Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 4 (December), pp. 668-680

Effectiveness of Program Visualization in Learning Java: a Case Study with Jeliot 3

S. Maravić Čisar, D. Radosav, R. Pinter, P. Čisar

Sanja Maravić Čisar, Robert Pinter
Subotica Tech-College of Applied Sciences, Department of Informatics
Serbia, 24000 Subotica, Marka Oreškovića 16
E-mail: sanjam@vts.su.ac.rs, probi@vts.su.ac.rs

Dragica Radosav
University of Novi Sad, Technical Faculty "Mihajlo Pupin" Zrenjanin,
Department of Informatics
Serbia, 23000 Zrenjanin, Djure Djakovica bb
E-mail: radosav@tfzr.uns.ac.rs

Petar Čisar
Telekom Srbija, Serbia, 24000 Subotica, Prvomajska 2-4
E-mail: petarc@telekom.rs

Abstract: This study was carried out to observe, measure and analyze the effects of using software visualization in teaching programming with participants from two institutions of higher educations in Serbia. Basic programming learning is notorious for complex for many novice students at university level. The visualizations of examples of program code or programming tasks could help students to grasp programming structures more easily. This paper describes an investigation about the possibilities of enhancement of learning Java using the visualization software Jeliot. An analysis of 400 students' test results indicates that a significant percentage of students had achieved better results when they were using a software visualization tool. In the experience of the authors Jeliot may yield the best results if implemented in with students who are new to the art of programming.
Keywords: software visualization, computer assisted learning, programming learning, Jeliot 3.

1 Introduction

Programming is a difficult cognitive skill to learn. Mastering the basis of a programming language is a huge problem for many students. In order to write a simple program they need to have a basic knowledge of variables, input/output of data, control structures and other areas. An even greater problem is mastering the more complex concepts such as pointers, abstraction or exception handling. And even when they have the necessary theoretical knowledge, the problem arises when they have to apply their knowledge as a whole and actually write a programming code.

Jenkins [1] identified reasons for these difficulties, such as:

- The need for good competence in problem solving;
- Students are used to courses that depend mostly on theoretical knowledge and memorization, but learning the basics of programming needs a more practical approach, based mostly on problem solving activities;

- Traditional teaching, generally based on lectures and specific programming language syntaxes, often fails to motivate students to get involved in meaningful programming activities;
- Programs have a dynamic nature, but most learning materials have a static format which makes it difficult to analyze the program's dynamic behavior;
- Students' learning needs are frequently very different within the same group. Different learning styles and previous experience make a common approach to the whole group rather difficult. Group sizes often make an individualized support to students difficult.

Educational software visualization tools are used to give students graphical representations of aspects of software systems that are inherently intangible and not obvious from them, such as the exact control flow, side effects in expression evaluations, and data dependencies. When students are given the ability to visually explore programs and algorithms, teachers expect them to be able to make better sense of program executions and programming concepts.

2 Related work

Boyle, Bradley, Chalk, Jones and Pickard [2] defined the new curriculum for London Metropolitan University's course of introductory programming. They specifically focused on visual approach. Over 600 students took part in the course. The increase in pass rates was from 12% to 23% compared to the previous year. Boyle et al. reported some significant problem concerning handling the course transition, but on average they described the graphical approach as 'very successful with the students'. Kannusmäki, Moreno, Myller and Sutinen [3] 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. Gathered data showed that those who were most successful in the course used Jeliot more than the other groups involved in the research. However, most of the students (in general) by and large still used other tools to code and test their programs. The usage problems reported were mostly of a technical nature or related to the usability of the editor. The animation was criticized for 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 [4] conducted a meta-study, analyzing 24 experimental studies on the effectiveness of algorithm visualization. They state that one of the main reasons for why 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 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. [4] also discovered that the sole use of visualization systems does not necessarily improve the learning results; it is more important to engage the learners in the subject using visualization system as an aid. Myller and Bednarik [5] presented their experiences with three empirical methodologies to study human behavior, interaction and learning with program visualization. Each of the approaches provides the research agenda with important source of data. Classroom studies inform about the practices taking place in this context and can generate testable hypotheses. Controlled experiments, when designed well, can provide answers to the previously established hypotheses and can give accurate insights into interaction and cognitive processes involved in programming. Data from surveys and questionnaire studies can be used both to collect data related to attitudes and current practices,

and generate testable hypotheses. Furthermore, all these methods can indicate issues for further development in the form of usability problems or unexpected behavior of users.

Programming is one of the essential areas taught in university studies of Computer Science and other engineering degrees and some experiences in teaching programming with e-learning application could be found in [6] and [7].

3 Software visualization

Software visualization is "the visualization of artefacts related to software and its development process" [8] and is used in the presentation, navigation and analysis of software systems. Price, Baecker and Small [9] presents the following general definition of software visualization: "Software visualization is 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." Given that the underlying purpose of algorithm visualization is to be educationally effective, it is worth nothing that eight extant taxonomic reviews of algorithm visualization and software visualization technology have focused largely on system expressiveness, as shown in Table 1.

