Informatics in Education, 2009, Vol. 8, No. 1, 17–34

Effects, Experiences and Feedback from Studies of a Program Visualization Tool

Erkki KAILA, Teemu RAJALA, Mikko-Jussi LAAKSO,
Tapio SALAKOSKI
TUCS, Turku Centre for Computer Science, University of Turku
Joukahaisenkatu 3–5 B, 6th floor, FI-20520 Turku, Finland
e-mail: {ertaka, temira, milaak, sala}@utu.fi

Received: December 2008

Abstract. Program visualization (PV) is potentially a useful method for teaching programming basics to novice programmers. However, there are very few studies on the effects of PV. We have developed a PV tool called ViLLE at the University of Turku. In this paper, multiple studies on the effects of the tool are presented. In addition, new qualitative data about students' feedback of using the tool is presented. Both, the results of our studies and the feedback indicate that ViLLE can be used effectively in teaching basic programming concepts to novice programmers.

Keywords: program visualization, programming, education, effects,experiences, student feedback.

1. Introduction

Programming is one of the main objectives in computer science studies. However, according to several studies (see, e.g., McCracken *et al.* (2001) and Ala-Mutka (2005)) students have significant problems in learning the very basics of programming. Because of scarce teaching resources and large group sizes students often get inadequate personal instruction. Thus, there is clearly a need for instructional tools that support students' independent learning.

Visualization systems use various graphical means in concretizing abstract programming and algorithmic problems. According to Wiggins (1998), the purpose of visualization is to help the user understand *what* a program does, *why* it does it, *how* it works, and *what* the end result is. Hence, visualization systems can supposedly help students to understand programs better, thus improving the learning results. There are several studies on the effects of different algorithm visualization systems (see, e.g., Hundhausen *et al.* (2002) and Laakso *et al.* (2008a)), but very few on program visualization.

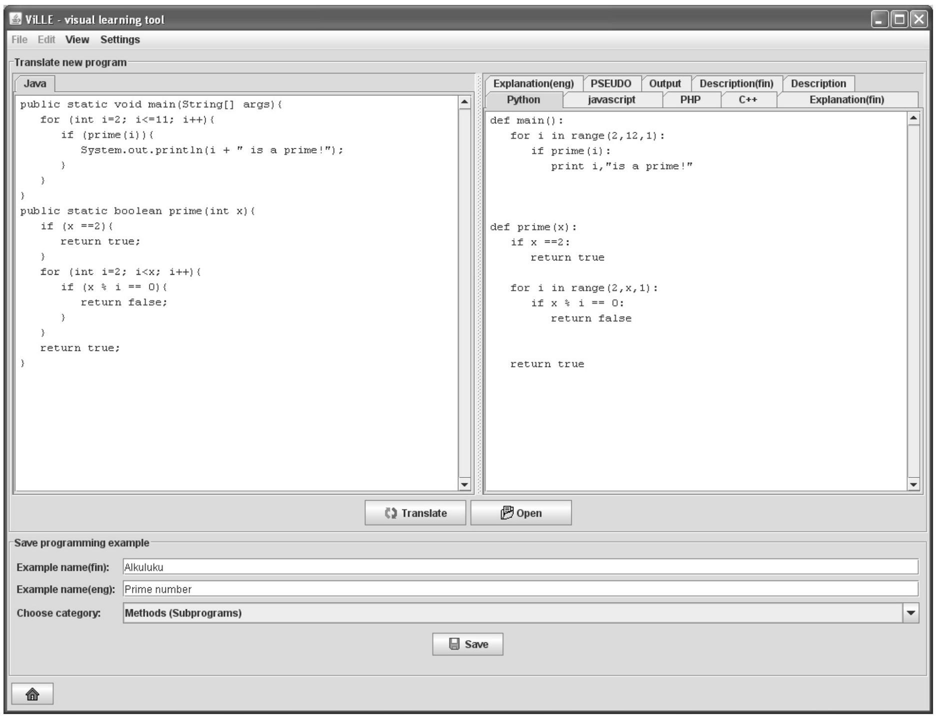
We have developed a program visualization tool called ViLLE at the University of Turku, Finland. The purpose of developing the tool was to find out if program visualization can indeed help students in learning to program. ViLLE has been tested with different kind of setups taking student competence levels and backgrounds into account. In this paper we describe the tool itself for both in teacher's and student's point of view and present the results of our studies on the effects of ViLLE so far. Moreover, we present new quantitative results, based on the student feedback about using the tool. Finally, the future of ViLLE is discussed.

2. ViLLE

ViLLE is a program visualization tool, developed at the University of Turku. Its main purpose is to illustrate the changes in the programs states during the execution with various graphical and textual means. ViLLE supports multiple programming languages and has built-in editors for defining and editing syntaxes, examples and questions. With the export function the defined examples can be published as an independent collection, distributable in web or other media. By using the TRAKLA server (see Malmi *et al.* (2004)), ViLLE's automatically assessed exercises can be easily integrated as a part of a programming course.

2.1. *Teacher's Point of View*

From a teacher's point of view, the suitability for different kinds of courses can be seen as the number one feature of ViLLE. Programming languages, examples and questions are all customizable with the built-in editors. Hence, it should be relatively easy to integrate



```
public static void main(String[] args){
   for (int i=2; i<=11; i++){
      if (prime(i)){
         System.out.println(i + " is a prime!");
      }
   }
}
public static boolean prime(int x){
   if (x ==2){
      return true;
   }
   for (int i=2; i<x; i++){
      if (x % i == 0){
         return false;
      }
   }
   return true;
}
```

```
def main():
   for i in range(2,12,1):
      if prime(i):
         print i,"is a prime!"



def prime(x):
   if x ==2:
      return true

   for i in range(2,x,1):
      if x % i == 0:
         return false


   return true
```

Example name(fin): Alkuluku
Example name(eng): Prime number
Choose category: Methods (Subprograms)

Fig. 1. Example editor in ViLLE.

