Breaking the Programming Language Barrier

Using Program Visualizations to Transfer Programming Knowledge in One Programming Language to Another

Johannes Holvitie, Teemu Rajala, Riku Haavisto, Erkki Kaila, Mikko-Jussi Laakso and Tapio Salakoski
Department of Information Technology
University of Turku
Turku, Finland

{jjholv,temira,rahaav,ertaka,milaak,sala}@utu.fi

The transition from one programming language to another is an issue, which usually needs to be addressed in programming curricula, as the learning is typically started with syntactically easier languages. This study explores the possibility to use a short interactive tutorial with visualization exercises to ease the transition from Python to Java.

In the experiment, the students first took a pre-test to measure their earlier programming knowledge with Python. After that, they used the tutorial with visualization exercises for 45 minutes. The tutorial and the exercises were designed to underline the syntactical and structural differences between Python and Java. Finally, the students answered to post-test, which contained questions similar to pre-test, but in Java.

The results indicate, that the students were able to obtain similar program comprehension skills in Java that they previously had with Python. Moreover, the students seem to think that using such tutorials is highly beneficial in the transition. Hence, we conclude, that ViLLE can be effectively used to ease the transition from one language to another.

Keywords: program visualization, language transition, programming education

I. INTRODUCTION

The decision of the first programming language taught is not easy: the pressure to use commercially successful languages is high, mainly because of the industry support and demands. The problem with industrial strength languages is that they tend to be overly complex and usually carry a large syntactical baggage of definitions and structures that are required for them to fulfill the standards in software development.

However, there are easier languages to start with, languages designed with aesthetics and ease of readability in mind. Pseudo languages and Python, for example, are examples of languages where the confusing syntactical features have been hidden by replacing them with structures that are easier to comprehend, even for novices. Table 1 holds an example of simple 'Hello world' program written in both, Java and Python. The Python version doesn't contain the syntactical baggage of Java (including class and method definitions, accessibility keywords and semicolons).

Even if an easier-to-comprehend language is used in the introductory courses, the transition to more complex industry standard languages awaits. Understanding and being able to use commercially successful language(s) is essential in any programming curriculum: the students need to perceive and understand the intrigue details that can be used as an advantage to gain the most performance and control over the execution, and to properly understand all the underlying principles of the chosen implementation method and technique.

The transition from one language to another should be made as effortless as possible. Providing information and examples about the similarities and differences in core constructs, as well as about the logical differences in syntax, the gradual transition could be made easily. In this paper, we present an experiment, where the students are introduced to a new language by using a short intensive tutorial with visual and interactive programming exercises.

II. RELATED WORK

Studies on moving from one programming language to another are scarce. However, several studies provide valuable information on choosing a programming language

TABLE I. HELLO WORLD PROGRAM IN JAVA AND PYTHON

Java	Python

```
public class HelloWorld{
  public static void main(String[] args){
    System.out.println("Hello world!");
  }
}
```
	`print "Hello world!"`

for the purposes of teaching various programming concepts. However, the inevitable transition from one language to another is rarely discussed.

The effect of first programming language on later "programming career" has been studied by McIver [2] and Wexelblat [3]. They conclude that programming style, coding technique and program code quality are affected by this choice, and that the decision is one that should not be made lightly.

McIver & Conway [4] discuss this decision of choosing the first language by studying the characteristics of introductory programming languages. They present three sets of seven criteria containing the observed pedagogical problems of studied languages, design principles for creating introductory languages and criteria for evaluating existing languages for teaching introductory programming.