SV taxonomy	Descriptive Dimensions
Myers	Aspect (code, data, algorithms) × Form (static, animated)
Shu	What is visualized (data or information about data, program and/or execution, software design)
Brown	Content (direct, synthetic) × Persistence (current, history) × Transformation (incremental, discrete)
Stasko and Patterson	Aspect × Abstractness × Animation × Automation
Kraemer and Stasko	Visualization task (data collection, data analysis, storage, display) × Visualization purpose (debugging, performance evaluation or optimization, program visualization)
Roman and Cox	Scope × Abstraction × Specification method × Interface × Presentation
Price et al.	Scopex Content × Form × Method × Interaction × Effectiveness

Table 1: The descriptive dimensions of the eight extant taxonomies of SV [4]

In particular, these taxonomies have focused on three main questions [9]:

1. What kinds of programs can be visualized with a given visualization system?
2. What kinds of visualizations can a given visualization system produce?
3. What methods can one use to produce and interact with visualizations?

The primary goal of visualization is to convey information. It should convey this information in an understandable, effective, easy-to-remember way [4]. Despite its intuitive appeal as a pedagogical aid, algorithm visualization technology has failed to catch on in mainstream computer science education [9]. While those few educators who are also algorithm visualization technology developers tend to employ their own algorithm visualization technology, the majority of computer science educators tend to stick to more traditional pedagogical technologies, such as blackboards, whiteboards and overhead projectors. Why do computer science educators tend not to use algorithm visualization technology? Instructors commonly cite several reasons, including the following:

- They feel they do not have the time to learn about it.

- They feel that using it would take away time needed for other class activities.
- They feel that creating visualizations for classroom use requires too much time and effort. Note that, in the algorithm visualization technology literature, this reason is frequently used to motivate new technology that is easier to use, and that supports the more rapid creation of visualizations.
- They feel that it is simply not educationally effective.

The reason that "it is simply not educationally effective" stands out as very important because there is no reason to adopt new technology if it does not bring some improvements in the educational process. So, the aim of this paper was to research the possibilities of enhancement in learning programming language Java using the visualization software Jeliot 3. The authors concentrate on effectiveness of (algorithm) visualizations i.e. how well one performs with the visualization tool compared to others who do not use the tool. The results of successful solution of certain programming problems are compared, solutions given by students who have used this tool, and those who have not. This will lead to answers regarding the efficient application of visualization software in education.

A number of classroom studies have been conducted in order to evaluate and analyze the usage of tools in the Jeliot family by using both qualitative and quantitative methods [8].

3.1 Jeliot 3

Jeliot 3 is a program visualization application. It visualizes how a Java program is interpreted. Method calls, variables, operation are displayed on a screen as the animation goes on, allowing the student to follow the execution of a program step by step as shown in Figure 1. Programs can be created from scratch or they can be modified from previously stored code examples; all the visualization is automatically generated [10]. Jeliot 3 understands most of the Java constructs and is able to animate them.

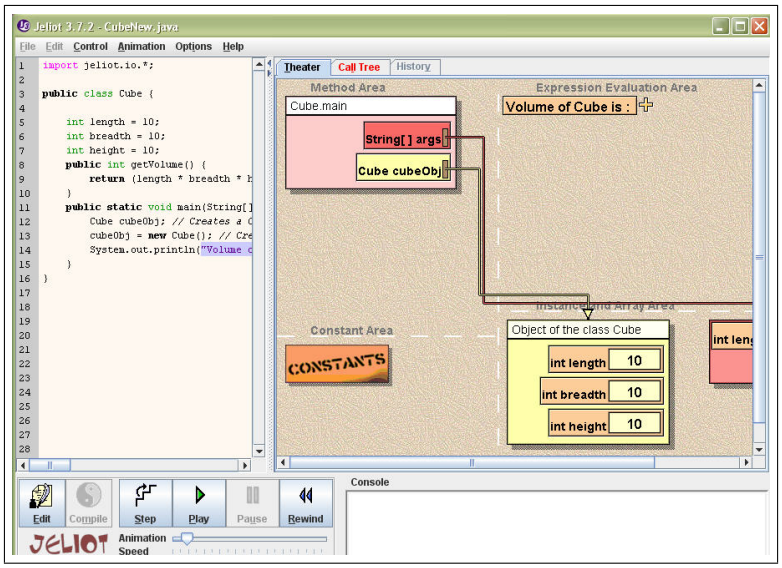

Figure 1: A screenshot of Jeliot 3

Jeliot 3 can be used in several ways for teaching and learning to program. Here are some examples from [3]:

- Lecturers can use Jeliot 3 as a part of the lecture material. They can explain the different concepts of programming through Jeliot animations. This will facilitate the construction by the students of the correct relationship between the animation and the concept, and enable them to apply it later with a reduced possibility of error [11].
- The students may use Jeliot 3 by themselves after lectures to do assignments.
- Jeliot 3 can be used in an interactive laboratory session, where students may utilize their recently acquired knowledge by writing programs and debugging them through Jeliot 3.
- Finally, Jeliot 3 provides a tool that can aid in courses where external help is not available (e.g. in distance education). Its visualization paradigm creates a reference model that can be used to explain problems by creating a common vocabulary between students and the teacher [11].