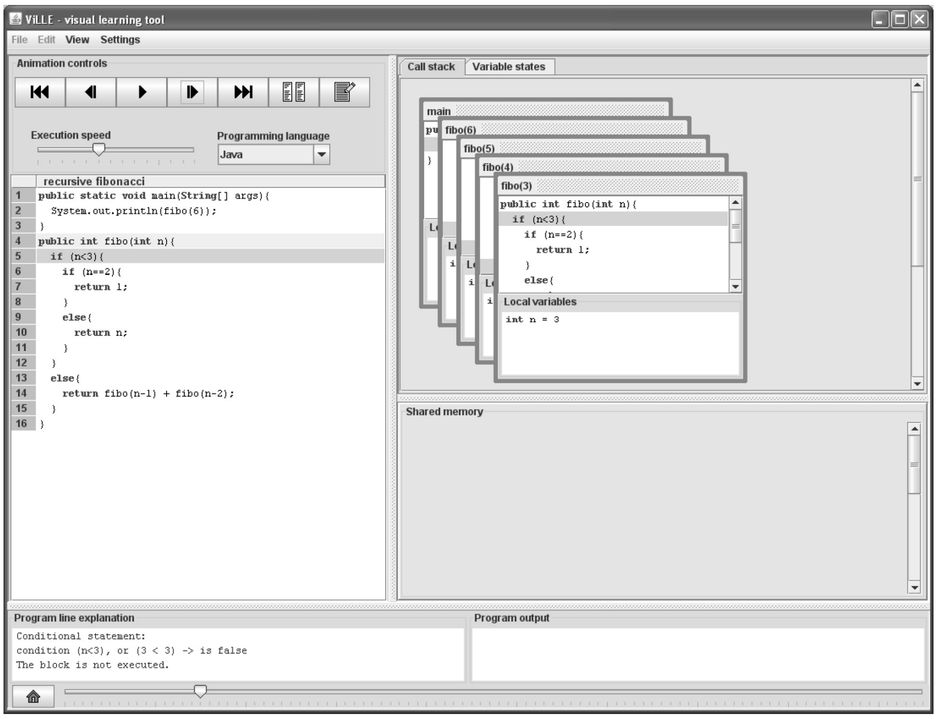
ViLLE into almost any programming course. Moreover, the flexibility gives teachers a chance to fulfil their own teaching philosophy, without a need to adjust to tool's limitations.

2.1.1. *Example Editor*

Examples are divided into categories. Teacher can create new examples and edit the existing ones by using the built-in example editor (see Fig. 1). The examples are written in Java, and ViLLE automatically translates the program code to all defined languages. Moreover, the visualizations of examples are automatically generated, as well as explanations on each program line. The list of supported Java features is limited, mainly, because all the programs should be easily translatable to other defined languages, and because the tool is directed for novice programmers. Furthermore, the visualization of more complex features (e.g., GUI components) can get quite tricky.

2.1.2. *Question Editor*

To further engage the learners to visualization, teacher can create questions about the example programs. Currently multiple choice questions and graphical array questions are supported. The questions are attached to program execution with a built-in question editor (see Fig. 2), and are automatically launched when desired point of program is reached.



recursive fibonacci

```
public static void main(String[] args){
  System.out.println(fibo(6));
}
public int fibo(int n){
  if (n<3){
    if (n==2){
      return 1;
    }
    else{
      return n;
    }
  }
  else{
    return fibo(n-1) + fibo(n-2);
  }
}
```

```
int n = 3
```

```
Conditional statement:
condition (n<3), or (3 < 3) -> is false
The block is not executed.
```

Fig. 2. Question editor.

The TRAKLA2 server can be used with ViLLE to keep the score of answered questions and correct answers.

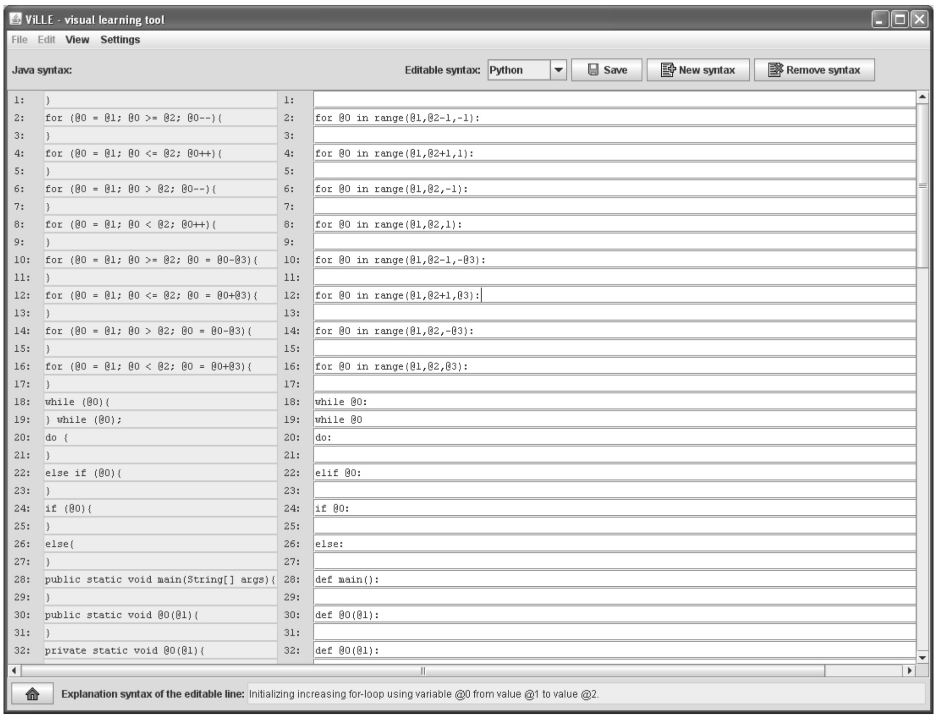
2.1.3. *Syntax Editor*

Teacher can add or edit syntaxes with the built-in syntax editor (Fig. 3). The editable syntax is presented on the right hand side of the window, and the correspondent Java syntax on the left hand side. Additionally, an explanation of the edited syntax line is presented in the bottom window. All defined syntaxes must have matching lines with the Java syntax to ensure the flow of execution and the consistency of questions.

