Effects of Collaboration in Program Visualization

Teemu Rajala, Erkki Kaila, Mikko-Jussi Laakso, Tapio Salakoski, University of Turku and Turku Centre for Computer Science (TUCS), 20014 Turun yliopisto, Finland.
Email: temira@utu.fi, ertaka@utu.fi, milaak@utu.fi, sala@utu.fi

Abstract: Program visualization is a method shown to be beneficial in supporting programming learning. In this paper we study the effects of student collaboration on learning basic programming concepts with a program visualization tool called ViLLE. Students participated in a two hour computer lab session, during which they first answered to a pre-test, then rehearsed programming with ViLLE, and finally answered to a post-test. Students were divided into two groups: in the other group students studied alone, and in the other in pairs. The results show that all students learned significantly during the session, and that the students working in pairs learned significantly better than students working alone. Thus, it seems that the learning effects of program visualization can be enhanced with collaboration and the effect is greater in tougher questions related to functions and parameter passing.

Introduction

Learning to program is typically hard. Several studies indicate that the outcome of programming courses is substandard (see next section). Utilizing visualization in teaching programming has been tried as a mean of supporting the learning process, and it has been found out that it is beneficial only when used actively (Hundhausen, 2002).

The idea of studying the effects of collaboration seems sensible, because visualization provides something concrete for the students to discuss. Students working in pairs with a program visualization tool have a common reference point (the current state of the visualization) which is enhanced even more with the help of pop-up questions. Additionally, as collaborative methods are used increasingly today, it should be interesting to see what kind of effect collaboration has in program visualization.

Similarly to pair-programming (see Williams & Kessler, 2000) the student with more experience in programming can help their partner in understanding the problem. There is actually inherent motivation for stronger students to help the weaker students because the pair shares a common goal (finishing the task at hand). Also, students with different skill levels spot different things in programs, so even the weaker student can contribute to the common task.

ViLLE is a program visualization tool developed at the University of Turku, Finland. The studies on the effects of ViLLE have shown that it is beneficial for novice students in learning the basics of programming. In this paper we report the effects of student collaboration when utilizing a program visualization tool.

To study the effects of collaboration, we conducted a session where half of the students used ViLLE alone and the other half in collaboration with another student. Students were encouraged to discuss the visualizations during the session.

The paper has the following structure: the subsection in the introduction briefly describes the program visualization tool used in our experiment, in the next section some previous studies related to the subject of the paper are presented, the section after that describes the research setup used in our experiment, and the results of the study are presented in the following section. Finally, in the last two sections the results are discussed, and conclusions presented.

ViLLE

ViLLE is a program visualization tool developed at the University of Turku, Finland. Its purpose is to support the learning process of novice programmers. ViLLE can be used to visualize the execution of programs in several different programming languages, and the language support is easily extendable (at least for imperative

programming languages). The visualization includes code line highlighting, visualization of variable states, call stack for the method calls, automatically generated textual explanation of program events, and comparison of program execution with two different languages simultaneously in a parallel view. Controlling the progress of visualization and execution is flexible: it is possible to move one step at a time both forwards and backwards in the program, play the execution automatically and adjust the playback speed, and use a slider to move at any state of the execution. It is also possible to add breakpoints to code lines, and jump between the points with the animation controls. New programming languages can be added to the tool with a built-in syntax editor. Multiple-choice questions and array value assignment exercises can be created with a built-in question editor.