One of the possibilities that the software Jeliot 3 provides is the ability to select the *Ask Questions During Animation* option from the main menu. Whenever an expression is to be evaluated, a popup window will ask for the result. However, currently questions are generated only for assignment statements (Figure 2). The continuation of the animation is not possible until a student gives an answer to the question. In this way students have the opportunity to self evaluate their knowledge.

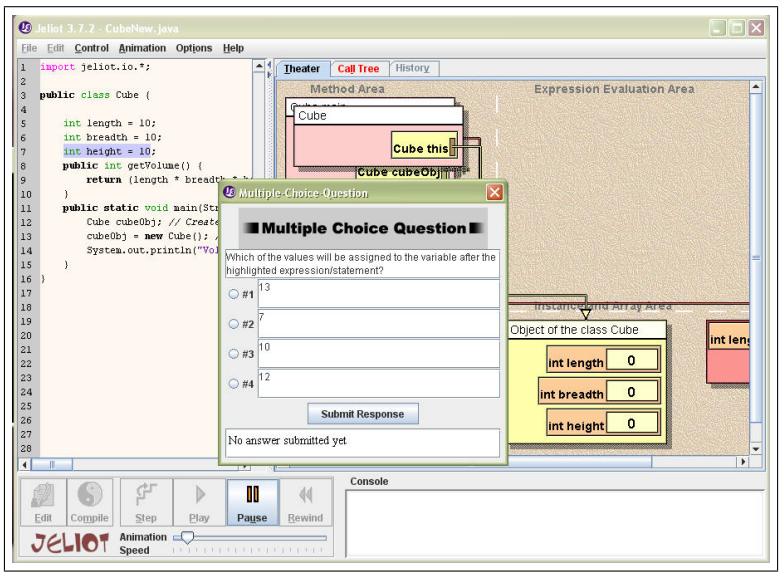

Figure 2: A screenshot with multiple choice questions

The Jeliot family's key feature has been the fully or semi-automatic visualization of the data and control flows. The development of the Jeliot family has taken more than ten years with different kinds of stages. Several versions of the concept have been developed, namely Eliot (developed at University of Helsinki, Finland), Jeliot I (developed at University of Helsinki, Finland), Jeliot 2000 (developed at Weizmann Institute, Israel). This has led to the stage when, as a product the software has become both usable and stable. The new version Jeliot 3 is a free piece of software published under General Public License (GPL). This means that the future platforms can be developed by networked teams presenting the idea of learning communities. In these communities the distinction between a teacher, a learner and a developer disappears, thus the learner can develop the tools he or she needs with the other members of the community.

Jeliot-together with its documentation, research publications, and learning materials-can be downloaded for free from http://www.cs.joensuu.fi/jeliot/.

4 Research

Wolfgram [12] states the following, "People only remember 15 percent of what they hear and 25 percent of what they see, but they remember 60 percent of what they interact with". The aim of this study was to explore the impact of the software for visualization Jeloit 3 in learning the programming language Java. Emphasis is placed on examining whether there are differences in the achievements of students who were taught Java with the help of Jeliot and those who were not. Java was selected as it has evolved to be most predominant and popular general purpose programming language of the current age. The research was the result of cooperation between two institutions of post-secondary education in Serbia: the Technical Faculty "Mihajlo Pupin" in Zrenjanin and Subotica Tech-College of Applied Science in Subotica. Preparations for the survey began in 2006, a year before the start of the experimental part of the research. Because the students from both institutions have similar education profile and many students from Subotica Tech continue their education in Zrenjanin's Technical Faculty, a team-based approach was taken for adjustment of the teaching materials. In this way one of the principals of the Bologna process of students' mobility was satisfied. The semester lasts 15 weeks and the students have a weekly lecture. The Java course consists of 30 hours of lectures and 30 hours of lab works. The lesson duration is 2x45 minutes. During the lectures the assignments for lab works were given to students. They had the task to analyze the problem at home and to draw an algorithm flow chart based on which they could write the program code at school, at lab works. At these practices students worked in pairs, two students per one PC.