2.2. *Student's Point of View*

ViLLE's key features from the student's point of view can be divided into following categories:

Visualizing the program execution: The execution of the example program is visualized line by line (see Fig. 4). Currently and previously executed lines are highlighted, and the variable values, program line explanation and the output of the program are presented in their own frames. All subprograms, and their return values, are presented in the call order in their own frames in the call stack. Moreover, all global variables (namely arrays) are presented graphically in their own area.

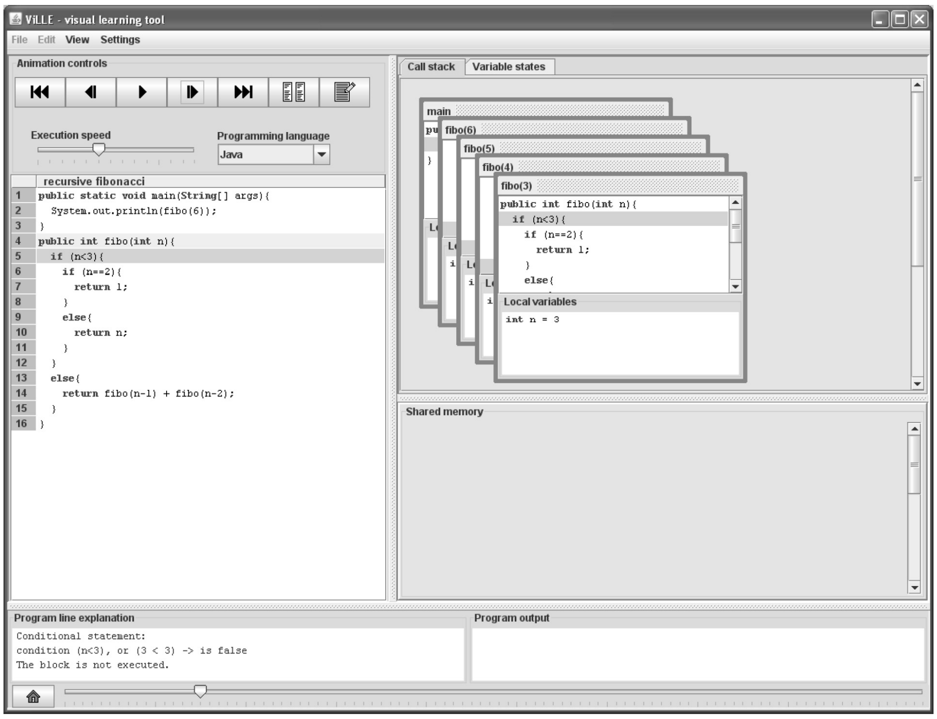


Editable syntax: Python

Java syntax:	
`}`	
`for (@0 = @1; @0 >= @2; @0--){`	`for @0 in range(@1,@2-1,-1):`
`}`	
`for (@0 = @1; @0 <= @2; @0++){`	`for @0 in range(@1,@2+1,1):`
`}`	
`for (@0 = @1; @0 > @2; @0--){`	`for @0 in range(@1,@2,-1):`
`}`	
`for (@0 = @1; @0 < @2; @0++){`	`for @0 in range(@1,@2,1):`
`}`	
`for (@0 = @1; @0 >= @2; @0 = @0-@3){`	`for @0 in range(@1,@2-1,-@3):`
`}`	
`for (@0 = @1; @0 <= @2; @0 = @0+@3){`	`for @0 in range(@1,@2+1,@3):`
`}`	
`for (@0 = @1; @0 > @2; @0 = @0-@3){`	`for @0 in range(@1,@2,-@3):`
`}`	
`for (@0 = @1; @0 < @2; @0 = @0+@3){`	`for @0 in range(@1,@2,@3):`
`}`	
`while (@0){`	`while @0:`
`} while (@0);`	`while @0`
`do {`	`do:`
`}`	
`else if (@0){`	`elif @0:`
`}`	
`if (@0){`	`if @0:`
`}`	
`else{`	`else:`
`}`	
`public static void main(String[] args){`	`def main():`
`}`	
`public static void @0(@1){`	`def @0(@1):`
`}`	
`private static void @0(@1){`	`def @0(@1):`

Explanation syntax of the editable line: Initializing increasing for-loop using variable @0 from value @1 to value @2.

Fig. 3. Syntax editor.



```
public static void main(String[] args){
  System.out.println(fibo(6));
}
public int fibo(int n){
  if (n<3){
    if (n==2){
      return 1;
    }
    else{
      return n;
    }
  }
  else{
    return fibo(n-1) + fibo(n-2);
  }
}
```

```
public int fibo(int n){
  if (n<3){
    if (n==2){
      return 1;
    }
    else{
```

```
int n = 3
```

```
Conditional statement:
condition (n<3), or (3 < 3) -> is false
The block is not executed.
```

Fig. 4. Visualization view.

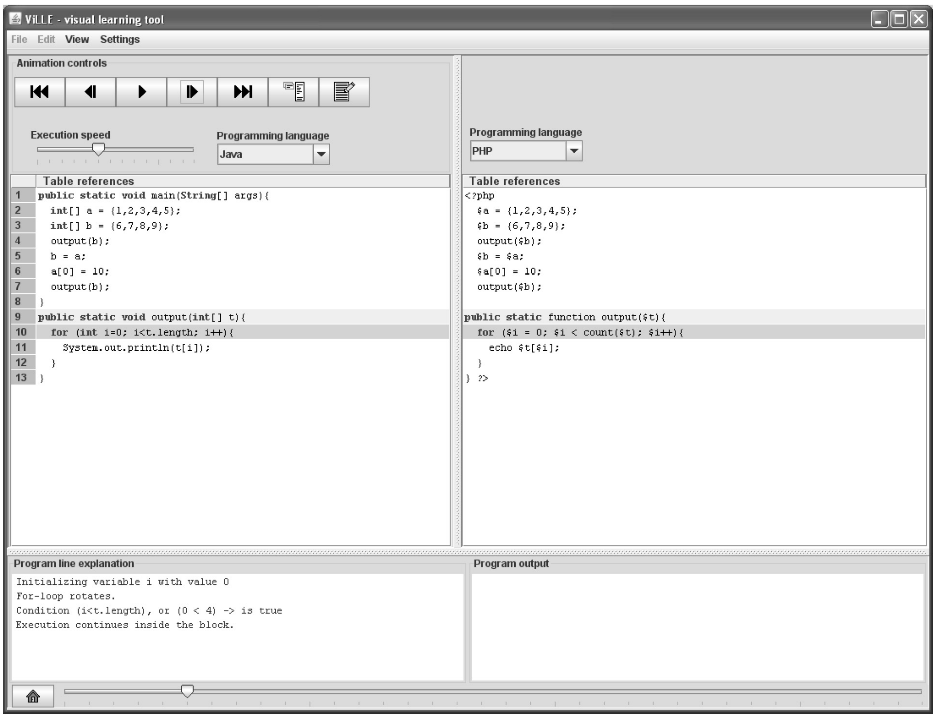
Language independency: The visualization of the execution works similarly, regardless of the chosen programming language. Programming language can be changed anytime during the visualization. In addition, ViLLE has a parallel mode (see Fig. 5) where execution can be viewed in two different languages simultaneously.

