in terms of swapping the data elements in the data structure(s)

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).

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).

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).

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.

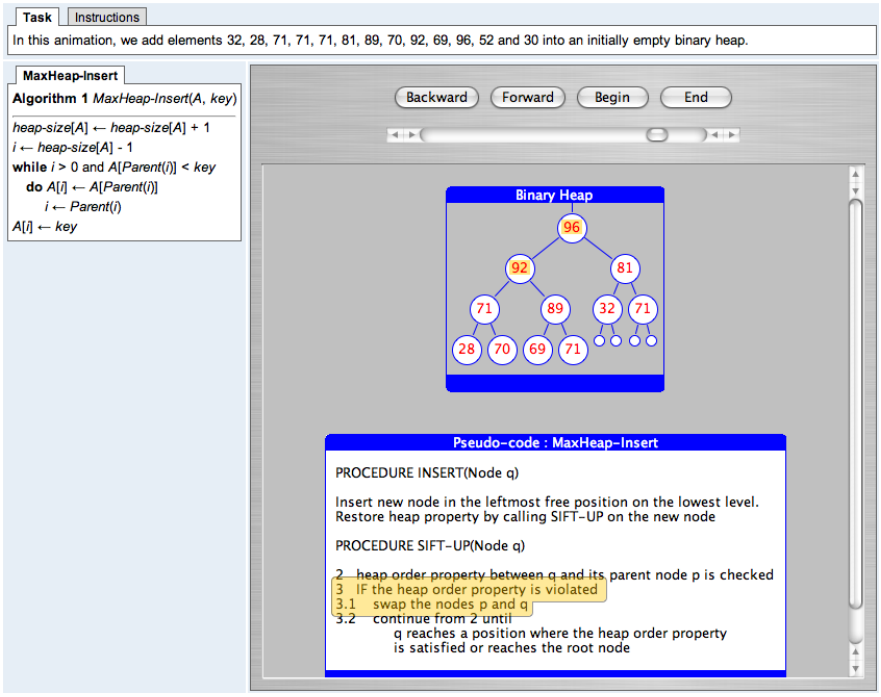


Figure 2: Binary heap insert animation in the tutorial. The student acts on EET level *controlled viewing*. The user has VCR like buttons (Backward, Forward, Begin, End) to interact with the animation

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).

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.

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.

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).

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.

Results

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).

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.

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

	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 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?

Table 2: Motivation of students based on a questionnaire. *Note*. Questions Q1 to Q4 are discussed in the text

	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)

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

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.

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.

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.

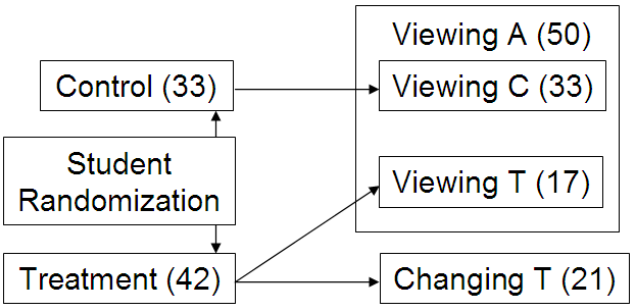


Figure 3: The division of the groups

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.**.

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.

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

	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 4 shows the results from the same motivational questionnaire that was also reported in the Table 2 for the experimental groups (See Section 0 for the description of the questions). None of the differences were statistically significant.

Table 4: Motivation of students based on a questionnaire. *Note*. Questions from Q1 to Q4 are discussed in Section 0

	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)

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.

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

	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 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.

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

	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)

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).

Table 7: Post-test results. *Note*. Post-test questions were discussed in Section 0 and compostion of the groups in Figure 3

	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)

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.

Discussion

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., in press). 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.

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.

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.

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.

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*, Atlanta, GA: Graphics, Visualization, and Usability Center, Georgia Institute of Technology.

Collazos, C., Guerrero, L., Redondo, M., & Bravo, C. (2007). Visualizing Shared-Knowledge Awareness in Collaborative Learning Processes. *Lecture Notes in Computer Science, 4715*, 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 Ed.)*, Upper Saddle River, NJ: Allyn & Bacon.

Grissom, S., McNally M., & Naps, T. L. (2003). Algorithm visualization in CS education: comparing levels of student engagement. *Proceedings of the First ACM Symposium on Software Visualization*, New York: 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? *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education*, New York: 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*, Helsinki: 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. *Proceedings of Cognition and Exploratory Learning in Digital Age*, Lisbon: 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. *Proceedings of the 25th International Conference on Software Engineering*, Los Alamitos, CA: IEEE Computer Society, 602–607.

Myller, N., Bednarik, R., Ben-Ari, M., & Sutinen, E. (In press). Extending the Engagement Taxonomy: Software Visualization and Collaborative Learning. *ACM Transactions on Computing Education*.

Myller, N., Laakso, M., & Korhonen, A. (2007). Analyzing engagement taxonomy in collaborative algorithm visualization. *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, New York: 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. *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, New York: 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. *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, New York: 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*, London: Taylor & Francis, 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. *Proceedings of the SIGCHI conference on Human factors in computing systems*, New York: ACM Press, 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.

Paper 5

Rajala, T., Laakso, M.-J., Kaila, E. and Salakoski, T. (2008). Effectiveness of Program Visualization: A Case Study with the ViLLE Tool.

Journal of Information Technology Education: Innovations in Practice, 7, IIP 15-32.

Effectiveness of Program Visualization: A Case Study with the ViLLE Tool

Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, and Tapio Salakoski
University of Turku, Turku, Finland

temira@utu.fi; milaak@utu.fi; ertaka@utu.fi; sala@utu.fi

Executive Summary

Program visualization is one of the various methods developed over the years to aid novices with their difficulties in learning to program. It consists of different graphical – often animated – and textual objects, visualizing the execution of programs. The aim of program visualization is to enhance students' understanding of different areas of program execution. Typical program visualization techniques include code highlighting, visualization of the call stack, and presenting information on variables. Despite the large number of studies performed on program visualization, little is known about the effects of such systems on learning.

We have developed a program visualization tool called ViLLE, with the main objective of offering an environment for students to study the execution of example programs – whether written by students themselves or prepared by the teacher – and explore the changes in the program state data structures. A key feature of ViLLE is language independency, including parallel execution of a program in two different languages and the ability to define new languages. ViLLE also provides role information of program variables and supports the design and use of interactive pop-up questions.

In this paper, we report and discuss the results of a study on the effectiveness of ViLLE. The research was conducted on university students in their first programming course. Students participated in a two hour session in a computer class, where they were randomly divided into two groups. The control group used only traditional textual material during the session, whereas for the treatment group, the same material was extended with interactive examples using ViLLE. With this research setting, we tried to answer two research questions: "Does ViLLE help students in learning to program?", and "Is there any difference in learning when previous programming experience is taken into account?" We found some support for a positive answer to the first question, although we couldn't fully reject the null hypothesis. For the second question, we obtained solid evidence that ViLLE enhances the learning of students with no prior programming experience substantially, so that the statistical differences between the novices and the more experienced learners disappeared as a result of a single training session. This indicates that program visualization indeed improves novice students' learning.

Keywords: program visualization, novice programmers, effectiveness of visualization, programming, programming learning, programming teaching.

Editor: Kam Vat

Introduction

Learning to program is not an easy task. According to multi-national studies published in recent years, students have problems in writing program code (McCracken et al., 2001), in reading and tracing skills (Lister et al., 2004), and in designing software (Tenenberg et al., 2005). Since constructing and even understanding computer programs have proven to be a highly non-trivial task for most learners, various techniques and means have been suggested to aid the learning process of beginner programmers. Visualization – generally defined as presenting the execution of program or algorithm with graphical components – is one of these. According to Ben-Ari (2001) visualization includes everything even remotely graphical, from complex animations to indentation of program blocks, and for the effective use of visualizations, the textual and graphical descriptions have to be synchronized. Hyrskykari (1993) states that visualizations can be useful in providing learning models that can be used in linking new information with old knowledge.

Program visualization is a research area that studies ways of visually assisting learners in understanding behaviour of programs. The visualization of programs can be either dynamic or static. Dynamic program visualization tools visualize execution of programs. They usually show how the execution of programs progresses by highlighting parts of the code under execution and by visualizing changes in variable states. An example of a dynamic program visualization tool is Jeliot3 (Moreno, Myller, Sutinen, & Ben-Ari, 2004). Static visualization tools visualize program structures and relations between program objects. An example of a popular static program visualization tool is BlueJ (Kölling, Quig, Patterson, & Rosenberg, 2003).

We have recently developed a dynamic program visualization tool called ViLLE (Rajala, Laakso, Kaila, & Salakoski, 2007). ViLLE is a language-independent visualization tool aimed at providing a more abstract view of programming. The tool can be utilized both in lectures and for independent learning. It has a built-in syntax editor, with which the user can add new languages to the tool or modify the syntax of the built-in languages, currently including e.g. Java, C++, and a pseudo language. The visualizations can be viewed in any of the (user or pre-) defined languages. To emphasize the language independency, ViLLE has a parallel view in which the execution of a program and the program code itself can be viewed simultaneously in two languages. While the execution progresses, the user can observe program outputs and changes in variable values. In addition, to enhance the effectiveness and clarity of the visualization, there is an automatically generated textual description of each code line. The description also includes information about the roles of variables (Sajaniemi, 2002). However, according to Nikula, Sajaniemi, Tedre, and Wray (2007), to get the most benefit from the roles of variables, they should be employed in all aspects of teaching.