Considering the fact that the participants in the survey were students from two different higher education institutions with different experience in object-oriented programming, and the fact that they were mainly beginners in programming, teaching process involved procedural programming and an introduction to object-oriented paradigm as in the material [13]. During the programming course the following topics were taught: variables, operators, decision-making statements (if-then, if-then-else, switch), the looping statements (for, while, do-while), the branching statements (break, continue, return), arrays (one-dimensional and two-dimensional), methods, constructors and inheritance. The research lasted for two school years (2007-2009) and 400 students were included. In the first year of study the sample consisted of 200 students. From the total number of respondents 47 were female (23.5%) and 153 were male students (76.5%). In the second year of study the sample consisted again of 200 students, among them 34 were female (17%) and 166 were male (83%). A certain part of the results of this research relating to only 45 students from Subotica Tech collected in the spring of 2009 could be found in [14]. The paper includes the results of a questionnaire that was used to collect reflections on programming and the application of the Jeliot program of students. The aim of this questionnaire was to find out how students had accommodated Jeliot into their learning processes.

In the research the model of pedagogical experiment with parallel groups was applied. Students were divided into three groups. The control group learned and worked in the traditional way, in the classrooms and under teachers monitoring. This group has lectures in the traditional way. The lecturers in this group used blackboard or whiteboard to present teaching material. All necessary explanations were drawn on the board, such as explaining the program code that was used as example during the class, or what took place during checking conditions in if-then-else statement, or in an algorithm flow chart. Students of the control group used JCreator (lightweight development environment for Java technologies) for writing and debugging code examples at the lab exercises, as well as for solving homework tasks. Students of this group used a textbook for

studying. For the first experimental group, in addition to the traditional ways of learning, an experimental factor was introduced - learning using visualization software Jeliot and PowerPoint presentations (blended learning) while in the second experimental group only e-teaching materials were applied. The lectures notes were in form of PowerPoint presentations. For explaining the program code that was used as example during the classes the lecturers used Jeliot. In this way the lectures become more interactive. Students of both experimental groups had a two-hour workshop at the beginning of the course to become familiar with it. The main purpose of Jeliot 3 is not to be used as a debugging tool, but it is convenient for analyzing smaller program codes as was used in this research. During the first month of the semester (a total of 8 lab exercises) they solved their tasks with the help of the lecturer. Based on the algorithm flow chart which they had to do as preparation for the class, they had to write program code in Jeliot and to use Jeliot as compiler. The students debugged the errors which were indicated by Jeliot together with the lecturer. After this first period of learning with Jeliot they had to complete all task alone (in groups of two students). Of course, during the entire class the lecturer was present and his role was to give an extra explanation if it was necessary.

Most of the time, students are trained to develop very simple programs starting from scratch. In fact, many students will be involved in evolution-related software after completion of their studies, so it is very important for them to know how to read and change existing large program codes [14]. However, evolving and understanding large existing software includes quite different activities such as recovering/understanding the actual architecture of the system, understanding some part of its source code and documentation (may be in some programming language that the student had not studied before). Including a new feature in such software then requires performing impact analysis, regression testing, etc [15]. Considering this, the next type of problems that students had was to figure out what an unknown program code did and to create appropriate test cases. Depending on which group they belonged to, students used Jeliot or JCreator to complete this task.

The comparison of the results of the control and experimental groups was performed by testing the hypothesis. Significant differences of the three groups of students were determined based on the critical values for probability levels of 0.05 and 0.01. The following hypothesis was stated: *Application of software for visualization Jeliot 3 does not affect the process of learning Java programming language.* The test was used to examine the students' knowledge. The exam was a traditional paper-and-pencil test with 20 multiple choice questions, where a standard grading procedure has been applied to grade these test items (one point for a correct answer, zero points for an incorrect one). The statistical analysis of the performed test included calculation of the arithmetic mean, standard deviation, standard error of the mean and interval of variation. The one-way ANOVA was used to test for differences among three groups of students. In order to determine which groups differ from each other the post-test for pair-wise comparisons Tukey's HSD test (honestly significant difference) was implemented. Statistical analysis of the results was done using the site VassarStats [16] that offers the possibility of online statistical calculations.

The total number of students who participated in the first year of the research was 200. The control group consisted of 60 students who had lectures in the traditional way. In the first experimental group there were 65 students who had the traditional way of lecturing accompanied with the PowerPoint presentations and software for the visualizations. In the second experimental group there were 75 students who only had e-learning material. The results of the research are given in Table 2.

The analysis of the descriptive statistical parameters has shown that the average value of the scored points for the first experimental group was the largest (11.05) with a standard deviation of 4.78 points. The lowest average of points had a control group, 7.33 with standard deviation of 4.85 points. Based on these results, it can be said that the first experimental group had the

Group	N	Mean	Std.Dev.	Std.Err.	Min	Max
Control	60	7.3333	4.8526	0.6265	0	18
Experimental I	65	11.0462	4.7777	0.5926	0	20
Experimental II	75	10.9067	5.5585	0.6418	0	20

Table 2: Descriptive statistical parameters of the first year of the research

highest average score and thus showed the best results on the test. Graphical representation of the average number of points is given in Figure 3.