Visualization controls: Controls are flexible – the user can move one step at a time, both forwards and backwards in the execution of a program. In addition, the program can be executed continuously in adjustable speed. The execution slider at the bottom of the window can be used to move directly to a desired state of execution. The slider also has a secondary function: the number of steps can be used to determine the complexity of the program, and with suitable examples to compare the complexities of algorithms.

Interaction: Besides answering questions (see Fig. 6) the students can edit the program code in the visualization view. The changes in program code can be visualized instantly. However, since the editing must be done in Java, the feature can't naturally be utilized in all courses.

2.3. *Automatic Assessment of Exercises*

Examples made with ViLLE can be transferred to a collection in a web (see Fig. 7) by using the TRAKLA2 server. The server keeps score on students' logins and submissions, and makes it possible to set opening and expiration dates for example rounds. Individual



```
public static void main(String[] args){
  int[] a = {1,2,3,4,5};
  int[] b = {6,7,8,9};
  output(b);
  b = a;
  a[0] = 10;
  output(b);
}
public static void output(int[] t){
  for (int i=0; i<t.length; i++){
    System.out.println(t[i]);
  }
}
```

```
<?php
  $a = {1,2,3,4,5};
  $b = {6,7,8,9};
  output($b);
  $b = $a;
  $a[0] = 10;
  output($b);

public static function output($t){
  for ($i = 0; $i < count($t); $i++){
    echo $t[$i];
  }
} ?>
```

```
Initializing variable i with value 0
For-loop rotates.
Condition (i<t.length), or (0 < 4) -> is true
Execution continues inside the block.
```

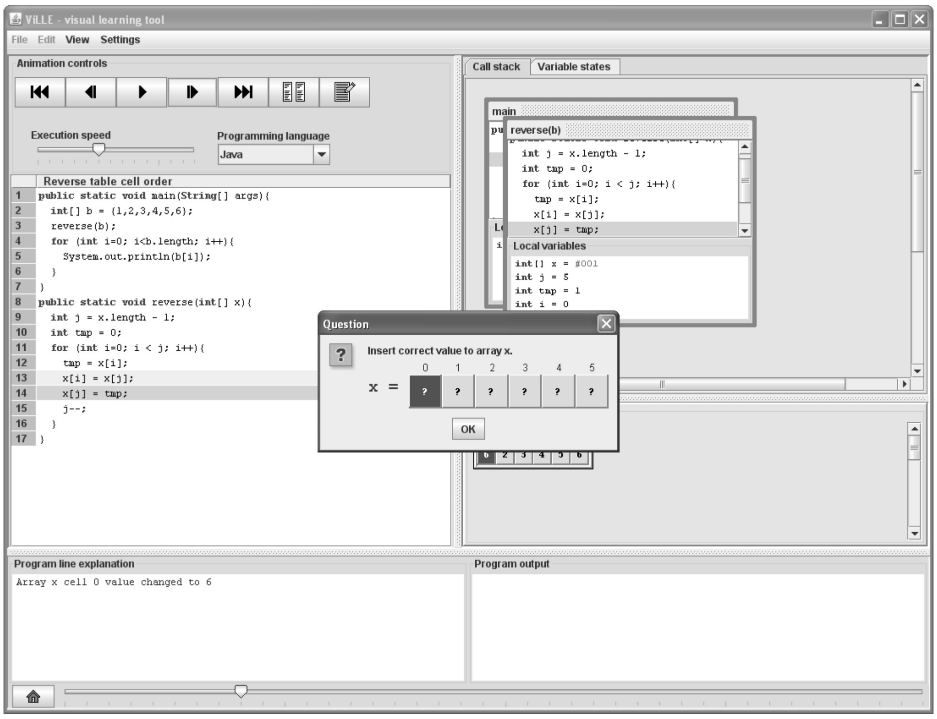
Fig. 5. Parallel view.

examples can be retaken an unlimited number of times. Additionally, the teacher can set the minimum number of points required for each round to pass the course. Web-based ViLLE exercises are nowadays in use in most universities in Finland, and the student feedback (see Subsection 3.4) has been mostly positive: it seems that ViLLE has a positive role in enhancing the reading and tracing skills of novice programmers.

3. The Studies on ViLLE

3.1. *The Effectiveness of ViLLE*

The effectiveness of ViLLE was studied at the University of Turku, Finland, in a course called ”Introduction to information technology”. There were 72 students participating in the study. We tried to find the answer to two research questions: 1) “Is ViLLE useful in learning basic programming concepts?” and 2) “Is there any difference in learning results when earlier programming experience is taken into account?”. The null-hypotheses were that ViLLE is not useful in learning to program, and that the effect is same for both novices and more experienced students. No programming was taught before the study, taking place in the third week of the course. However, a lecture was arranged before the study where the programming language and the tool used were introduced. A link



Reverse table cell order

```
public static void main(String[] args){
  int[] b = {1,2,3,4,5,6};
  reverse(b);
  for (int i=0; i<b.length; i++){
    System.out.println(b[i]);
  }
}
public static void reverse(int[] x){
  int j = x.length - 1;
  int tmp = 0;
  for (int i=0; i < j; i++){
    tmp = x[i];
    x[i] = x[j];
    x[j] = tmp;
    j--;
  }
}
```

Local variables

```
int[] x = #001
int j = 5
int tmp = 1
int i = 0
```

Insert correct value to array x.