The goal of this paper is to find out what kind of effects ViLLE has on programming learning with following research questions: "Does ViLLE help students in learning to program?" and "Is there any difference in learning when previous programming experience is taken into account?". To study these questions, we conducted a study in the first programming course at the University of Turku, Finland, in fall 2007.

This paper has the following structure. In the next section we consider previous work on program and algorithm visualization. In the third section, ViLLE and its key features are described. The research design and results are presented in the fourth and fifth sections, respectively. These are followed by a section in which the results are discussed and, finally, conclusions and future directions are presented.

Related Work

Many visualization systems have been developed over the past few decades. These include JavaVis (Oechsle & Schmitt, 2002) which visualizes object and sequence diagrams, one based on WYSIWYC (What You See Is What You Code) model and direct manipulation of program structures called ALVIS LIVE! (Hundhausen & Brown, 2007), and Raptor (Carlisle, Wilson, Humphries, & Hadfield, 2005) a visualization tool that utilizes dataflow diagrams. The main part of the development in this field is focused on algorithm animation, which visualizes data structures and algorithms. Notable algorithm animation tools include JHAVE (Grissom, McNally, & Naps, 2003), BALSA-II (Brown, 1988), ZEUS (Brown, 1991), XTANGO (Stasko, 1992), and TRAKLA2 (Malmi et al., 2004).

Boyle, Bradley, Chalk, Jones, and Pickard (2003) paid particular interest in a 'visual approach' – portraying the abstract programming concepts with graphical shapes – while defining the new curriculum for London Metropolitan University's course of introductory programming. Over 600 students took part on the course, and the results of the new 'blended learning environment' were quite promising; according to a questionnaire answered at the mid-semester stage more than 80 % of students described their motivation level as high or very high, and over 70 % were happy or very happy about their progress in studies. The increase in pass rates was between 12 and 23 % compared to previous year. Boyle et al. reported some major issues in handling the course transition, but on average they described the graphical approach 'very successful with the students' (Boyle et al., 2003, p. 177).

Kannusmäki, Moreno, Myller, and Sutinen (2004) evaluated the use of the Jeliot 3 program visualization system during the second course of programming in the Virtual Studies of Computer Science distance learning program at the University of Joensuu, Finland. The emphasis was on ways of using the tool and on features students would like to have included in the tool. The qualitative data was collected from the course's discussion forum messages. Gathered data was divided into three categories: usage patterns, usage problems, and opinions and suggestions. Messages in the first category revealed that the students most successful in the course used Jeliot more than the other groups. However, most of the students in general still used other tools to code and test their programs. The usage problems reported were mostly technical or related to the usability of the editor. The animation was criticized on being too slow and some students even found the whole system unnecessary and unsuitable for advanced courses. The positive aspects identified in the feedback included the ability to make conditional statements, loops, and objects more understandable.

Hundhausen, Douglas, and Stasko (2002) conducted a comprehensive meta-study, analyzing 24 experimental studies on effectiveness of algorithm visualization. They state that one of the main reasons visualizations are not widely used is because the teachers responsible for the courses refuse to use new methods in teaching. They also found out that the main focus in articles about visualizations is normally on their graphical means of expression – in other words their visualization capabilities - instead of their learning benefits. Of the 24 studies examined, 11 showed statistically significant results of visualizations positive effects on learning, meaning that the group using a visualization system gained better learning results than the control group. Hundhausen et al. (2002) also discovered that the sole use of visualization systems doesn't necessarily improve the learning results; it is more important to engage the learners in the subject using visualization system as an aid.

Other studies concerning evaluation of visualization systems include, for example, studies (see Grissom et al., 2003, Laakso, Salakoski, Grandell, et al., 2005; Laakso, Salakoski, Korhonen, 2005) about adapting algorithm animation systems successfully in teaching, and a study about educational impacts of visualization (see Naps et al., 2003). Laakso, Myller, and Korhonen (in

press) studied the effectiveness of algorithm visualization system TRAKLA2 in different engagement levels. With a similar research setup to ours (two hour controlled experiment), they were able to confirm some of the hypotheses presented in the taxonomy of learner engagement with visualization technology (Naps et al., 2002).

ViLLE

ViLLE is a program visualization tool for teaching programming to novice programmers. Teachers can use the tool in lectures to demonstrate the dynamic behaviour of program execution, and students can use it independently over the web. ViLLE contains a predefined set of programming examples grouped into different categories based on their topic. Teachers can easily add new examples to the tool or modify the existing ones. The tool contains also a question editor with which the teacher can attach multiple choice and array related pop-up questions to program events of a chosen programming example. The pop-up questions are then shown to the students as they go through the execution of a programming example, engaging them more deeply in learning process.

ViLLE supports all the programming concepts generally featured in introductory programming courses. The support for more advanced concepts is limited: for example objects – excluding arrays, strings and records – are not supported. These limitations however make it possible to define new syntaxes with corresponding features to existing languages in ViLLE.

Key Features

This section presents ViLLE's key features divided into four categories: level of abstraction, user interaction, tracing execution, and customization. The categories reflect the main functions and features in this tool.

Level of abstraction

Language-independency. One of the most important aspects of ViLLE is the ability to view programming examples in several different programming languages. When observing program execution in different languages, a user can discover similarities in their basic functionalities. It is far more important for the novice programmer to learn how different programming concepts actually work than to focus on the syntactical issues of a specific language. We call this the *programming language independency paradigm*.

Defining and adding new languages. As built-in, ViLLE supports Java, pseudo code, and C++. The pseudo code's definition can be altered to suit a teacher's needs. It is also possible to define and add new programming languages to further extend the language support.

The parallel view. The program code execution can be viewed simultaneously in two different programming languages. This way the user can see how the execution progresses similarly regardless of syntactical differences between the languages.

Role information. The role information of variables is integrated into the code line explanation. According to Sajaniemi and Kuittinen (2003) the role information of variables helps learning and enhances understanding of programs.

User interaction

Code editing. Besides the example creation and editing view, the program code can also be edited in the visualization view, allowing users to trace the effects of changes in execution and visualization. The user's edits are not saved to the original program.

Pop-up questions. With the built-in editor the teacher can create multiple-choice questions and set them to be triggered at certain stages of the program execution.

Flexible control of the visualization both forwards and backwards. The user can move one step at a time, both forwards and backwards in the execution of a program. Examples can also be run automatically with adjustable speed. Moving backwards in the program execution isn't usually possible in similar applications. Additionally, ViLLE has an execution slider with which the user can progress to any state of the program execution.

Tracing execution

Call stack. The progress of the program execution between different methods due to function calls and returns is visualized with a call stack. When a method is called, a new window is opened on the call stack. The window remains on the stack until the method is finished. When the execution returns to the caller, the return value is shown on top of the stack. The call stack can be especially useful in learning recursion.

Code line explanation. Every code line has an automatically generated explanation, in which all the program events on the line are clearly explained. Furthermore, all possible outputs and variable states are shown. Code line explanation is not a feature in most similar applications.

Visualization row by row. Progress of the program execution is visualized by highlighting rows in the code. In addition to highlighting the program row under execution, ViLLE also highlights the previously executed row with a different colour. This makes the following of the program execution easier.

Breakpoints. The user can set breakpoints in program code lines and move between them both forwards and backwards. This functionality enables debug-based control and observation of the program execution. Backward tracing between breakpoints is not a standard feature in visual debuggers.

Customization

Example collection. ViLLE contains a predefined set of programming examples grouped into categories based on their subject. A user can create new categories and examples or edit the predefined ones. By creating and editing examples, the teacher can illustrate topics essential in his programming courses.

Publish examples. With the export feature ViLLE's examples can be saved to an example collection. The example collection contains a version of ViLLE with example creation and modification functions disabled; runtime modification however is still enabled. The export feature can be used to publish course's programming examples on the web for the students to use.

Visualization View

The visualization view of ViLLE (Figure 1) consists of three areas. The left side of the view contains the program controls and the program code of the current example. The controls can be used to move both forwards and backwards in the visualization. The right side of the view displays the call stack. Each method call creates a new window on top of the call stack, and as the execution of the method is finished, the return value is shown on top of the stack. The fields at the bottom of the view display an explanation of the current program line, program outputs and variable states. The programming example can also be edited in the visualization view to directly see how the modifications affect the execution. Additionally, the call stack area can be replaced with a large variable state visualization area, which visualizes arrays and matrices with graphical presentations.