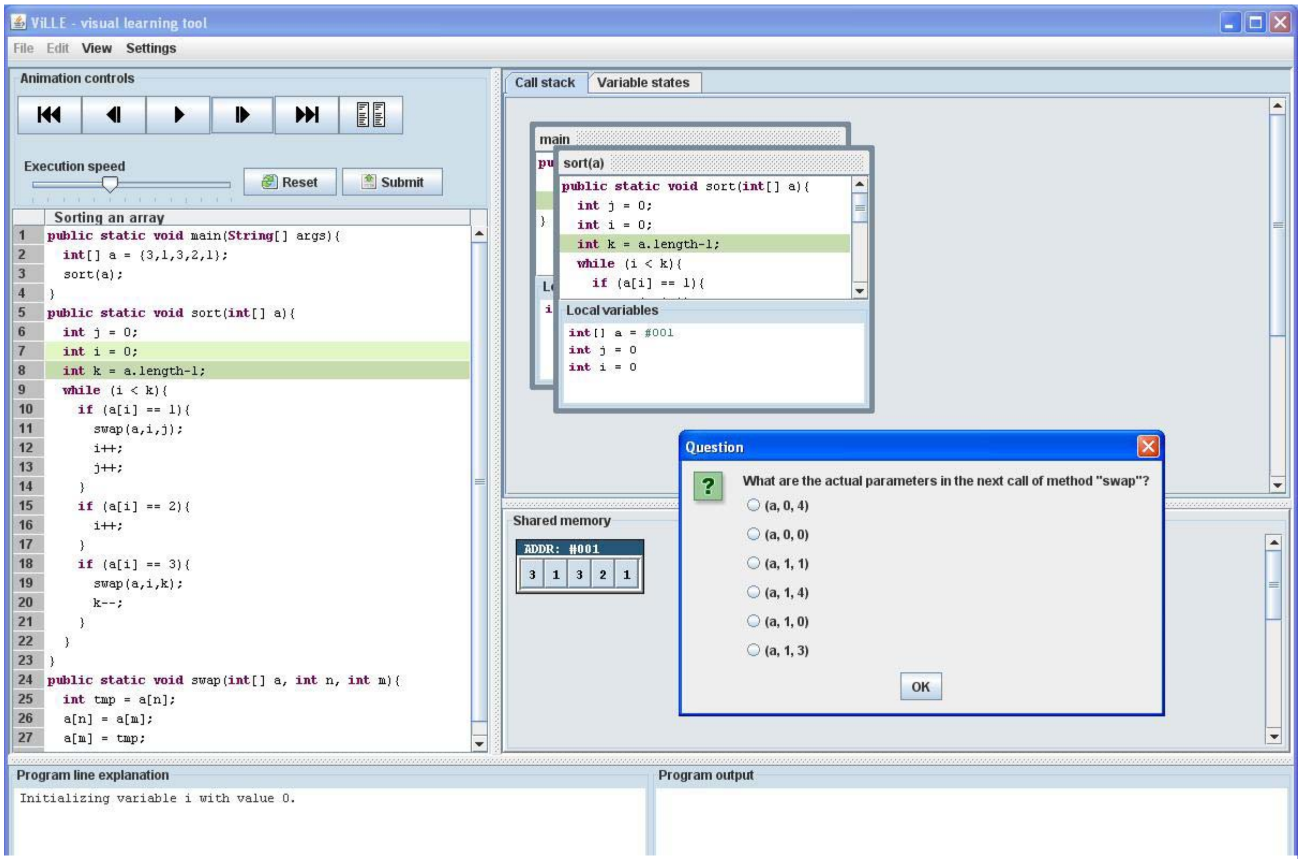
Java and Python are commercially popular languages often used in teaching. Grandell et al. [5] found Python to be better suited in introductory programming than Java, due to its lighter syntactical baggage. The use of Python allowed them to teach more advanced topics in the introductory course. Additionally, the comparison made by Laakso et al. [1] between Java and a pseudo-language (a subset of Python) showed that the syntactic weight of Java is a disadvantage in the first stages of learning programming.

However, selecting an easily understandable programming language may have its downsides. Oldham [6] reports a study where Python was adopted in college introductory level programming teaching. Python is an easily understandable language and it primarily follows the Object-Orientation (OO) paradigm. Though it is generally considered excellent as an introductory language, its usage is limited. Python's narrow set of abilities to restrict data access make teaching data hiding – a core feature of OO – a hard task and require the usage of get- and set-methods to demonstrate how for example the class invariant is maintained.