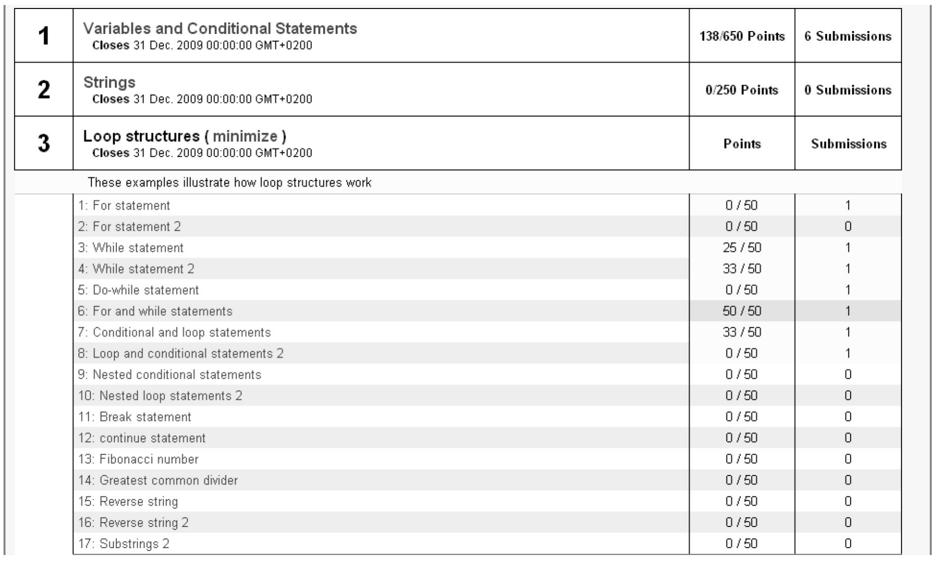
x =	0	1	2	3	4	5
	?	?	?	?	?	?

OK

Program line explanation

```
Array x cell 0 value changed to 6
```

Fig. 6. Array question in visualization view.



1	Variables and Conditional Statements **Closes** 31 Dec. 2009 00:00:00 GMT+0200	138/650 Points	6 Submissions
2	Strings **Closes** 31 Dec. 2009 00:00:00 GMT+0200	0/250 Points	0 Submissions
3	**Loop structures** (minimize) **Closes** 31 Dec. 2009 00:00:00 GMT+0200	Points	Submissions
	These examples illustrate how loop structures work		
	1: For statement	0 / 50	1
	2: For statement 2	0 / 50	0
	3: While statement	25 / 50	1
	4: While statement 2	33 / 50	1
	5: Do-while statement	0 / 50	1
	6: For and while statements	50 / 50	1
	7: Conditional and loop statements	33 / 50	1
	8: Loop and conditional statements 2	0 / 50	1
	9: Nested conditional statements	0 / 50	0
	10: Nested loop statements 2	0 / 50	0
	11: Break statement	0 / 50	0
	12: continue statement	0 / 50	0
	13: Fibonacci number	0 / 50	0
	14: Greatest common divider	0 / 50	0
	15: Reverse string	0 / 50	0
	16: Reverse string 2	0 / 50	0
	17: Substrings 2	0 / 50	0

Fig. 7. ViLLE exercises in TRAKLA2 environment.

to ViLLE was added to course homepage in the second week so that the students could familiarize themselves with the tool before the study.

The students were randomly divided into two groups: there were 32 ($N = 32$) students in the *treatment* group and 40 ($N = 40$) in the *control* group. At the beginning of the two hour session students first took a pre-test that lasted for 15 minutes. The pre-test included three program reading exercises where students were asked to write down the output of each program. After the pre-test, students rehearsed programming concepts presented in the test with a web-based tutorial for 45 minutes. The treatment group could also visualize the program code examples presented in the tutorial with the ViLLE tool. After the tutorial session students had 30 minutes to answer to a post-test. The post-test had the same three questions as the pre-test with two more demanding questions, in which the students had to write a program implementing a given task, and write down the output of a recursive program.

Pre- and post-test exercises were analyzed in the scale of zero to ten. Thus, the maximum points of the pre- and post-test were 30 and 50. Table 1 presents the pre-test scores for both groups, including point averages, standard deviations (in parentheses) and p-values calculated with a two-tailed t-test.

No statistically significant differences were found in any of the questions. In absolute scale the treatment group performed better in questions Q2 and Q3, and the control group in question Q1.

The scores from the post-test are presented in Table 2. Corresponding questions from the pre-test are shown in square brackets beside the post-test question names. Total points averages are presented both for the shared questions (questions that were the same in the pre- and post-test) and for all questions. Additionally, the point difference between the shared questions in the pre- and post-test is shown.

The comparison between the shared questions shows that the control group performed better in question PQ1 and the treatment group in questions PQ2 and PQ3. One reason for the smaller difference in the scores of the treatment group in PQ1 is the relatively high scores they got from the pre-test. In any case, the differences are so small that the null-hypothesis can not be rejected.

Table 3 displays a comparison between the pre- and post-test scores inside the groups.

As seen in the table, both groups performed statistically significantly better in the post-test than in the pre-test (p-value < 0.01). Based on this, it seems that it is possible

Table 1
Pre-test scores

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

Table 2
Post-test scores

Question	Control group ($N = 40$)	Treatment Group ($N = 32$)	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
Difference PQ1 – Q1	1.10 (2.60)	− 0.06 (2.81)	0.073
Difference PQ2 – Q2	2.40 (3.30)	3.38 (4.02)	0.262
Difference PQ4 – Q3	3.48 (4.81)	4.41 (4.53)	0.405
Total difference	6.98 (6.81)	7.72 (6.76)	0.646

Table 3
Pre- and post-test scores

Scores	Control Group	Treatment Group
Pre-test	10.58	10.41
Post-test	17.55	18.13
Total difference	6.98	7.72
p-value	0.000	0.000

to efficiently study basic programming concepts independently in these kinds of tutorial sessions.