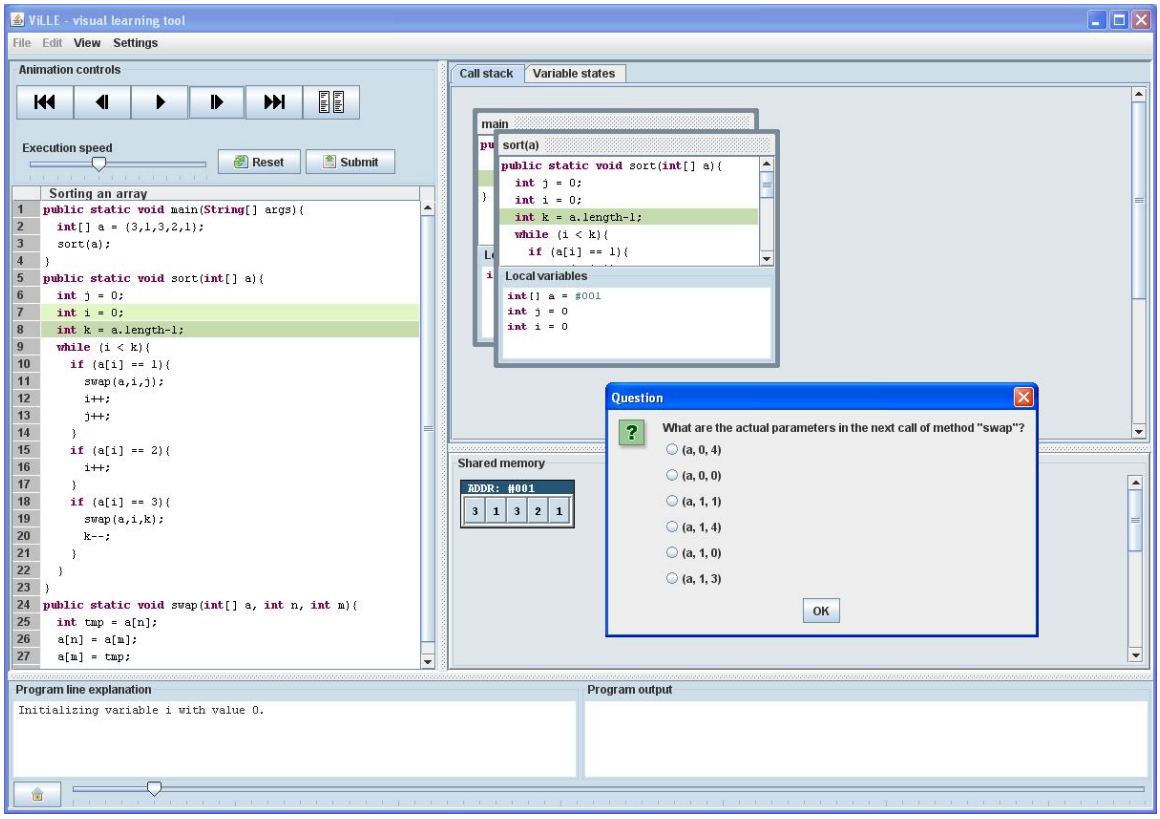

Figure 1. The visualization view in ViLLE

In addition to multiple choice questions and array value assignment exercises, two new exercises types have been developed recently. In code sorting exercise program code is shown to student with code lines shuffled. Student has to arrange the code lines to match a given specification. In coding exercise student has to code a program or part of a program, which is then assessed automatically by the tool. The new exercise types were not used in this study.

ViLLE exercises can be made available on the web with the help of TRAKLA2 (Malmi et al., 2004) web environment. TRAKLA2 handles student logins and keeps track of student submissions and points achieved. The exercises can be organized as rounds and each round can be given opening and closing dates. Teacher can use the environment to easily monitor students' progress. For more detailed information on ViLLE see Rajala et al. (2007) and Kaila et al. (2008).

Related Research

The first steps in learning to program are generally considered to be quite troublesome. According to a multinational study by Lister et al. (2004), the students lack the prerequisite skills for problem solving related to programming. A study by Lahtinen et al. (2005) found that one of the areas the students had most problems with

was in understanding the structure of programs. According to McCracken et al. (2001) most of the students don't know how to program at all after the introductory course. Moreover, Bergin and Reilly (2005) state that only one fifth of the students in introductory courses are actually interested in learning to program.

One of the goals of program visualization (PV) is to make the first steps easier for the novices. PV systems typically illustrate the execution of program code with various graphical and textual objects, with the idea to enhance the program code reading skills of the students. Though there have been various PV systems developed (see e.g. BlueJ (Kölling et al., 2003) or JavaVis (Oechsle & Schmitt, 2001)) and some qualitative research conducted (e.g. Kannusmäki et al. (2004)), there seems to be little or no quantitative research on the subject, especially when program visualization is used during collaborative learning.

Laakso et al. (2009) studied learning performance when algorithm visualization tool TRAKLA2 was used collaboratively, and found out that the students working on the engagement level (Naps et al., 2002) of changing performed statistically significantly better than the students in the viewing level. Myller et al. (2008) studied the extended engagement taxonomy and its applicability when using visualizations collaboratively. Hundhausen (2002) also studied the collaborative aspects of algorithm visualization. Pair programming – quite a typical form of collaboration – has been studied (and found to be efficient) by e.g. Williams et al. (2000) and McDowell et al. (2002).