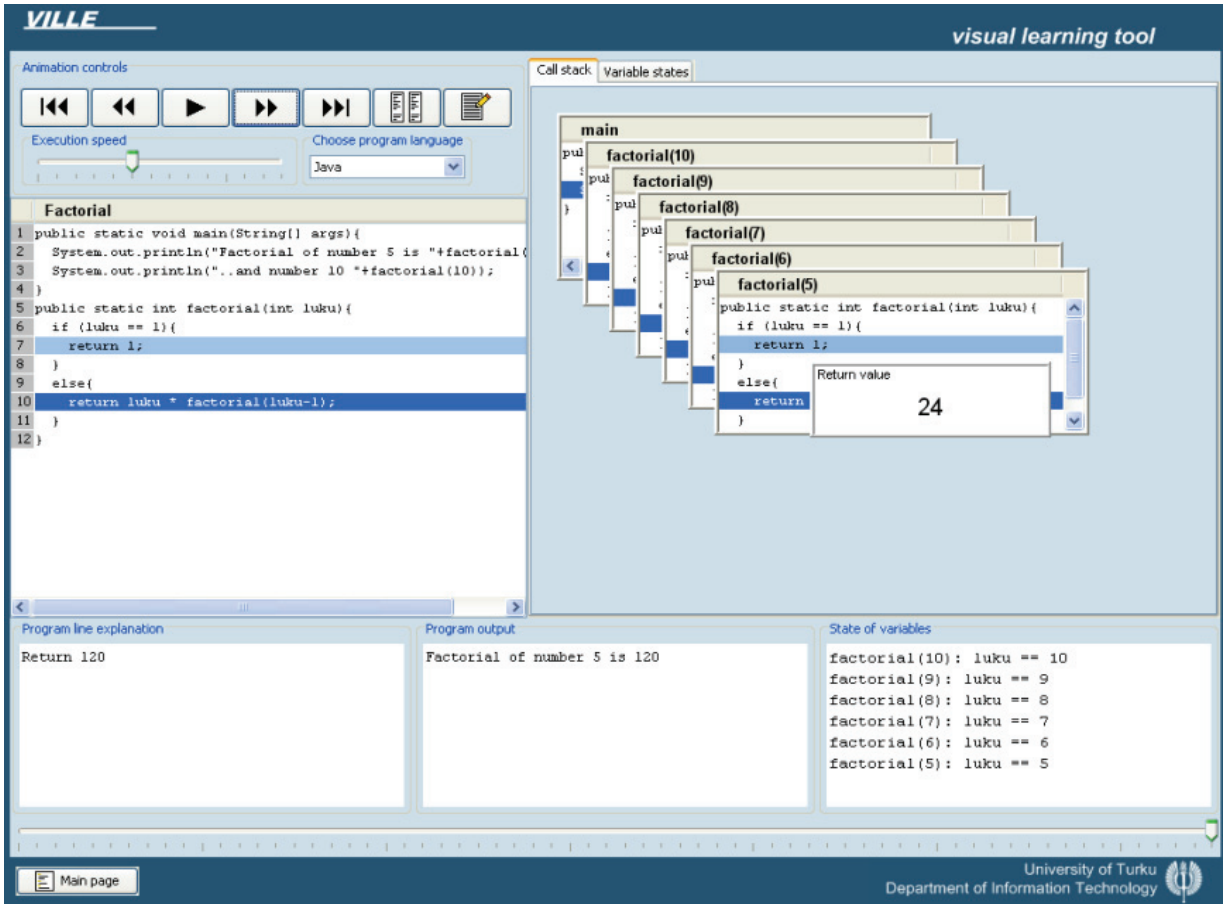


Figure 1: The visualization view in ViLLE

The main idea of ViLLE is to provide a language-independent and, thus, a more abstract view on programming. As built-in, ViLLE supports three programming languages (Java, C++, and a pseudo language) that can be used in the visualization of programs. A user can define a new programming language or modify the existing ones with the built-in syntax editor. This support for multiple languages enables simultaneous viewing of the program visualization in two different languages in parallel, which should help students in understanding the similarity between various programming concepts in imperative programming languages. Another abstraction of programming used in ViLLE is the concept of roles of variables (Sajaniemi, 2002). The role is a description of variable's behaviour in a program. In the visualization view of ViLLE, the program line explanation field also contains information about the roles of variables.

With the above features we try to demonstrate the importance of understanding how the programming concepts actually work in contrast to just learning some specific issues related to the syntax of programming languages. For more detailed information on ViLLE, see Rajala et al. (2007).

Research Design

We conducted an experiment in which we evaluated ViLLE's effectiveness in learning basic programming concepts. There were two main research questions in the study: 1) "Does ViLLE help students in learning to program?", and 2) "Is there any difference in learning when previous programming experience is taken into account?" The null hypotheses were that ViLLE doesn't aid

the learning of basic programming concepts, and the effect is the same for novice and experienced students, respectively.

The experiment was conducted in the third week of the first programming course at the University of Turku. The objective of the course was to learn how computers function logically and to understand the essential concepts of programming. An additional goal was the development of good program reading skills. The course consisted of 28 lecture hours spread over seven weeks. During the first two weeks, topics covered were related to information technology in general instead of programming specifically. Additionally, students had to return four programming assignments at the end of the course.

One two-hour lecture, in which e.g. the syntax of the programming language used was presented, was held before the experiment. A link to ViLLE and its examples was provided to students in the second week and the students were advised to use it before the experimentation. The reason for this was that we wanted them to be familiar with the syntax and the system – including its look and feel. The usage of the tool was however not included in the course's curriculum after the experiment.

The students were divided into two groups: the control group used a textual programming tutorial without access to ViLLE; the treatment group, however, could visualize the examples in the tutorial with the ViLLE tool. The results were analyzed with a two-tailed and pair-wise t-test. In addition, Levene's test was used to calculate the variance for every statistics to determine if the data holds equal or non-equal variances. Unequal variances are marked with '*'-character in presented tables.

Method

The experiment was a between subject design with a pre- and post-test (dependant variable). We had two between-subject factors (independent variables): previous programming experience and previous usage of ViLLE. Students acted alone during the experiment and answered individually to the pre- and post-test. Textual material (provided in a web page) was exactly the same for both conditions, and the only difference was that the treatment group was able to explore integrated examples with ViLLE.

Participants

The participants were university students who attended the first programming course presented in the curriculum. Most of the students were either Computer Science or Mathematics majors.

Students were randomly divided to computer lab sessions and the sessions were randomly assigned to the treatment or the control condition. The total number of participants was 72 ($n = 72$) students. There were 40 students in the control group and 32 students in the treatment group. More than half of the students didn't have any previous knowledge of programming. Moreover, there were three students that participated in the lab session, but who didn't give permission to use their results in this research. There were two lab sessions for each condition. Students who attended the lab session received two bonus points to their final exam results.

Materials

The *pre-test* consisted of three questions. In each question the students were presented a code fragment and asked to define the output or the state of the program. In the first question the program code presented contained three numeric variables and three consecutive conditional statements. The students were supposed to track down the changes in variables and type their values in different points of execution. In the second question the students were presented a loop in

which the values of two variables were changed and printed out. The students were asked to give the complete output of the program. The third question included a recursive function which calculated the sum of the sequence from given parameter down to 1. The assumption was that the students with some earlier programming experience would be able to solve at least some of the assignments. In addition, the students were asked some background information, including the amount of earlier programming experience on the scale of 0 to 4, programming languages they had used, and whether they had used ViLLE before taking the test.

After completing the pre-test the students went through *a programming tutorial* that we had prepared earlier. The tutorial consisted of few basic programming subjects – the same subjects the students were tested on with pre- and post-tests. The subjects covered (in this order) variable usage and manipulation, printing, conditional statements, loop statement (`while`-statement, to be exact), function calls, and finally recursive functions. There was a textual description on all topics with some examples on how to use them. The tutorial contained 14 programming examples, and the students were instructed to write down the output of each example on paper. This was to ensure that each student really went through the tutorial. The group using ViLLE could examine the execution of each example by selecting a link titled *'run this example'* next to it.

The *post-test* included all the questions of the pre-test in exactly the same form. In addition, there were two extra questions. In the first one the students were asked to complete the given program code so that it would output all the even numbers from 2 to 24. The template given consisted of `while` and `print`-statements without parameters and some blank fields with proper indentations for the students to fill in. The second question was a follow-up to the question about the recursive function: the students were asked to deduce the outcome of the same function with two different parameters.

Procedure

The study was performed in the third week of the seven week course at the computer lab sessions that lasted for two hours. The students were divided randomly into two groups. Both groups had the same programming tutorial, but the second group could additionally execute the examples in the tutorial with ViLLE. In the beginning of the session students took the pre-test independently. The time reserved for filling out the questions was 15 minutes.

After the pre-test each of the students used the programming tutorial for 45 minutes. To monitor the involvement, the students were instructed to write down the output of each example (14 in total). Students went through the tutorial independently; they were allowed to ask for assistance only if they encountered technical difficulties.

The session ended with answering the post-test. Since the post-test had two extra questions compared to the pre-test (and since the extra questions were more demanding) the time reserved for answering the questions was 30 minutes.

Each question in the pre- and post-tests was analyzed in the scale of 0 to 10. Zero points meant that the answer was totally wrong, and each point advanced meant the increase of 10 percent in the correctness of the answer. The total maximum in the pre-test was 30 points and in the post-test 50 points.