The other research problem was the effect of earlier programming experience on learning results. When the results were analyzed, both groups were divided into two: students with no earlier programming experience (NPE) and students with some earlier programming experience (SPE). The pre-test scores with earlier programming experience taken into account are presented in Table 4.

The figures in the table show that students with some earlier programming experience got statistically significantly better scores in both groups. The corresponding results from the post-test are presented in Table 5.

As we can see from the table, statistically significant difference between NPE and SPE remained in the post-test in the control group. However, the difference vanished in the treatment group (p-values 0.212 and 0.151). Thus, it seems that ViLLE is especially useful for novice programmers. Because of the relatively short exposure to the tool, the result can be seen as significant. Cronbach alpha-values calculated for pre- and post-test questions (pre-test $\alpha = 0.667$ and post-test $\alpha = 0.831$) indicate high reliability. This study is presented in more detail in (Rajala *et al.*, 2008).

Table 4
The effect of earlier programming experience on pre-test scores

Question	Control Group			Treatment Group		
	NPE ($N = 23$)	SPE ($N = 17$)	p-value	NPE ($N = 20$)	SPE ($N = 12$)	p-value
Q1	4.17 (2.33)	6.59 (2.53)	0.003	5.60 (2.11)	7.17 (2.76)	0.107
Q2	1.22 (1.78)	4.71 (4.31)	0.005	1.00 (2.22)	4.00 (4.51)	0.049
Q3	1.00 (2.86)	4.94 (4.62)	0.005	1.65 (3.62)	2.83 (4.34)	0.414
Total	6.39 (4.68)	16.24 (9.63)	0.001	8.25 (5.44)	14.00 (8.48)	0.051

Table 5
The effect of earlier programming experience on post-test scores

Question	Control Group			Treatment Group		
	NPE ($N = 23$)	SPE ($N = 17$)	p-value	NPE ($N = 20$)	SPE ($N = 12$)	p-value
PQ1 [Q1]	5.74 (2.78)	7.06 (2.75)	0.144	5.90 (2.86)	6.50 (2.43)	0.533
PQ2 [Q2]	3.39 (3.97)	7.41 (3.81)	0.003	4.70 (4.58)	6.83 (4.22)	0.199
PQ3	5.30 (4.06)	7.59 (2.90)	0.045	5.05 (3.65)	7.25 (3.65)	0.109
PQ4 [Q3]	5.22 (4.83)	7.41 (3.94)	0.122	6.00 (4.71)	7.33 (3.94)	0.418
PQ5	6.09 (4.09)	8.35 (2.96)	0.049	6.05 (4.20)	7.75 (3.82)	0.261
Total (shared)	14.35 (8.27)	21.88 (8.51)	0.008	16.60 (9.29)	20.67 (7.64)	0.212
Total (all)	25.74 (14.44)	37.82 (13.68)	0.011	27.70 (15.49)	35.67 (13.53)	0.151
Difference PQ1 – Q1	1.57 (2.48)	0.47 (2.70)	0.198	0.30 (2.62)	− 0.67 (3.11)	0.354
Difference PQ2 – Q2	2.17 (3.07)	2.71 (3.65)	0.620	3.70 (4.38)	2.83 (3.46)	0.564
Difference PQ4 – Q3	4.22 (4.73)	2.47 (4.87)	0.261	4.35 (4.73)	4.50 (4.38)	0.929
Total difference	7.96 (5.80)	5.65 (7.98)	0.295	8.35 (7.98)	6.67 (4.14)	0.439

3.2. *The Effect of Engagement on Learning*

Naps *et al.* (2002) presented a taxonomy of learner engagement with a visualization tool. The taxonomy defines six levels of engagement:

1. *No Viewing*: There is no visualization tool in use.
2. *Viewing*: User follows the visualization passively. Despite of its name, all forms of observation belong to this level. The user can control the flow of the visualization but is not allowed to actively participate in any way.
3. *Responding*: User answers questions presented during the visualization.
4. *Changing*: User changes the visualization, for example by modifying the program code or algorithm used in the visualization.
5. *Constructing*: User actively participates in the construction of visualization, for example by writing program code.
6. *Presenting*: User presents the visualization and evaluates it together with the audience.

The study presented in Subsection 3.1 was extended so that a third group was formed which used ViLLE in a lower level of the engagement taxonomy. Hence, there were three groups: no viewing ($N = 40$), viewing ($N = 65$) and responding ($N = 32$). The questions were removed from the version of ViLLE used by the viewing group. The purpose of the setup was to confirm the hypothesis of Naps *et al.* (2002), which states that visualizations can have an effect on learning only if they are used in engagement level three or higher.

Statistical differences between the groups in the pre- and post-test were calculated with one-way ANOVA-test, and are presented in Table 6.

As seen in the table, there were no statistically significant differences between the groups. Students in the viewing group also improved their results statistically significantly during the session (the p-value of pre- and post-test difference inside the group < 0.01).

The differences in learning results between the novices and more experienced programmers inside the groups were examined next. The results are shown in Table 7.