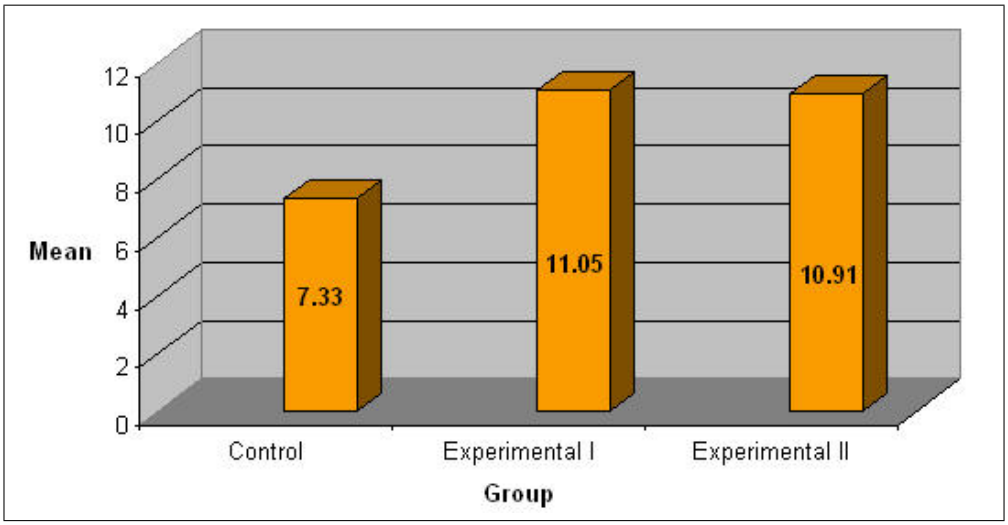

Figure 3: The average number of points in the first year of the experiment

The results of the ANOVA analysis are shown in Table 3 ($p \leq 0.05$).

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	556.5784615	2	278.2892308	10.67313056	3.97E-05	3.04175303
Within Groups	5136.541538	197	26.07381492			
Total	5693.12	199				

Table 3: The results of the ANOVA analysis in the first year of the experiment ($p \leq 0.05$)

The results of the ANOVA analysis are shown in Table 4 ($p \leq 0.01$).

The calculated value for F of 10.67 indicates significant differences between the studied groups. Significant differences between individual groups were determined using the Tukey test. The absolute (unsigned) difference between any two sample means required for significance at the designated level (M1=mean of the control group, M2=mean of the first experimental group and M3=mean of the second experimental group) is:
M1-M2=3.7162
M1-M3=3.5767
M2-M3=0.1395

Next the HSD is computed (HSD). If the difference is larger than the HSD, then the difference is said to be significant.

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	556.5784615	2	278.2892308	10.67313056	3.97E-05	4.71452043
Within Groups	5136.541538	197	26.07381492			
Total	5693.12	199				

Table 4: The results of the ANOVA analysis in the first year of the experiment ($p \leq 0.01$)

HSD[.05]=2.1; HSD[.01]=2.62
M2 vs. M3 insignificant
M1 vs. M3 $P < .01$
M1 vs. M2 $P < .01$

Based on the results of HSD, it can be concluded that there is a significant difference between the control and experimental groups I and II ($p < 0.01$). The hypothesis of equality of the control and experimental groups cannot be accepted, which means that the implementation of software for visualization Jeliot 3 does have an influence on the process of learning Java. No significant differences between the experimental groups have been recognized.

The total number of students who participated in the second year of the research was 200. The control group consisted from 68 students who had lectures in the traditional way. In the first experimental group there were 66 students who had the traditional way of lecturing accompanied with the PowerPoint presentations and software for the visualizations. In the second experimental group there were 76 students who had only e-learning material. The results of the research are given bellow in Table 5.

Group	N	Mean	Std.Dev.	Std.Err.	Min	Max
Control	68	7.2206	4.6548	0.5645	0	17
Experimental I	66	10.8333	4.6923	0.5776	1	19
Experimental II	66	11.5	5.3787	0.6621	1	20

Table 5: Descriptive statistical parameters of the second year of the research

The analysis of the descriptive statistical parameters has shown that the average value of the scored points for the second experimental group was the largest (11.5) with a standard deviation of 5.378 points. The lowest average of points had a control group, 7.22 with a standard deviation of 4.65 points. Based on these results, it can be said that the second experimental group had the highest average score and thus showed the best results on the test. The graphical representation of the average number of points is given in Figure 4.

The results of the ANOVA analysis are shown in Table 6 ($p < 0.05$).