The effectiveness of ViLLE has been previously studied (Rajala et al., 2008) and the results show that it can enhance novices' learning significantly. Furthermore, we found out that the tool should be utilized in higher engagement level in order to promote active learning (Kaila et al, 2009) and that it is important to familiarize students with the tool before studying its effectiveness to reduce the cognitive load required in learning to use the tool (Laakso et al., 2008). Moreover, feedback gathered from students revealed that most of the students thought that ViLLE is beneficial when learning the basic programming skills; some students even preferred it over traditional learning methods (lectures, demonstrations). (Kaila et al. 2009).

Research Setup

We conducted a study on the effects of collaboration when learning to program with a program visualization tool. The purpose of the study was to find out if there is any difference in learning between students working in pairs and students working alone. Students participated in a two hour computer lab session where they used a program visualization tool called ViLLE to learn basic programming concepts.

The participants were students from a first year introductory CS course called 'Introduction to Information Technology'. The course has two main goals: 1) students learn the different components used in computers and how those components work, and 2) they learn the most relevant basic concepts of programming and acquire good skills in reading computer programs. A total of 110 students participated in the study, 62 of them worked in pairs and 50 alone.

The study was conducted in the beginning of the fourth week of the eight week course. Students had completed one round of exercises (13 exercises in total) with the ViLLE tool before the computer lab session. The completed round covered variable initialization, value assignment, and conditional statements.

At the beginning of the study students first answered individually to a pre-test which consisted of four program tracing exercises covering conditional statements, loop structures and methods. Students were also asked how much programming experience they had before the course, the number of different programming languages they had used, had they used ViLLE before the test, and how many lectures they had attended. Also, if they were not CS majors, they were asked how many CS related courses they had planned to study. Students had 15 minutes time to answer to the pre-test.

After the pre-test, students rehearsed programming concepts with a web-based programming tutorial. In the tutorial various basic programming concepts (conditional statements, variable types, loops, and methods) were shortly explained, and for each individual concept students solved few program comprehension exercises with ViLLE (15 exercises in total). Moreover, students' actions were recorded with screen capture software as well as the discussion of students working in pairs. The pair-group was encouraged to discuss during the tutorial that lasted 45 minutes.

After the tutorial students answered to a post-test which included the same four questions as the pre-test plus two additional questions: one where students had to write a loop that prints even numbers between 2 to 24 and one tracing exercise illustrating behavior of methods. So, in total the post-test included five code tracing exercises and one writing exercise. Students had 30 minutes time to answer the exercises in the post-test.

Results

In this section we present the results on the research question "is there any difference in the learning when the visualization tool is used in collaboration with another student". In addition, results are presented with previous programming knowledge taken into account.

Effects of Collaboration

The results of the pre-test are presented in Table 1. The table includes means and standard deviations in parentheses for each question and the mean points of all groups. In addition, the p-values obtained from two-tailed t-test between treatment and control groups are displayed.

Table 1: Pre-test scores

	Control (N=50)	*Treatment (N=62)*	p-value
Q1	7.60 (3.05)	8.19 (2.69)	0.277
Q2	3.96 (3.52)	4.03 (3.47)	0.914
Q3	4.30 (3.99)	5.60 (3.97)	0.089
Q4	2.72 (4.19)	3.32 (4.62)	0.476
Total	18.58 (11.25)	21.15 (8.95)	0.193

As seen in Table 1, treatment group outperformed control group slightly in all individual questions and in total score. However, no statistically significant differences between the groups were found.

Post-test results for both groups are presented in Table 2. Since the post-test included all of pre-test's questions, the corresponding question numbers are presented in parentheses. In addition, the total points of shared questions (questions in pre- and post-test), total points (all six questions) and the total difference in shared questions, are presented.