Results

Effectiveness of ViLLE

In this section we present results to research question related to the independent variable of using ViLLE. The treatment group used ViLLE in the lab session while the control group didn't use it

at all. There were 32 students in the treatment group and 40 students in the control group. The groups were randomly formed.

Previous knowledge

Table 1 presents the results from pre-test for the treatment and the control group. The table includes averages, standard deviations (in parenthesis) and p-values obtained from two-tailed t-test.

Table 1: Pre-test results

Question	Control (n = 40)	Treatment (n = 32)	p-value
Question 1 (Q1)	5.20 (2.67)	6.19 (2.46)	0.111
Question 2 (Q2)	2.70 (3.53)	2.13 (3.53)	0.494
Question 3 (Q3)	2.68 (4.15)	2.09 (3.88)	0.546
Total	10.58 (8.64)	10.41 (7.18)	0.930

There were no statistically significant differences between groups in any pre-test questions. In absolute scale, the control group outperformed treatment group in Q2 and Q3, while the treatment group achieved more points in Q1.

As stated earlier, students were advised to familiarize themselves with ViLLE's interface before the test; 20 students in the control group and 19 students in the treatment group reported having done this. There were no statistically significant differences inside or between the groups in pre-test results related to ViLLE's previous usage.

Previous programming experience

We also asked about students' previous programming experience and divided the treatment group and the control group based on this covariant (previous programming experience). The question's scale was from 0 to 4. Based on this gathered data we computed a new discrete (boolean) 0,1-variable for previous programming experience; 0 is equal to no previous programming experience (NPE) and all the other values were counted for some previous experience (SPE). Tables 2 and 3 present the pre-test results between following groups: 1) treatment and NPE vs. control and NPE 2) treatment and SPE vs. control and SPE, respectively.

Table 2: Pre-test results of students with no previous programming experience (NPE)

Question	Control (n = 23)	Treatment (n = 20)	p-value
Q1	4.17 (2.39)	5.60 (2.33)	0.041
Q2	1.22 (1.78)	1.00 (2.22)	0.724
Q3	1.00 (2.86)	1.65 (3.62)	0.514
Total	6.39 (4.68)	8.25 (5.44)	0.235

Table 3: Pre-test results of students with some previous programming experience (SPE)

Question	Control (n = 17)	Treatment (n = 12)	p-value
Q1	6.59 (2.53)	7.17 (2.76)	0.564
Q2	4.71 (4.31)	4.00 (4.51)	0.673
Q3	4.94 (4.62)	2.83 (4.36)	0.226
Total	16.24 (9.63)	14.00 (8.48)	0.524

There were no statistically significant differences between the treatment and control groups. Notice that in Q1 in Table 2, the seemingly significant p-value (0,041) does not indicate a statistically significant difference, because there were three questions in the pre-test and thus the p-value should be three times smaller (Bonferroni correction). Based on the data from the Tables 1, 2, and 3, we conclude that there is no difference between the control and the treatment group while comparing the pre-test data with or without the previous experience of programming.

Post-test results

The post-test included all the questions presented in the pre-test, with two additional questions. Table 4 presents statistics for the control group and the treatment group. In the first column (question) there is a correspondent pre-test question label. The table includes averages, standard deviations (in parenthesis) and p-values obtained from two-tailed t-test for each question. In addition, there are total points of shared questions (pre- and post-test), total points (post-test), differences between each question in the pre- and post-test and total difference.

We also calculated Cronbach's alpha reliability values for pre- and post-test questions. The results (pre-test $\alpha = 0{,}667$ and post-test $\alpha = 0{,}831$) indicate high reliability.

Table 4: Post-test results

Question	Control (n = 40)	Treatment (n = 32)	p-value
PQ1 (Q1)	6.30 (2.81)	6.13 (2.69)	0.790
PQ2 (Q2)	5.10 (4.35)	5.50 (4.50)	0.704
PQ3	6.28 (3.75)	5.88 (3.75)	0.654
PQ4 (Q3)	6.15 (4.56)	6.50 (4.42)	0.744
PQ5	7.05 (3.78)	6.69 (4.08)	0.698
Total (shared)	17.55 (9.08)	18.13 (8.81)	0.788
Total (all)	30.88 (15.20)	30.69 (15.08)	0.959
Diff PQ1	1.10 (2.60)	-0.06 (2.81)	0.073
Diff PQ2	2.40 (3.30)	3.38 (4.02)	0.262
Diff PQ4	3.48 (4.81)	4.41 (4.53)	0.405
Total diff	6.98 (6.81)	7.72 (6.76)	0.646

When comparing the shared questions in the pre- and post-test, we see that in absolute scale the control group outperformed the treatment group in PQ1 while the treatment group did better in PQ2 and PQ3. The better performance in PQ1 is related to the fact that achieved points were quite high in treatment group in pre-test. Still, the differences are too small to reject the null hypothesis. Similarly to the pre-test, the previous usage of ViLLE as a factor didn't reveal any statistically significant differences, either inside or between the groups.

The same statistics calculated with the previous programming experience taken into account are shown in Tables 5 and 6. ('*'-character indicates non-equal variances)

Table 5: Post-test results with NPE

Question	Control (n = 23)	Treatment (n = 20)	p-value
PQ1 (Q1)	5.74 (2.78)	5.90 (2.86)	0.853
PQ2 (Q2)	3.39 (3.97)	4.70 (4.58)	0.321
PQ3	5.30 (4.06)	5.05 (3.65)	0.831
PQ4 (Q3)	5.22 (4.83)	6.00 (4.71)	0.595
PQ5	6.09 (4.09)	6.05 (4.20)	0.977
Total (shared)	14.35 (8.27)	16.60 (9.29)	0.405
Total (all)	25.74 (14.44)	27.70 (15.49)	0.670
Diff PQ1	1.57 (2.48)	0.30 (2.62)	0.113*
Diff PQ2	2.17 (3.07)	3.70 (4.38)	0.189
Diff PQ4	4.22 (4.73)	4.35 (4.73)	0.927
Total diff	7.96 (5.80)	8.35 (7.98)	0.853

Table 6: Post-test results with SPE

Question	Control (n = 17)	Treatment (n = 12)	p-value
PQ1 (Q1)	7.06 (2.75)	6.50 (2.43)	0.577
PQ2 (Q2)	7.41 (3.81)	6.83 (4.22)	0.703
PQ3	7.59 (2.90)	7.25 (3.65)	0.783
PQ4 (Q3)	7.41 (3.94)	7.33 (3.94)	0.958*
PQ5	8.35 (2.96)	7.75 (3.82)	0.635
Total (shared)	21.88 (8.51)	20.67 (7.64)	0.696
Total (all)	37.82 (13.68)	35.67 (13.53)	0.678
Diff PQ1	0.47 (2.70)	-0.67 (3.11)	0.303
Diff PQ2	2.71 (3.65)	2.83 (3.46)	0.925
Diff PQ4	2.47 (4.87)	4.50 (4.38)	0.252*
Total diff	5.65 (7.98)	6.67 (4.14)	0.689

As seen in Tables 5 and 6, the previous programming experience had no statistically significant effect. The previous statistics are summarized in Table 7, including the averages from the pre-test, post-test, differences and p-values for the treatment group, the control group, treatment with NPE (TNPE), treatment with SPE (TSPE), control with NPE (CNPE), and control with SPE (CSPE). The p-value is obtained by comparing total points from the pre- and post-test in shared questions with a pair-wise t-test.

Table 7: Pre- and post-test results

Points	Control (C)	Treat-ment (T)	CNPE	CSPE	TNPE	TSPE
Pre-test	10.58	10.41	6.39	16.24	8.25	14.00
Post-test	17.55	18.13	14.35	21.88	16.60	20.67
Total diff	6.98	7.72	7.96	5.65	8.35	6.67
p-value	0.000	0.000	0.000	0.010	0.000	0.000

Statistics in the table 7 confirm that learning occurred in both groups, and there was a statistically very significant difference between pre- and post-test results ($p \leq 0.01$) in all groups.

Based on the data presented, we can not fully reject our null hypothesis, which was that ViLLE does not aid the learning of basic programming concepts. The absolute values and the difference between CSPE and TSPE groups, however, indicate that there is a trend towards treatment group, suggesting that ViLLE might have a positive effect on students' learning.

Novices vs. Experienced

The other research question was, whether the effect of ViLLE is the same for novices and experienced students. The null hypothesis was that there is no difference between novices and experienced students. The treatment and control groups were both divided into two groups based on the previous programming experience. In contrast to the first research question, the students' results were compared *inside* the group, rather than between the groups.

Previous knowledge

The results from the pre-test are compared between novices (NPE) and experienced (SPE) in the control group (Table 8) and the treatment group (Table 9). Tables include averages, standard deviations (in parenthesis) and p-values obtained from two-tailed t-test for each question separately and for total number of points acquired in the pre-test.