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	713.5172	2	356.7586	14.7546	1.07E-06	3.04175303
Within Groups	4763.358	197	24.17948			
Total	5476.875	199				

Table 6: The results of the ANOVA analysis in the second year of the experiment ($p \leq 0.05$)

The calculated value for F of 14.75 indicates the existence of significant differences between

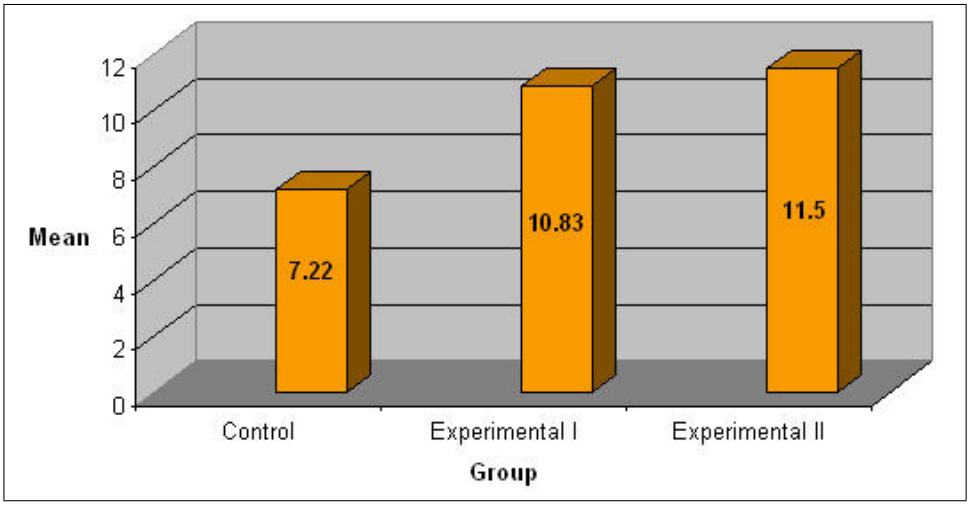

Figure 4: The average number of points in the second year of the experiment

the studied groups. Significant differences between individual groups were determined using the Tukey test. The absolute (unsigned) difference between any two sample means required for significance at the designated level is:
M1-M2=-3.6127
M1-M3=-4.2794
M2-M3=0.6667.

The following step is computing HSD:
HSD[.05]=2.01; HSD[.01]=2.51
M1 vs. M2 $P < .01$
M1 vs. M3 $P < .01$
M2 vs. M3 insignificant.

The results of the ANOVA analysis are shown in Table 7 ($p \leq 0.01$).

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	713.5172	2	356.7586	14.7546	1.07E-06	4.71452043
Within Groups	4763.358	197	24.17948			
Total	5476.875	199				

Table 7: The results of the ANOVA analysis in the second year of the experiment ($p \leq 0.01$)

Based on the results of HSD, it can be concluded that there is a significant difference between the control and experimental groups I and II ($p < 0.01$). The hypothesis of equality of the control and experimental groups cannot be accepted, which means that the implementation of the software for visualization Jeliot 3 does have an influence on the process of learning Java. There are no recognizable differences between the experimental groups. Based on the results of the two-year research in which a total of 400 students have been included it can be stated that the experimental groups I and II have been highly successful compared with the control group in learning Java. Thus, the set hypothesis that application of the software for visualization Jeliot 3 has no influence on the process of learning Java programming language was refuted. Between the experimental groups no significant statistical difference was found.

Students who were in the experimental groups after completing the course filled out the survey in order to obtain information about their opinion on learning using visualization software. First,

at the middle of the semester, students' satisfaction with their progress in learning and their motivation for further work was examined. To express views on the motivation of the students the Likert scale with 5 responses was applied (very high, high, medium, low and very low). The results are shown in Fig. 5 (all values are expressed in percentages).