Table 2: Post-test results

	Control (N=50)	*Treatment (N=62)*	p-value
PQ1 (Q1)	8.22 (2.80)	8.44 (2.26)	0.654
PQ2 (Q2)	6.18 (4.35)	6.52 (3.77)	0.667
PQ3	5.96 (3.84)	6.84 (3.67)	0.220
PQ4 (Q3)	7.72 (2.96)	8.55 (2.43)	0.107
PQ5 (Q4)	5.08 (4.75)	6.84 (4.38)	*0.046*
PQ6	4.78 (4.62)	6.92 (4.00)	*0.011*
Total (shared)	27.20 (10.80)	30.34 (9.34)	0.108
Total (all)	37.94 (17.40)	44.10 (14.78)	***0.045***
Total (relational)	0.51 (0.34)	0.63 (0.30)	***0.042***

As seen in Table 2, treatment group outperformed the control group in all questions, in total points, and in total difference in shared question (9.19 vs. 8.62). The difference was statistically significant in total points ($t(110) = -2.024$, $p < 0.05$). The relational scores were calculated with the formula

(Post-test total – Pre-test total) / (Post-Test Maximum (60 points) – Pre-test total),

which produces a value between 0 and 1 to describe the relation of increase in learning compared to maximum

increase possible. Again, the difference was statistically significant in favor of the treatment group ($t(110) = -2.055$, $p < 0.05$).

In conclusion, we can reject the null hypothesis and state that the collaboration has an effect when learning basic programming concepts with a program visualization tool. In addition, in the PQ6 the difference is almost statistical significant even if we apply Bonferroni correction, and there is this same trend in the PQ5 in absolute scale. Since, the PQ6 and PQ5 were questions about function calls, it seems that collaboration is especially beneficial in harder questions like functions and parameter passing.

Effects of Previous Knowledge

We also wanted to find out whether previous programming experience would affect the learning results. To achieve this, we asked the students to estimate their programming experience on scale of 0 to 4. Based on this, the students were divided into two groups: the ones who estimated their programming skill as zero belong to the group no previous programming experience (NPE), and all the others to the group some previous programming experience (SPE).

The results for pre- and post-tests for control and treatment groups with previous experience taken into account are presented in Table 3. The p-values were calculated inside the control and treatment groups with one-way ANOVA test.

Table 3: Pre- and post-test results with previous programming experience taken into account

	No previous experience (NPE)			*Some previous experience (SPE)*		
	Control (N=30)	Treatment (N=34)	p-value	Control (N=18)	Treatment (N=28)	p-value
Pre-test total	16.53 (10.05)	17.76 (7.60)	0.580	23.11 (11.95)	25.25 (8.84)	0.489
Post-test shared	24.43 (11.10)	27.06 (9.14)	0.303	32.67 (8.24)	34.32 (8.07)	0.504
Post-test total	33.13 (17.24)	38.68 (14.53)	0.168	48.00 (13.07)	50.68 (12.41)	0.488
Difference	7.90 (8.81)	9.30 (6.84)	0.480	9.56 (8.80)	9.07 (6.57)	0.832

As seen in Table 3, the difference in the pre-test between the control and treatment groups SPE remains in the post-test, while in NPE it grows a little. The differences in absolute scale seem to support our previous results (Rajala et al. 2008) which indicate that ViLLE is most useful for students with no previous programming experience. However, no statistically significant differences between the groups were found.

Learning Results inside Groups

Finally, we wanted to find out whether any learning occurred in all groups during the session. The results are displayed in Table 4. Pre- and post-test scores are compared inside groups with pairwise t-test.

Table 4: Pre-test and post-test scores compared inside groups

Group	*N*	*Pre-test*	*Post-test (shared)*	*p-value*
Treatment group	62	21.15 (8.94)	30.34 (9.34)	0.000
Control group	50	18.58 (11.25)	27.20 (10.81)	0.000
All students	112	20.00 (10.07)	28.94 (10.10)	0.000

As seen in Table 4, statistically significant learning occurred in both groups and also when comparing the scores of all students. This further supports our previous findings that ViLLE is highly beneficial tool when learning basic programming skills.

Reliability

To ensure the reliability of variables in the procedure, Cronbach's Alpha values were calculated for the questions in the pre- and post-test. The values (0.598 for pre-test and 0.815 for post-test) indicate high reliability.

We also looked at previous math scores of students participating in the study. The math scores studied were the grade of their last high school course and the grade of the math test students took when they begun computer science studies. We found no statistically significant difference in the math grades between the control and treatment group. However, we could only get math results from 8 students in the control group and from 23 in the treatment group.

Discussion

As the results show, the treatment group outperformed the control group in the post-test. There were no statistically significant differences in the pre-test but there was a statistically significant difference in the post-test total score. In addition, we found a trend favoring the treatment group in harder questions PQ5 and PQ6. The difference is almost statistically significant in PQ6, and there is absolute trend in PQ5. Since these questions were both about function calls, it seems that collaboration is most beneficial when tracking down the progress of execution between main and subprograms although the treatment group still outperformed the control group in every question. The statistically significant difference in the total score and the increase in absolute difference in all questions support the findings of Laakso et al. (2009), that the collaboration is highly beneficial when using visualizations.