Table 8: Pre-test scores for CNPE and CSPE

Question	Control and NPE (n = 20)	Control and SPE (n = 12)	p-value
Q1	4.17 (2.33)	6.59 (2.53)	0.003
Q2	1.22 (1.78)	4.71 (4.31)	0.005*
Q3	1.00 (2.86)	4.94 (4.62)	0.005
Total	6.39 (4.68)	16.24 (9.63)	0.001*

Table 9: Pre-test scores for TNPE and TSPE

Question	Treatment and NPE (n = 20)	Treatment and SPE (n = 12)	p-value
Q1	5.60 (2.11)	7.17 (2.76)	0.107*
Q2	1.00 (2.22)	4.00 (4.51)	0.049*
Q3	1.65 (3.62)	2.83 (4.34)	0.414
Total	8.25 (5.44)	14.00 (8.48)	0.051*

We can see that there is a statistically very significant difference between CNPE and CSPE. The difference between TNPE and TSPE is also statistically significant ($t(30) = -2.11$, $p = 0.051$). Thus, we conclude that there is statistically significant difference between NPE and SPE in both groups.

Post-test results

Table 10 presents statistics between CNPE and CSPE and Table 11 between TNPE and TSPE. In the first column the correspondent pre-test question label is shown in parenthesis. The tables include averages, standard deviations (in parenthesis), and p-values obtained from two-tailed t-test for each question. In addition, the total points of shared questions (pre- and post-test), total points (post-test), differences between each question in the pre- and post-test, and the total difference in shared questions are displayed.

Table 10: Post-test scores for CNPE and CSPE

Question	CNPE (n = 23)	CSPE (n = 17)	p-value
PQ1 (Q1)	5.74 (2.78)	7.06 (2.75)	0.144
PQ2 (Q2)	3.39 (3.97)	7.41 (3.81)	0.003
PQ3	5.30 (4.06)	7.59 (2.90)	0.045*
PQ4 (Q3)	5.22 (4.83)	7.41 (3.94)	0.122*
PQ5	6.09 (4.09)	8.35 (2.96)	0.049*
Total (shared)	14.35 (8.27)	21.88 (8.51)	0.008
Total (all)	25.74 (14.44)	37.82 (13.68)	0.011
Diff PQ1	1.57 (2.48)	0.47 (2.70)	0.198*
Diff PQ2	2.17 (3.07)	2.71 (3.65)	0.620
Diff PQ4	4.22 (4.73)	2.47 (4.87)	0.261
Total diff	7.96 (5.80)	5.65 (7.98)	0.295

Table 10 shows that there is a statistically very significant difference between CNPE and CSPE in the post-test scores. The same phenomenon was observed also in the pre-test. As shown in Table 7, learning occurred both in CNPE ($p < 0.01$) and CSPE ($p < 0.05$). Yet, a very significant difference remains between CNPE and CSPE in shared questions ($p = 0.008$), and there also is a very significant difference ($p = 0.011$) in the total points in the post-test.

Table 11: Post-test scores for TNPE and TSPE

Question	TNPE (n = 20)	TSPE (n = 12)	p-value
PQ1 (Q1)	5.90 (2.86)	6.50 (2.43)	0.533*
PQ2 (Q2)	4.70 (4.58)	6.83 (4.22)	0.199
PQ3	5.05 (3.65)	7.25 (3.65)	0.109
PQ4 (Q3)	6.00 (4.71)	7.33 (3.94)	0.418
PQ5	6.05 (4.20)	7.75 (3.82)	0.261
Total (shared)	16.60 (9.29)	20.67 (7.64)	0.212
Total (all)	27.70 (15.49)	35.67 (13.53)	0.151
Diff PQ1	0.30 (2.62)	-0.67 (3.11)	0.354
Diff PQ2	3.70 (4.38)	2.83 (3.46)	0.564
Diff PQ4	4.35 (4.73)	4.50 (4.38)	0.929
Total diff	8.35 (7.98)	6.67 (4.14)	0.439*

As seen in Table 9, the difference between TNPE and TSPE in the pre-test was statistically significant ($p = 0.051$). In the post-test, however, there was no statistically significant difference in any of the questions, in total points, or in differences in the shared questions (see Table 11). Therefore, the null hypothesis can be rejected, and we can conclude that ViLLE is more beneficial for the novice students than for the experienced ones.

Discussion

The evaluation of our research results was studied in two separate cases. In the first case we compared the learning results of the treatment and the control group. The control group used only a textual programming tutorial, while the treatment group using the same material could in addition execute the examples with ViLLE.

In the first research question we compared learning results between control and treatment groups. We found no statistically significant difference between the groups, and thus we can not reject the null hypothesis. Similarly we found no difference in results between genders, between students' that had used ViLLE before the research, or students' with no earlier experience with ViLLE.

In absolute scale, the results favoured the treatment group, indicating that ViLLE might have a positive effect on students' learning. However, the differences were too small in order to get statistically significant results. One reason for that might be that the treatment group's students were coping with a heavier cognitive load (see Chandler & Sweller, 1996) due to the fact that they used ViLLE in addition to the textual material. This load was even heavier for those who hadn't used ViLLE beforehand. We believe that the cognitive load combined with the short learning session was the primary reason for not achieving statistically significant results between treatment and control groups. Another reason might be the low count of participants (n=32 in treatment group; n=40 in control group) as well as the short duration of the experiment. However, there was a significant (at 0.01 level) medium correlation (0.452) between the post-test and the final exam scores. Hence, the results of the two hour session seem to somewhat predict the outcome of the whole course.

In the second research question, we compared the students' learning performance inside both groups when previous programming experience was taken into account. The treatment group was divided into two groups: one with no previous programming experience (TNPE) and the other with some previous programming experience (TSPE). In the pre-test, the difference between the groups was statistically significant ($p = 0.051$). In the post-test, on the other hand, there was no statistically significant difference at all. So, there is solid evidence that ViLLE is more beneficial to novices, and thus we can reject the null hypothesis. The control group was divided identically to CNPE and CSPE. There was a statistically very significant difference between these groups both in the pre- and post-test, which was opposite to TNPE vs. TSPE. Hence, it seems that ViLLE has a substantial effect on narrowing the gap between novices and more experienced students. The learners' short exposure to the tool makes the result even more remarkable.

With these findings combined, it seems that ViLLE enhances students' learning of basic programming concepts. ViLLE proved to be particularly beneficial for novice students, effectively evening out the differences caused by previous programming experience.

Conclusions

We conducted an experiment focusing on program visualization's effectiveness on learning basic programming concepts. We utilized the ViLLE tool in the first programming course at the University of Turku. We found evidence that program visualization, more specifically the ViLLE tool, enhances students' learning regardless of previous programming experience. Moreover, it seems that the tool benefits novice learners more than learners with some previous experience. The differences between the novices and more experienced learners disappeared in the treatment group during a very short training period. In the future, we plan to carry out a study in which ViLLE is used throughout the course and evaluate its individual features separately.

Acknowledgments

This work was supported by the Academy of Finland under grant number 111396.

References

Ben-Ari, M. (2001). Program visualization in theory and practice. *Informatik/Informatique, 2,* 8-11.

Boyle, T., Bradley, C., Chalk, P., Jones, R. & Pickard, P. (2003). Using blended learning to improve student success rates in learning to program. *Journal of Educational Media, special edition on Blended Learning, 28*(2-3), 165-178.

Brown, M. H. (1988). Exploring algorithms using Balsa II. *IEEE Computer, 21*(5), 14-36.

Brown, M. H. (1991). Zeus: A system for algorithm animation and multi-view editing. *Proceedings of IEEE Workshop on Visual Languages,* 4-9. New York: IEEE Computer Society Press.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2005). RAPTOR: A visual programming environment for teaching algorithmic problem solving. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, Missouri, USA, 176-180.

Chandler, P., & Sweller, J. (1996). Cognitive load while learning to use a computer program. *Applied Cognitive Psychol, 10*, 151-170.

Grissom, S., McNally, M., & Naps, T. (2003). Algorithm visualization in CS education: Comparing levels of student engagement. *Proceedings of the ACM Symposium on Software Visualization*, San Diego, California, 87–94.

Hundhausen, C. D., & Brown, J. L. (2007). 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., Douglas, S. A. & Stasko, J. D. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing, 13*, 259-290.

Hyrskykari, A. (1993). Development of program visualization systems. Report, Department of Computer Science, University of Tampere, Finland. Presented at the *2nd Czech British Symposium of Visual Aspects of Man-M6achine Systems*, Praha, 1-21.

Kannusmäki, O., Moreno, A., Myller, N., & Sutinen, E. (2004). What a novice wants: Students using program visualization in distance programming course. *Proceedings of the Third Program Visualization Workshop (PVW'04)*, Warwick, UK.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13*(4).

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

Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A., & Malmi, L. (2005). 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. (2005). The feasibility of automatic assessment and feedback. *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*, Porto, Portugal, 113-122.

Lister, R., Adams, S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., et al. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin, 36*(4), 119-150.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., & Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education, 3*(2), 267-288.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y., et al. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*(4), 125-140.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. *Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 373-376.