There was a statistically significant difference in all groups between the NPE and SPE. As discovered in the previous section, the difference in the responding group vanished in the post-test. However, the difference remained in the viewing group (p-value < 0.001). Thus we can conclude that for the ViLLE to be useful, it should be used in an engagement level higher than viewing. To confirm the result, a post-hoc Student-Newman-Keuls test

Table 6

Statistical differences between the groups

	No viewing ($N = 40$)	Viewing ($N = 65$)	Responding ($N = 32$)	p-value
Pre-test total	10.58 (8.64)	10.85 (8.89)	10.41 (7.18)	0.968
Post-test total (shared)	17.55 (9.08)	17.94 (9.53)	18.13 (8.81)	0.963
Total difference	6.97 (6.81)	7.09 (6.63)	7.72 (6.76)	0.881

Table 7

The effect of previous programming experience on pre- and post-test score

	No viewing			Viewing			Responding		
	NPE ($N = 23$)	SPE ($N = 17$)	p-value	NPE ($N = 36$)	SPE ($N = 29$)	p-value	NPE ($N = 20$)	SPE ($N = 12$)	p-value
Pre-test total	6.39	16.24	*0.001*	6.81	15.86	*0.000*	8.25	14.00	*0.051*
Post-test total (shared)	14.35	21.88	*0.008*	13.72	23.17	*0.000*	16.60	20.67	*0.212*
Total difference	7.96	5.65	*0.320*	6.92	7.31	*0.812*	8.35	6.67	*0.439*
Post-test total (all)	25.74	37.82	*0.011*	24.72	39.90	*0.000*	27.70	35.67	*0.151*

Table 8
Pre-test scores divided into homogenous subsets

Group	*N*	Subset for alpha = 0.05	
		1	2
No Viewing NPE	23	6.39	
Viewing NPE	36	6.81	
Responding NPE	20	8.25	
Responding SPE	12		14.00
Viewing SPE	29		15.86
No Viewing SPE	17		16.24

Table 9
Post-test scores divided into homogenous subsets

Group	*N*	Subset for alpha = 0.05	
		1	2
Viewing NPE	36	13.72	
No viewing NPE	23	14.35	
Responding NPE	20	16.60	16.60
Responding SPE	12		20.67
No viewing SPE	17		21.88
Viewing SPE	29		23.17

was used. The analysis forms two homogenous subsets from the pre-test scores so that the novices and more experience programmers belong to different subsets (Table 8).

When the post-test scores were analyzed similarly, the results show that the novices in the responding group caught up all the SPE groups (Table 9).

The analysis confirms the previous finding that ViLLE is especially useful for novices learning programming basics, but only if used in higher levels of engagement. Similar learning results are not achieved if visualizations are viewed passively.

3.3. *The Impact of Prior Experience on Learning*

We also wanted to study what kind of effects prior experience on using a visualization tool has on the learning results. Presumably, students who have familiarized themselves with the usage of a tool can focus better on the subject taught because the cognitive load of using the tool is lighter. The study was organized in two instances of a high school programming course. The only difference between the courses was that in the latter course students were familiarized with the use of ViLLE and its features. A session similar to the ones presented in previous sections was arranged, although the questions in pre- and post-test were partly modified (for example recursion was thought to be concept

Table 10

Math and CS grades (scale 4...10)

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

Table 11

Pre- and post-test scores

	Pre-test total	Post-test total (shared)	Post-test total (all)
Control Group ($N = 17$)	7.12	12.59	16.94
Treatment Group ($N = 7$)	9.43	19.57	26.43
p-value	*0.515*	*0.047*	*0.046*

too complex for high school students). Students who hadn't used ViLLE before belonged to control group ($N = 17$) while students with prior experience on ViLLE formed the treatment group ($N = 7$). Though the groups were quite small, results were statistically significant.

To confirm the equality of the groups, participants' earlier math and CS grades were analyzed. The averages and standard deviations are presented in Table 10.

There were no statistically significant differences in grades between the groups. The absolute differences are less than one point.

Pre- and post-test scores are presented in Table 11.

There was no statistically significant difference between the groups in the pre-test scores, but there is a difference in the post-test. The statistically significant difference can be found both in the shared (same questions in pre- and post-test) questions and all questions. Based on the results we can conclude that prior knowledge of ViLLE clearly improves learning results. Furthermore the results support our earlier findings that ViLLE is useful in learning basic programming skills: both groups got statistically significantly better results from the post-test in comparison to the pre-test. The results are presented in more detail in (Laakso *et al.*, 2008b).

3.4. *Student Feedback*

In addition to quantitative tests we wanted to find out what students think about the tool. The opinions were gathered from students participating in a course called 'Introduction to Information Technology' at the University of Turku. ViLLE was an integral part of the course as all the assignments were done with it. 114 students answered to the questionnaire consisting of three parts: general questions about the tool, how useful the system is when learning new programming concepts, and opinions about the features of the tool.

In the first section six statements about system were presented, and students were asked to evaluate those in a scale of one to seven (1: completely disagree, 7: completely agree). Based on the answers, the students seem to think that the tool is quite suitable for teaching programming (average of all answers 5.64), that it is fairly easy to use (avg. 5.49) and that it helps in understanding the basic programming concepts (avg. 5.41).

In the second section the students were asked to evaluate the usefulness of ViLLE in different areas related to programming (see Table 12). Based on the answers, the students found ViLLE as an useful tool for teaching all the basic concepts – arrays were the only concept which got an average less than five (avg. 4.73). Based on students' comments, this is probably because of the usability issues in answering the array questions.

In the third section the students were asked to evaluate the usefulness of different features of ViLLE (see Table 13). All features – except the visualization of programs in different languages – got an average of five or higher. The features the students found most useful were the visualization of variable states and the automatic assessment of exercises (averages 5.90 and 5.80). The worse average in "visualization in multiple languages" is probably due to a fact, that the students didn't use the feature. For that matter, the feature can be found most useful when the students already know a programming language, and a different language is taught.

Table 12
Usefulness of ViLLE in understanding programming concepts

How useful did you find ViLLE in understanding the following concepts? (scale 1–7)	
Variables and assignments	5.41 (1.37)
Conditional statements	5.52 (1.15)
Loops	5.61 (1.19)
Boolean statements	5.38 (1.23)
Subprogram definitions	5.38 (1.25)
Subprogram calls	5.34 (1.32)
Subprogram parameters	5.24 (1.33)
Arrays	4.73 (1.58)

Table 13
Usefulness of ViLLE's individual features

How useful did you find the following features in ViLLE (scale 1–7)?	
Visualization of programs in different languages	4.93 (1.46)
Visualization of subprograms with call stack	5.35 (1.24)
Visualization of variable states	5.90 (1.17)
Explanation of program code line	5.40 (1.49)
Questions about program execution	5.50 (1.24)
Automatic assessment of exercises	5.80 (1.28)

Additionally, the students were asked to evaluate the number of ViLLE exercises in the course and the number of questions in the exercises in the scale of one to seven (1 – too few, 7 – too many). Based on the answers, the amounts were quite suitable (exercises avg. 4.48 and questions avg. 4.13).