Secondly, we wanted to find out if previous programming experience has any effect on the learning results. In our previous studies (see e.g. Rajala et al. (2008)) we have found out, that ViLLE benefits the novice students the most; however, it seems that when using the tool in collaboration with another student, the previous experience doesn't have statistically significant effect. In absolute scale, the novices in the treatment group did improve their results more, though.

Finally, we wanted to determine if learning occurred in all groups during the session. This was done by comparing the pre- and post-test (shared) scores with a pairwise t-test. The results support our previous findings: ViLLE can be used to greatly enhance the basic program comprehension skills of students. What makes the results even stronger is the short amount of time the tool was used. The learning effect seems to exist despite of the previous programming experience (total difference between the pre- and post-test scores varied between 7,90 and 9,56 points, and the difference was statistically significant in both groups regardless of previous programming experience).

All in all, collaboration seems to be quite natural form of using a program visualization tool. After a brief quiet period at the beginning of the session, the students engaged in lively discussion for the rest of the session. The pop-up questions in the visualizations activate students (Naps, 2005) and encourage students to start discussing the visualizations. We didn't gather the information about which of the students in the pair actually controlled the visualization; this information should be included in the future studies, since it might have substantial effect on the learning results.

Conclusions

We conducted a study about the benefits of collaboration in using a program visualization tool ViLLE. Our results indicate that collaboration seems to be most beneficial when tracing the execution of function calls. In addition, the statistical difference in total points of the post-test indicates that the collaboration seems to be a rather good method of using a visualization tool all in all; previous programming experience doesn't seem to have any effect on these results. Moreover, it seems that ViLLE can be used effectively to learn basic programming concepts, alone or in collaboration with another student. This further supports our earlier findings.

References

Bergin, S. & Reilly, R. (2005). The Influence of Motivation and Comfort-Level on Learning To Program. In Proceedings of the 17th Workshop on Psychology of Programming, PPIG'05.

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.

Kaila, E., Rajala, T., Laakso, M.-J. & 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. Appeared in 8th Koli Calling International Conference On Computing Education Research, November 13.-16., Joensuu, Finland.

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.

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., Myller, N. and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Journal of Educational Technology & Society, 12(2), 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-M. (2005). A Study of the Difficulties of Novice Programmers. ITiCSE'05. Caparica, Portugal, 14-18.

Lister, R., Adams, S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Ssnders, K., Seppälä, O., Simon, B. and Thomas, L. 82004). 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 Volume 3(2), 267-288.

McDowell, C., Werner, L., Bullock, H. & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course, Proceedings of the 33rd SIGCSE technical symposium on Computer science education, February 27-March 03, 2002, Cincinnati, Kentucky.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I. and 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.

Myller, N., Bednarik, R., Ben-Ari, M. & Sutinen, E. (2008). Extending the Engagement Taxonomy: Software Visualization and Collaborative Learning. Submitted to the Journal of Educational Resources in Computing (JERIC).

Naps, T.L. (2005). JHAVÉ – Addressing the Need to Support Algorithm Visualization with Tools for Active Engagement. *IEEE Computer Graphics and Applications 25*, 49-55.

Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. and Velásquez-Iturbide, J. Á. (2002). Exploring the Role of Visualization and Engagement in Computer Science Education. In Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education, 35, 2, 131-152.

Oechsle, R. and Schmitt, T. (2001). JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI). Revised Lectures on Software Visualization, International Seminar, May 20-25, 76-190.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2007). ViLLE – A language-independent program visualization tool. In 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, 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.

Urquiza-Fuentes, J. and Velázquez-Iturbide, J.Á. (2009). Pedagogical Effectiveness of Engagement Levels - A Survey of Successful Experiences. Electron. Notes Theor. Comput. Sci. 224, 169-178.

Williams, L., Kessler, R., Cunningham, W. & Jeffries, R. (2000). "Strengthening the Case for Pair Programming," IEEE Software, vol. 17, no. 4, pp. 19-25.

Williams, L.A. & Kessler, R.R. (2000). All I Really Need to Know About Pair Programming I Learned in Kindergarten. Communications of the ACM, vol. 43, no. 5, pp. 108-114.