Naps, T., Cooper, S., Koldehofe, B., Leska, C., Rößling, G., Dann, W., et al. (2003). Evaluating the educational impact of visualization. *Working group reports from ITiCSE on Innovation and Technology in Computer Science Education*, ACM Press, 124-136.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., et al. (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, 35*, 2, 131-152.

Nikula, U., Sajaniemi, J., Tedre, M. & Wray, S. (2007). Python and Roles of Variables in Introductory Programming: Experiences from Three Educational Institutions. *The Journal of Information Technology Education, 6*, 199-214. Retrieved from http://jite.org/documents/Vol6/JITEv6p199-214Nikula269.pdf

Oechsle, R. & Schmitt, T. (2002). JAVAVIS: Automatic program visualization with object and sequence diagrams using the java debug interface (JDI). *Lecture Notes in Computer Science, Vol. 2269: Software Visualization,* 176-190.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2007). VILLE – A language-independent program visualization tool. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, Vol. 88*, Australian Computer Society. Raymond Lister and Simon, Eds.

Sajaniemi, J. (2002). PlanAni – A system for visualizing roles of variables to novice programmers. *University of Joensuu, Department of Computer Science, Technical Report, Series A, Report A-2002-4*.

Sajaniemi, J. & Kuittinen, M. (2003). Program animation based on the roles of variables. *Proceedings of the 2003 ACM Symposium on Software Visualization*, San Diego, California, 7-ff.

Stasko, J. (1992). Animating algorithms with XTANGO. *ACM SIGACT News, 23*(2), 67-71.

Tenenberg, J., Fincher, S., Blaha, K., Bouvier, D., Chen, T.-Y., Chinn, D., et al. (2005). Students designing software: A multi-national, multi-institutional study. *Informatics in Education, 4*(1), 143-162.

Biographies

Teemu Rajala is a PhD student at University of Turku. He received his master's degree from the same university in 2007. His research focuses on visualization of programs and algorithmic problem solving.

Mikko-Jussi Laakso is a PhD student working as a researcher in a joint project of University of Turku and Helsinki University of Technology. He received his M.Sc (Computer Science) in 2003. His research interest covers program and algorithm visualization, learning enviroments, computer aided and automatic assessment in computer science education.

Erkki Kaila has written his Master's thesis on program visualization in programming learning in University of Turku. His research interests include program visualization systems and IT education.

Tapio Salakoski is a professor of Computer Science at University of Turku, where he received his Ph.D. in 1997. His main research focus has been in methodology development using machine learning and other intelligent techniques. He is leading a multidisciplinary research group studying various task domains, including problems related to human learning and computing education research.

Paper 6

Laakso, M.-J., Rajala, T., Kaila, E. and Salakoski, T. (2008). The Impact of Prior Experience in Using a Visualization Tool on Learning to Program. In proceedings of CELDA 2008, Freiburg, Germany, 129-136.

THE IMPACT OF PRIOR EXPERIENCE IN USING A VISUALIZATION TOOL ON LEARNING TO PROGRAM

Mikko-Jussi Laakso, Teemu Rajala, Erkki Kaila and Tapio Salakoski
University of Turku, Turku Centre for Computer Science (TUCS)
Turku, Finland

ABSTRACT

Programming is typically hard for novices. Program visualization is one method suggested for aiding the learning process. However, it is important that the tools used in visualizing programs are used correctly. To reduce the users' cognitive load of learning to use the tool, they should be properly familiarized with it beforehand. We conducted a research on the effects of cognitive load in using a program visualization tool called ViLLE. The treatment group was familiar with the tool, while the control group used the tool for the first time. The results indicate that the students with previous experience with the tool learned significantly better. Therefore we conclude that to get the most benefit of a visualization tool, the students should be advised to use it effectively.

KEYWORDS

Program visualization, cognitive load, learning to program.

1. INTRODUCTION

Writing a computer program is a cognitively demanding task. Learners need to adapt new ways of thinking when learning to program, but as Eckerdal et al. (2005) report, students have hard time describing what is meant by 'programming thinking'. According to multi-national studies published in recent years students also have problems in writing (McCracken et al., 2001) and reading (Lister et al., 2004) programs as well as in designing software (Tenenberg et al., 2005). Clearly something should be done to improve the outcome of programming studies.

Visualization has been used as a method in assisting learners in understanding the behavior of programs and algorithms. Over the years many visualization tools that use graphical components in clarifying the execution of programs and changes in program states have been developed. However, the results on their effectiveness in learning programming have been mixed (Hundhausen et al., 2002). One reason for not getting positive effects on using a visualization tool might be the effects of additional cognitive load required in learning to use the tool correctly.

The purpose of visualizations is to facilitate the learning task, in other words to transfer some of the cognitive load to learner's perceptual system (Robertson et al., 1991). However, for novices it is often hard to determine which parts of the visualization are important and relevant, and in what states of the visualization (Ben-Ari, 2001). The relevant parts should of course be highlighted by the visualization tool itself, but that still doesn't ensure that learner knows how to interpret those visualizations. As Petre (1995) has noted: "The question is not 'Is a picture worth a thousand words?', but 'Does a given picture convey the same thousand words to all viewers?'" Thus, when using a visualization tool, one should clearly teach how to use the tool and what is the purpose of its different visualization components. When studying the effectiveness of a visualization tool, and especially when the study is conducted in a single short training session, one should ensure that the students participating in the study know how the visualizations work. A short tutorial session on how to use the tool correctly before such studies should even out the cognitive load of control and treatment groups at the beginning of the study.

ViLLE is a program visualization tool developed at the University of Turku and its main focus is on teaching programming basics to novice programmers (Rajala et al. 2007). ViLLE has an extendable support for multiple programming languages, which enables simultaneous visualization of programs with two

different languages. The visualization itself consists of highlighting the progress of execution, showing the states of variables during the execution both graphically and textually, displaying the call stack of methods and showing automatically generated textual descriptions of each event in the program. Learners can be engaged with interactive pop-up questions, which can be created with the tool's built-in editor.

In this paper we present a research conducted in two consecutive high school programming courses in fall 2007 and spring 2008. The purpose of this research was to find out what kind of effects cognitive load has on learning programming with a program visualization tool. The students in both courses took part in a two hour computer lab session, where they practiced basic programming concepts with the ViLLE tool. The difference between the courses was that the spring 2008 course was familiar with the tool's interface and it's usage before the session. Thus, we tried to find evidence that by minimizing the additional cognitive load caused by a visualization tool, students can better focus on learning the programming concepts. This evidence would then show that it is essential to familiarize learners with a visualization tool before studying the effectiveness of such tools.

This paper has the following structure. In section 2 we present some previous studies on visualization's effectiveness. The ViLLE tool and its features are described shortly in section 3. The setting of our research is presented in section 4, and in section 5 we report its results. Those results are then discussed in section 6, and finally conclusions are presented in section 7.

2. RELATED WORK

Jeliot 3 is a program visualization system used in tracing the execution of Java programs. Ben-Bassat Levy et al. (2002) conducted a research on its effectiveness and found out that animations improved the learning of students with difficulties understanding the abstract models. JIVE (Gestwicki & Jayaraman 2002) is a program visualization tool which visualizes object structure and method calls. According to Gestwicki and Jayaraman it has proven itself as a practical tool for visualization and debugging. Other program visualization tools include e.g. BlueJ (Kölling et al. 2003), JavaVis (Oeschle & Schmitt 2002) and ALVIS LIVE! (Hundhausen & Brown 2007). Notable algorithm visualization systems include e.g. JHAVE (Grissom et al. 2003), BALSA-II (Brown 1988) and TRAKLA2 (Malmi et al. 2004).

Previous experiments on effectiveness of visualization include e.g. Stasko et al. (1993), Crosby and Stelovsky (1995), Byrne et al. (1999), Kann et al. (1997) and Hansen et al. (2000). These all had somewhat similar setup to our experiment, including the pre- and post-tests. However, although there might be mentions about the previous usage of the system researched (see e.g. Ebel and Ben-Ari, 2006) the effects of cognitive load on learning the system are usually not taken into account. Naps et al. (2003) suggest that the easiness of system's usage should be one instrument for evaluating a visualization system. The effects to the learning outcome are however rarely measured. More about cognitive load in learning to use a computer program can be found for example in Chandler & Sweller (1996) and cognitive load in general in Mayer (2001).

3. VILLE

ViLLE is a program visualization tool for teaching programming to novice programmers. It can visualize programs in various programming languages, and by defining new syntaxes with the built-in editor, the user can easily extend the language support. ViLLE also includes a predefined set of programming examples, which cover various programming concepts. The user can add new examples to the tool or modify the existing ones. The examples can be exported from the tool and published in the web, so that the students can use them at any time and place. Additionally, ViLLE provides tools for creating multiple-choice and array related pop-up questions.