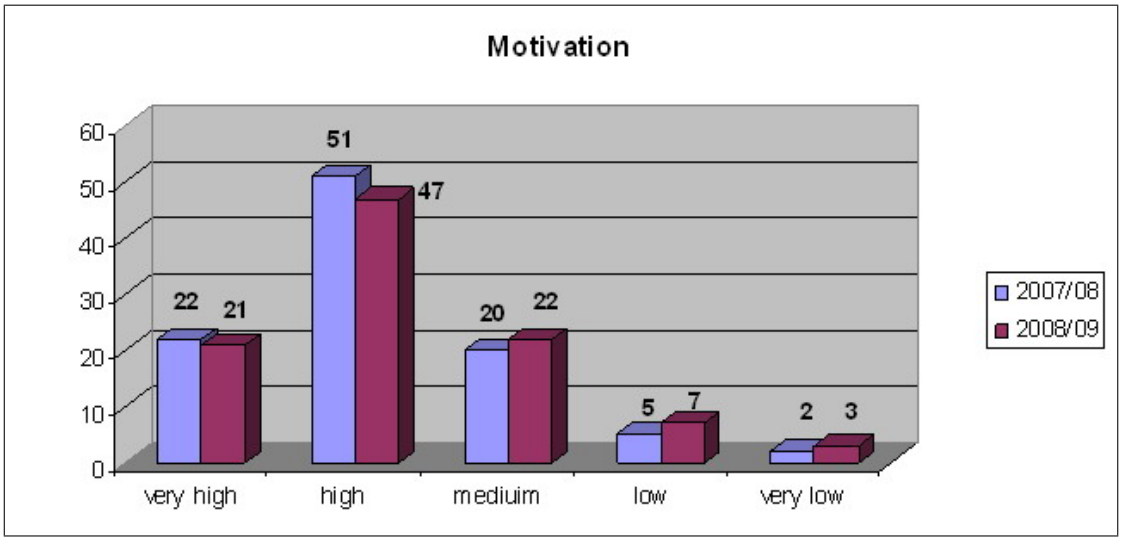

Figure 5: The motivation of the students

Figure 6 shows the results (given in percentages) of students' satisfaction in their learning progress. They could express their attitude with the following responses: very satisfied, satisfied, neutral, dissatisfied, very dissatisfied (Likert scale).

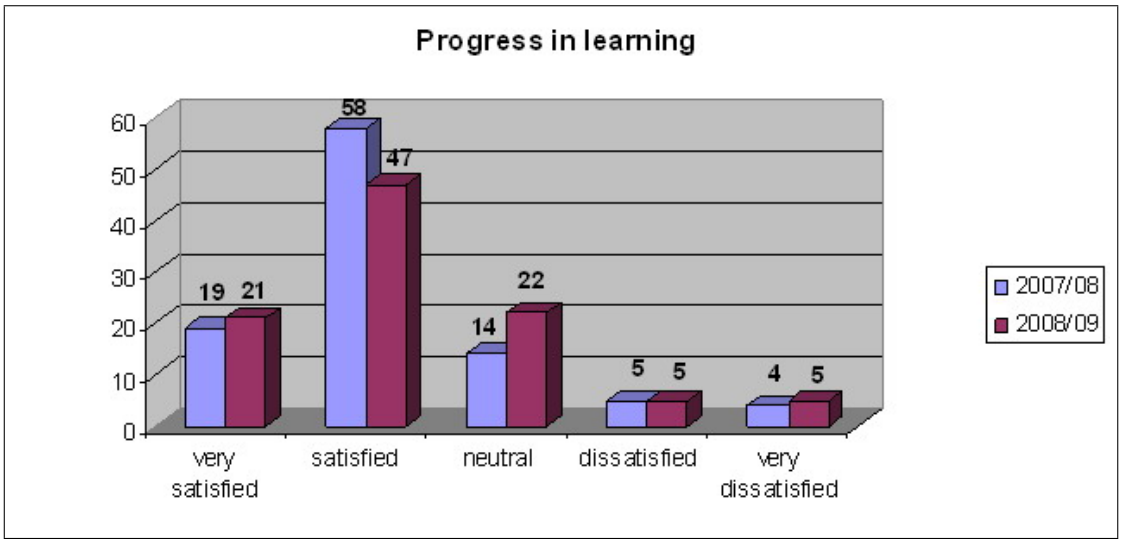

Figure 6: Student's satisfaction in learning progress

The students of the experimental group were asked what they think about the educational possibilities of Jeliot 3. Their answers depended on the level of knowledge of Java and object-oriented programming concepts in general. Responses were ranged from the answer that Jeliot3 is a great help for beginners, that it is a powerful educational tool thanks to the visualization and because of that, it made learning Java easy, to the opinions of some students who have already had experience in Java programming that it was a pure waste of time for them. Students

who expressed a negative opinion about Jeliot said that it was difficult for them to adapt their knowledge of object-oriented programming and their style of programming to the new tool, also they had objections about the elements of visualization code which were disturbing their attention. What all the negative comments had in common was that they were given by students who had already used the Java language, but almost all of them pointed out that they believed that Jeliot could be very useful at the beginning of the process of learning Java. The results of the research showed that the passing rate of the exam increased for a few percent when using the Jeliot3 (for 3% in the first year and 2.67% in the second year of research). But, in order to be able to claim that result of increased passing is the consequence of using Jeliot3, it would be necessary to perform some additional research focused solely on that piece of information.

5 Conclusions

The paper discussed the problem of the applications of software for visualization Jeliot3 in learning the programming language Java. The study involved 400 students of two higher education institutions in Serbia. Based on the research, which lasted for two years, it can be stated that there are significant differences in the achievements of students who were taught in the traditional way, and those who have used Jeliot3. Through well-defined messages from the compiler Jeliot indicates to the user where the syntax errors in code are. The goal is to help novices understand basic concepts of algorithms and programming like assignment, I/O and control flow, whose dynamic aspects are not easily grasped just by looking at the static representation of an algorithm in a programming language [11]. Software Visualization tools are intended to be used in the early stages of the learning path of a programmer, teaching the students the basics of programming, algorithms, and the software development cycle [17]. While a future experiment involving Jeliot and more advanced students (in terms of programming) may provide interesting results, it is the belief of the authors that this particular program works best with beginner students.