Moreover, the students had a possibility to give additional comments about the tool. Most of the feedback was quite positive. However, some criticism and thoughts about improving the tool were also given. Some of the positive things mentioned were:

– "In my opinion, ViLLE exercises were more effective than lectures."
– "I found ViLLE really useful: because of the ViLLE exercises, I didn't practically need the lecture handouts at all to learn programming."
– "With ViLLE I was able to pick up programming far better than with lectures."
– "There were a lot of different exercises and it was easy to do them wherever and whenever I felt like it."

The negative comments were mostly related to the functionality of the user interface:

– "ViLLE is great when it functions properly; however, the GUI needs some improvement: e.g., handling arrays is illogical and difficult."
– "It would be handy if you could see the execution history when answering the questions:"
– "All the essential things won't fit on the screen simultaneously. Why do you need to login to ViLLE?"
– "It's irritating that you can't scroll the window when the question appears."

Moreover, some of the students would like to have more conventional teaching, instead or alongside ViLLE exercises:

– "I'd rather attend the traditional computer lab sessions, where there would be some kind of help available on request."
– "ViLLE was a good tool for studying; however, more training in small groups would be highly appreciated."
– "ViLLE is ok for learning the basics, but personally I learned more during the lectures."

All in all, the students seemed to have quite positive image about ViLLE's usefulness, though some improvements were also wished for. In conclusion, the students would prefer a course, where ViLLE exercises are integrated with more traditional teaching methods. The reported problems related to the user interface will be taken into account in the further development of the tool – e.g., the answering to array questions should nowadays work more logically than in the version, which the opinions were gathered about.

4. The Future of the Tool

ViLLE is nowadays in use in basic programming courses in most universities in Finland, and in addition, it will be mobilized at least in Australia during this spring. In future, we plan to develop the tool further based on the user opinions and experiences, and further research the effects of the tool when used in programming teaching. Our goal is to add more features to support the higher levels of engagement taxonomy. Moreover, exercise templates are developed to enable the randomization of the exercise parameters (i.e., variable types and values etc.); this makes redoing the exercises more meaningful. Additionally, different kinds of exercise types are under development, including code sorting exercises, where program code lines are randomly shuffled and the students are supposed to sort them in an order which implements the given task or algorithm.

5. Conclusions

Based on the presented results, experiences and feedback we can conclude the following:

- By using ViLLE the novices can improve their learning results significantly, even when the tool is used quite briefly...
- ... but only if the tool is used in the higher level of engagement. The mere viewing of visualization doesn't seem to have the same effect. This supports the hypothesis presented by Naps *et al.* (2002).
- Moreover, to ensure the learning results, the students must be familiarized with the tool beforehand, hence reducing the cognitive load of learning to use the tool.
- Most of the students think that ViLLE is beneficial when learning the basic programming skills. However, some of the students seem to think that the best way to use such tool is to integrate it with the more conventional forms of teaching.
- ViLLE gives students a chance to learn and practice the very basics of programming independently. These basics can't normally be covered on the lectures as thoroughly as required by some students.
- The focus of ViLLE is in program code reading and comprehension skills. However, Lopez *et al.* (2008) state that the tracing skills correlate with students performance on code writing tasks.
- From a teacher's point of view, ViLLE's key feature is the flexibility, and most importantly the support for almost any imperative programming language.

All in all the experiences seem quite encouraging so far: the results and the students' feedback both confirm the assumption that ViLLE can be used effectively in teaching the basic programming skills to novices. As we all know – unless we have already forgotten – the first steps in programming really are quite difficult. Therefore there seems to be a demand for such system, both now and in the future.

References

Ala-Mutka, K. (2005). Ohjelmoinnin opetuksen ongelmia ja ratkaisuja. In *Tekniikan opetuksen symposium*, 20–21.10.2005. Helsinki University of Technology.
`http://www.dipoli.tkk.fi/ok/p/reflektori/verkkojulkaisu/index.php?p=verkkojulkaisu`

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.

Laakso, M.-J., Myller, N. and Korhonen, A. (2008a). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. *Journal of Educational Technology and Society* (to appear).

Laakso, M.-J., Rajala, T., Kaila, E. and Salakoski, T. (2008b). The impact of prior experience in using a visualization tool on learning to program. In *Proceedings of CELDA 2008*, Freiburg, Germany.

Lopez, M., Whalley, J., Robbins, P. and Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. In *Proceeding of the Fourth International Workshop on Computing Education Research*, September 6–7, 2008, Sydney, Australia, 101–112.

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.

McCracken, M., Almstrum, V., Diaz, D., Gudzial, 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.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. and Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, **35**(2), 131–152.

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**, 15–32.

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

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

T. 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.

M.-J. Laakso is currently working as a researcher at University of Turku. He received his MSc (computer science) in 2003. His research interest covers program and algorithm visualization, learning environments, computer aided and automatic assessment in computer science education.

T. Salakoski is a professor of computer science at University of Turku, where he received his PhD 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.

Programų vizualizavimo priemonė: poveikis mokymui, patirtis ir studentų atsiliepimai

Erkki KAILA, Teemu RAJALA, Mikko-Jussi LAAKSO, Tapio SALAKOSKI

Laikoma, jog pradedančiuosius mokant programavimo pagrindų naudinga taikyti programų vizualizavimo metodą. Tačiau tėra labai nedaug tyrimų apie šio metodo poveikį. Turku universitete buvo sukurta programų vizualizavimo priemonė, pavadinta ViLLE. Straipsnyje supažindinama su šios priemonės poveikio tyrimais, pateikiami nauji kokybiniai duomenys: studentų, naudojusių šią priemonę, atsiliepimai. Ir mūsų tyrimų rezultatai, ir atsiliepimai rodo, kad programos vizualizavimo priemonė ViLLE gali būti veiksmingai naudojama pradedančiuosius programuotojus mokant pagrindinių programavimo sąvokų.