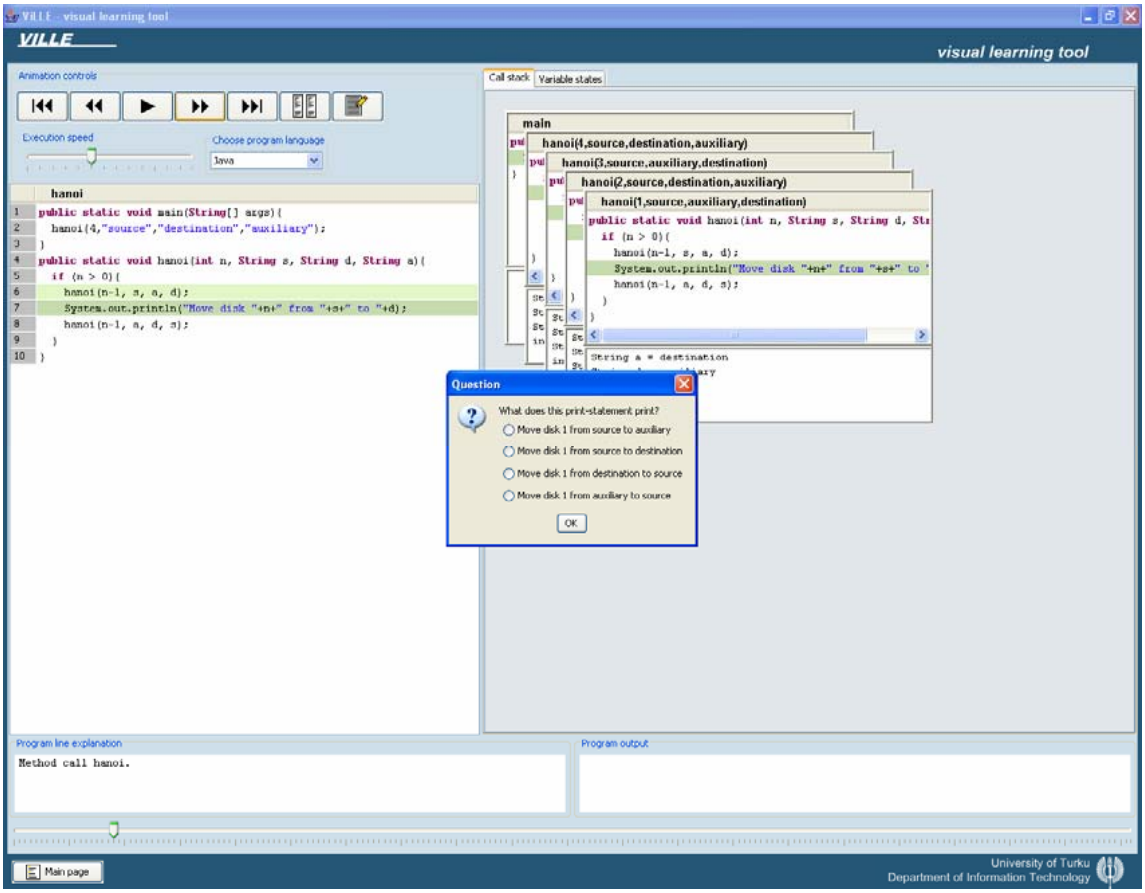


Figure 1. The visualization view of ViLLE in call stack mode.

The visualization view of ViLLE (Figure 1) consists of three areas. The left side of the view contains the program code, buttons for controlling the visualization and a drop-down menu for choosing the programming language in use. The right side of the view displays the call stack, which shows the order of method calls, the local variables, and the progress of the execution in methods. The call stack can be replaced with a variable visualization area, which presents arrays and matrices graphically. Additionally, the user can choose to view the execution in a parallel mode, in which the right side of the view displays the program code similarly to the left side, thus enabling the user to follow the execution in two different languages simultaneously. The fields at the bottom of the view show an automatically generated explanation of the current program line and the output of the program. The slider below them visualizes the progress of the execution and allows the user to easily move to any state of the execution.

The main idea of ViLLE is to provide a language-independent and thus a more abstract view on programming. The built-in support for multiple languages enables simultaneous viewing of the program visualization in two different languages in parallel, which should help students in understanding the similarity between various programming concepts in imperative programming languages. The purpose of this is to demonstrate the importance of understanding how the programming concepts actually work in contrast to just learning some specific issues related to the syntax of programming languages. For more detailed information on ViLLE, see Rajala et al. (2007).

We have previously studied the effectiveness of ViLLE (Rajala et al. 2008). The results showed that ViLLE significantly enhances the learning of novice students in a single learning session. In the other study (Kaila et al. 2008) we studied the effects of ViLLE in different levels of the engagement taxonomy (see Naps et al 2002). The results showed that ViLLE is most useful for students with no previous experience that used it in higher levels of engagement. This paper however focuses on the differences in effects of ViLLE for students with or without previous experience of the system.

4. RESEARCH

We conducted a research on ViLLE's effectiveness in learning programming basics at the High School of Kupittaa, Turku, Finland in fall 2007 and in spring 2008. The research was carried out in two consecutive instances of the first programming course. In the third week of both courses, a two hour computer lab session was organized where students rehearsed basic programming concepts with ViLLE. The session included a pre- and post-test, which were used to measure students' learning. The difference between the two student groups was that the spring 2008 course was familiarized with the tool's user interface and usage before the two hour session. With this setting we tried to find evidence that by minimizing the additional cognitive load caused by the visualization tool, students should be able to focus more on learning the programming concepts instead of the tool's user interface and graphical notation.

4.1 Method

We carried out a research which was between-subject design with pre-test and post-test (dependent-variable). We had one between-subject factor (independent variable): the previous usage of ViLLE. Our research question was: "Does the previous usage of the tool affect the learning outcome?" The null hypothesis was that the previous usage of the tool has no effect on learning.

4.2 Materials

To determine the level of programming knowledge before using ViLLE, a pre-test was used. The test consisted of three questions. In each question the students were asked to determine the output of a given program code. In the first question the program code contained three consecutive conditional statements. The students were supposed to track down the executed blocks and their effects on the values of variables. In the second question the students were presented a loop in which the values of two variables were changed and printed. The third question included a loop in which a function was called. The function returned the given parameter affected differently based on its original value. The time reserved for answering to the pre-test was 15 minutes.

After completing the pre-test the students used a previously prepared programming tutorial, which was presented as a web page. Tutorial consisted of basic programming concepts, including variable usage and manipulation, conditional statements, loops, and function definitions and calls. Each topic was covered with a brief description and examples. The students could visualize the execution of examples with ViLLE by selecting a link attached to them. Additionally, the students were asked to write down the output of each example to ensure that they really went through the entire tutorial. The time reserved for using the tutorial was 45 minutes.

After the tutorial a post-test was arranged to measure the actual learning outcome. In addition to the pre-test questions, there was an additional question, in which the students were asked to fill in the blanks in a given program code; the resulted program was supposed to output all even numbers between 2 and 24. The time reserved for answering the post-test was 30 minutes.

4.3 Participants

The participants were students from the high school of Kupittaa, which focuses on teaching information technology and media, and offer a discrete program for students interested in those fields. In fall 2007, the course included 17 students (the control group). The students were divided into two identical sessions; 11 participants in the first session and 6 participants in the second session. In spring 2008 the student count was 7 (the treatment group). The course was the first programming course in the curriculum for every student.

The difference between the groups was that the control group had no previous information of the visualization tool other than a brief glimpse of the user interface shown by the course instructor before the practice session, whereas the treatment group had gone through a tutorial of the tool.

4.4 Procedure

The study was performed during the first programming course. The sessions were arranged at the beginning of each course. Each session started with the students taking the pre-test independently. The time reserved for answering was 15 minutes.

After the pre-test the students used the programming tutorial for 45 minutes. The students were advised to use ViLLE to visualize the execution of the programming examples. After the instructions the students worked independently during the whole session: they were allowed to ask for assistance only if they encountered any technical difficulties.

After the programming tutorial session the students answered to the post-test in 30 minutes. The extra time was arranged because the additional question in the post-test was considered to be more demanding than the questions in the pre-test.

Each question in the pre- and post-tests were analyzed in the scale of 0 to 10. Zero points meant that the answer was totally wrong, and each point advanced meant an increase of 10 percent in the correctness. The total maximums were 30 points for the pre-test, and 40 points for the post-test.

5. RESULTS

In this section we present the result of our research related to the independent variable of previous usage of the ViLLE tool. The results were analyzed with a two-tailed and pair-wise t-test, and Kolmogorov-Smirnov -test was used to check the distributions of the gathered data. In addition, Levene's test was used to calculate variances for all statistics to determine if the data holds equal or non-equal variances.

5.1 Pre-test

The pre-test results are presented in the table 1. The means and standard deviations for each question are shown.

Table 1. Averages and standard deviations (in parenthesis) of the pre-test results for the treatment and control group

Group	*Question 1*	*Question 2*	*Question 3*	***Total***
Control (N=17)	5.06 (2.49)	1.41 (2.32)	0.65 (0.79)	7.12 (4.29)
Treatment (N=7)	5.57 (4.32)	1.86 (3.67)	2.00 (3.56)	9.43 (8.50)

As the table shows, the treatment group outperformed the control group in each individual question and in total points. There was no statistical significant difference between the treatment and control group in any single question or in total points. To confirm the similarity of the groups, we also looked at the participants' math and introductory CS course grades. Table 2 presents the means and standard deviations of the grades. We couldn't include the math grades of five students in the control group, since they didn't take the course.