It is not possible to "eject" teachers from the teaching process and replace them with a computer, but it is necessary to ensure their active participation during the animation code so that students receive the necessary explanation. Using visualization tools should be intense, in the sense that teachers need to use it in the teaching process, and students in laboratory exercises as well as when working at home. Of course, further research in this field is required. Although this study did not directly deal with the influence of Jeliot to accelerate learning processes, based on previous research of the authors [18] about the visualization of some selected parts of the course Analogue and Digital Electronics that has shown that interactive animations can significantly contribute to increasing the speed of learning, it can be assumed that Jeliot could be especially helpful to beginners in learning Java in the same way. It must be noted that it is already clear there is a need for different tools for learning depending on the level of students' knowledge of the studied materials. Advanced students and even students with only superficial experience in programming are very sensitive to the change of tools that are used as a code editor, if it does not provide significant improvement over the tools they are used on. In other words, the individual characteristics of students, including the level of knowledge, must be taken into thorough consideration because the demands of students are changing rapidly.

Bibliography

[1] T. Jenkins, "On the Difficulty of Learning to Program", *in Proc. of 3rd Annual LTSN-ICS Conference*,Loughborough University, UK, 53-58, 2002.

[2] T. Boyle, C. Bradley, P. Chalk, R. Jones, P. Pickard, 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, 2003.

[3] O. Kannusmäki, A. Moreno, N. Myller, E. Sutinen. What a novice wants: Students using program visualization in distance programming course, *Proc. of the Third Program Visualization Workshop (PVW'04)*, Warwick, UK, pp. 126-133, 2004.

[4] C. D. Hundhausen, S. A. Douglas, J. T. Stasko, A Meta-Study of Algorithm Visualization Effectiveness, *Journal of Visual Languages and Computing*, 259-290, 2002.

[5] N. Myller, R. Bednarik, Methodologies for studies of program visualization, *Proc. of the Methods, Materials and Tools for Programming Education Conference*, 37-42, 2006.

[6] M. D. Afonso Suarez, C. Guerra Artal, F. M. Tejera Hernandez, E-learning multimedia applications:Towards an engineering of content creation, *Int. J. of Computers, Communications & Control*, 3(2): 116-124, 2008.

[7] C. Guerra Artal, M. D. Afonso Suarez, I. Santana Perez, R. Quesada Lopez, OLC, On-Line Compiler to Teach Programming Languages, *Int. J. of Computers, Communications & Control*, 3(1): 69-79, 2008.

[8] S. Diehl, Evolution, *In Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software Springer Verlag*, pp. 149-160, 2007. [Online]. Available: `http://www.springerlink.com/content/m373254212740552/fulltext.pdf`

[9] B. A. Price, R. M. Baecker, I. S. Small, An Introduction to Software Visualization, in Software Visualization, J. Stasko, J. Dominque, M. Brown, B. Price (Eds.), London, England MIT Press, 4-26, 1998.

[10] [Online]. Available: `http://cs.joensuu.fi/jeliot/description.php`

[11] R. Ben-Bassat Levy, M. Ben-Ari, P. A. Uronen, The Jeliot 2000 program Animation System, *Computers & Education*, 40(1): 1-15, 2003.

[12] D. E. Wolfgram, Creating multimedia presentations, Que Corp, IN, USA, 1994.

[13] [Online]. Available: `http://stwww.weizmann.ac.il/g-cs/benari/lov/lov.html`

[14] S. Maravić Čisar, R. Pinter, D. Radosav, P. Čisar, Software Visualization: the Educational Tool to Enhance Student Learning, *Proc. of 33rd International Convention MIPRO 2010, Computers in Education*, May 24-28, 2010, Opatija, Croatia, ISSN 1847-3938, ISBN 978-953-233-054-0, 4: 234-238, 2010.

[15] A. Van Deursen, J. M. Favre, Experiences in Teaching Software Evolution and Program Comprehension. Available: `http://www.tzi.de/st/papers/teaching-iwpc03.pdf`

[16] Available: `http://faculty.vassar.edu/lowry/VassarStats.html`

[17] A. Moreno, M. S. Joy, Jeliot 3 in a Demanding Educational Setting, *Fourth International Program Visualization Workshop*, 29-30 June 2006, Florence, Italy

[18] R. Pinter, D. Radosav, S. Maravić Čisar, Interactive Animation in Developing e-Learning Contents, *Proceedings of 33rd International Convetion MIPRO 2010, Computers in Education*, May 24-28, 2010, Opatija, Croatia, ISSN 1847-3938, ISBN 978-953-233-054-0, 4: 251-254