Table 2. Math and introductory CS course grades (scale from 4 to 10).

Group	*Math grade*	***CS grade***
Control	6.75 (1.60)	7.94 (1.09)
Treatment	7.67 (2.25)	8.57 (1.62)

There was no statistical difference between groups in either grade. In absolute scale the differences were less than one point in favor of the treatment group.

5.2 Post-test

The post-test results are presented for the treatment and control group in table 2. The first three questions were exactly the same as in pre-test. Since there was an additional question (PQ4) in the post-test, the table includes the total points of the pre-test and the total points of the shared questions (PQ1, PQ2 and PQ3).

Table 3. Averages and standard deviations (in parenthesis) of post-test results for the treatment and control group

Group	*PQ1*	*PQ2*	*PQ3*	*Total shared*	*PQ4*	***Total all***
Control (N=17)	6.53 (2.45)	3.88 (3.74)	2.18 (3.03)	12.59 (5.94)	4.35 (3.55)	16.94 (8.62)
Treatment (N=7)	7.71 (4.07)	6.71 (4.35)	5.14 (3.81)	19.57 (10.33)	6.86 (3.44)	26.43 (12.96)

From table 3 we notice that the treatment group outperformed the control group in all questions and the difference in absolute scale is greater than in pre-test. However, both groups improved their results in all shared questions in the post-test. Pair wise t-test between *the pre-test total* and *post-test total shared* inside the groups confirmed that learning took place during the setup in both groups, and the differences were statistically very significant (Control group: $t(16) = -4.52$, $p < 0.01$, Treatment group: $t(6) = -3.85$, $p < 0,01$). Additionally, we can see that the difference between groups was greatest in the more demanding questions (PQ2, PQ3 and PQ4).

Table 3 presents the statistical differences between the groups in pre-test total points and post-test total points in shared questions and in all questions.

Table 4. Pre- and post-test total scores.

	Pre-test total (avg)	*Post-test total shared (avg)*	***Post-test total all (avg)***
Control (N=17)	7.12	12.59	16.94
Treatment (N=7)	9.43	19.57	26.43
t-test value (two-tailed)	*0.515*	*0.047*	*0.046*

There was no statistically significant difference between the groups in pre-test. However, the treatment group performed statistically significantly better in the post-test both in the total shared ($t(22) = -2.10$, $p < 0.05$) and in total all ($t(22) = -2.14$, $p < 0.05$). The reliability of the post-test questions was high ($\alpha = 0,748$). Thus, we can reject our null hypothesis which was that the previous usage of the tool has no effect on learning, and conclude that the previous usage of the tool significantly enhances the learning performance of students.

6. DISCUSSION

As the results show, the treatment group outperformed the control group in the post-test. This supports our hypothesis about the effect of cognitive load in using a visualization system. The distinctive factor between the groups was that the treatment group was familiar with the tool and the different visualization views before the session. Hence, it seems that instead of learning to use the tool, the treatment group could focus solely on learning the subjects presented. Based on the statistical analysis of the results, the answer to our research question is that the previous usage has a positive effect on the learning outcome, and thus the null hypothesis can be rejected. This supports the argument by Ben-Ari (2001) that the novice users don't necessarily understand the purpose of different visualizations.

We found evidence that in general the math grade correlated with the test scores: those with higher math grade performed better in pre- and post-test. This supports the findings of Moskal et al. (2004) and Butcher & Muth (1985). The CS course's grades correlate only with the post-test scores; because of this, and since the course was an introductory course, and not related to programming, we assume that the CS grades reflect the students' motivation.

There are some additional factors that might affect the differences in the learning results between the groups. Firstly, the group sizes were quite small. Secondly, the difference in math grades between the groups – although not statistically significant – can have an effect on the results. However, substantial learning occurred in both groups. This supports our previous findings (see Rajala et al., 2008) and confirms that ViLLE can be used effectively to teach basic programming concepts.

7. CONCLUSIONS

We conducted a research on the effects of prior usage of the tool when using it to learn basic programming concepts. The results indicate that the students who are familiar with the tool beforehand seem to benefit more from it than students using it the first time. Because the effect seems to be so significant, the prior experience should be taken into account when studying the effectiveness of such tools; to our knowledge studies about visualization's effectiveness seem to ignore this or not report it. In general, teachers should introduce the tool properly to students before the actual usage to ensure that the students can focus on learning. This is especially important when conducting studies where the duration of the experiment is short (see also Laakso et al. 2008).

REFERENCES

Ben-Ari, M. (2001). Program visualization in theory and practice. *Informatik/Informatique, 2*, 8-11.

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

Brown, M.H. (1988). Exploring Algorithms Using Balsa II. *IEEE Computer, 21*(5), 14-36.

Butcher, D.F. & Muth, W.A. (1985). Predicting performance in an introductory computer science course. *Communications of the ACM, 28*, 3, 263-268.

Byrne, M.D., Catrambone, R. & Stasko, J.T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & Education, 33*, 253-278.

Chandler, P. & Sweller, J. (1996). Cognitive load while learning to use a computer program. *Applied Cognitive Psychology, 10*, 151-170.

Crosby, M. E. & Stelovsky, J. (1995). From multimedia instruction to multimedia evaluation. *Journal of Educational Multimedia and Hypermedia, 4*, 147-162.

Ebel, G. & Ben-Ari, M. (2006). Affective effects of program visualization. *Proceedings of the 2006 international workshop on Computing education research, Canterbury, UK*.

Eckerdal, A., Thuné, M. & Berglund, A. (2005). What does it take to learn 'programming thinking'? *Proceedings of the 2005 international workshop on Computing education research*, Seattle, WA, USA, 135-142.

Gestwicki, P. & Jayaraman, B. (2002). Interactive visualization of Java programs. *Proceedings of Symposia on Human Centric Computing Languages and Environments*, 226-235.

Grissom, S., McNally, M. & Naps, T. (2003). Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. *Proceedings of the ACM Symposium on Software Visualization, San Diego, California*, 87-94.

Hansen, S. R., Narayanan, N. H. & Schrimpsher, D. (2000). Helping learners visualize and comprehend algorithms. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning 1*.

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

Kann, C., Lindeman, R.W. & Heller, R. (1997). Integrating algorithm animation into a learning environment. *Computers & Education, 28*, 223-228

Kölling, M., Quig, B., Patterson, A. & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13*(4).

Laakso, M.-J., Myller, N. & Korhonen, A. (2008). Comparing Learning Performance of Students Using Algorithm Visualizations Collaboratively on Different Engagement Levels. To appear in the *Journal of Educational Technology & Society.*

Lister, R., Adams, S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin, 36*(4), 119-150.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O. & Silvasti, P. (2004). Visual Algorithm Simulation Exercise System with Automatic Assessment: TRAKLA2. *Informatics in Education, 3*(2), 267-288

Mayer, R. E. (2001). Multimedia learning. Cambridge University Press, New York.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*(4), 125-140.

Moskal, B., Lurie, D. & Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach, Proceedings *of the 35th SIGCSE technical symposium on Computer science education*, Norfolk, Virginia, USA.

Naps, T., Cooper, S., Koldehofe, B., Leska, C., Rößling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R.J., Anderson, J., Fleischer, R., Kuittinen, M. & McNally, M. (2003). Evaluating the educational impact of visualization. *Working group reports from ITiCSE on Innovation and Technology in Computer Science Education*, ACM Press, 124-136.

Oechsle, R. & Schmitt, T. (2002). JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI). *In Diehl, S. (Ed.), Software Visualization. vol.2269 of Lecture Notes in Computer Science*, Springer-Verlag, 176-190.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2007). VILLE – A language-independent program visualization tool. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, Vol. 88*, Australian Computer Society. Raymond Lister and Simon, Eds.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2008). Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. *To appear in the Journal of Information Technology Education: Innovations in Practice.*

Robertson, G.G., Mackinlay, J.D. & Card, S.K. (1991). Cone Trees: animated 3D visualizations of hierarchical information, *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology,* New Orleans, Louisiana, United States, 189-194.

Stasko, J., Badre, A. & Lewis, C. (1993). Do algorithm animations assist learning? An empirical study and analysis. *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems*, ACM Press, New York, 61-66.

Tenenberg, J., Fincher, S., Blaha, K., Bouvier, D., Chen, T.-Y., Chinn, D., Cooper, S., Eckerdal, A., John-son, H., McCartney, R. & Monge, A. (2005). Students designing software: a multi-national, multi-institutional study. *Informatics in Education, 4*(1), 143-162.

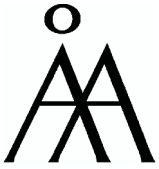
Turku Centre for Computer Science
TUCS Dissertations

98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming
128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics



Åbo Akademi University
- Department of Information Technologies

Turku School of Economics
- Institute of Information Systems Sciences

ISBN 978-952-12-2486-7
ISSN 1239-1883

Mikko-Jussi Laakso

Promoting Programming Learning: Engagement, Automatic Assessment With Immediate Feedback in Visualizations

Mikko-Jussi Laakso

Promoting Programming Learning