Lastly, Miller & Ranum [7] report a study where an introductory programming course utilizing Python was followed with an advanced course in Java. They observed that the previous knowledge in Python was clearly beneficial when several additional OO features enforced by Java were introduced. They call it the "build on what you know principal": The more intricate subjects of Java can be taught more easily due to students being familiar with most programming concepts in Python.

III. ViLLE

In this section the ViLLE tool is presented. It was used in the study to provide the visual and interactive tutorial for the students as well as to gather data. Furthermore, some features of ViLLE affecting its selection as the platform for



```
public static void main(String[] args){
  int[] a = {3,1,3,2,1};
  sort(a);
}
public static void sort(int[] a){
  int j = 0;
  int i = 0;
  int k = a.length-1;
  while (i < k){
    if (a[i] == 1){
      swap(a,i,j);
      i++;
      j++;
    }
    if (a[i] == 2){
      i++;
    }
    if (a[i] == 3){
      swap(a,i,k);
      k--;
    }
  }
}
public static void swap(int[] a, int n, int m){
  int tmp = a[n];
  a[n] = a[m];
  a[m] = tmp;
```

```
int[] a = #001
int j = 0
int i = 0
```

```
ADDR: #001
3 1 3 2 1
```

What are the actual parameters in the next call of method "swap"?

- (a, 0, 4)
- (a, 0, 0)
- (a, 1, 1)
- (a, 1, 4)
- (a, 1, 0)
- (a, 1, 3)

```
Initializing variable i with value 0.
```

Figure 1. The visualization view in ViLLE

the tutorial are presented.

ViLLE is a collaborative education tool (see Figure 1) developed at the University of Turku. Amongst other abilities, ViLLE can visualize program execution in various commercially popular languages or self-defined pseudo syntaxes. The visualizations include interactive components engaging the students into the task at hand. The tool was used in implementing the tutorial due to its features being designed with language-independency in mind, something that we can presume to be very valuable in aiding the transition process. The most important features of ViLLE, essential to building the tutorial, are described shortly.

Language-independency. ViLLE provides the ability to view the programming examples and exercises in several different programming languages and allows the user to fluently switch between them. The user can concretely observe the similarities between different languages.

The parallel view. The user is able to simultaneously observe the execution in two different languages and to see how similar the program flow is between languages of same programming paradigm.

Flexible visualization controls. The execution can be viewed step-by-step both forwards and backwards, or continuously with adjustable speed.

In addition to the features presumably helping the transition between programming languages, ViLLE provides the user with extensive amount of information during the execution including the states of the variables, explanation about the program line under execution, and visualizations of subprogram calls and data structures. For content creators ViLLE provides editors for defining new pseudo syntaxes as well as a variety of other types of automatically assessed exercises.

Further information about ViLLE's features can be found at http://ville.cs.utu.fi.

IV. Previous work

We have previously researched the usage of ViLLE in various studies. This section presents some of the most important results.

ViLLE's effectiveness was studied in "Introduction to Information Technology" course in University of Turku. 72 attendees were randomly divided into two groups. Both groups started the session by answering to pre-test, which measured their earlier experience and knowledge in programming. After that, the students used a programming tutorial independently for 45 minutes. The treatment group (N=32) could visualize the examples in tutorial with ViLLE, while the control group (N = 40) used only text based material. Finally, all students answered to post-test. The results showed no statistically significant differences between groups; however, when the earlier programming experience was taken into account, we found out that the statistically significant difference between novices and experienced students disappeared in the treatment group, while the difference in control group remained. Hence, we concluded that ViLLE is especially useful for novice programmers. The study is presented in detail in Rajala et al. [8].

The study was later extended to contain a third group so that each group was in different level of engagement (see Naps et al. [9]). The third group (N=62) could visualize the execution of example programs with ViLLE, but the visualization contained no interactive elements (such as questions about visualization). The results showed that the statistically significant difference between novices and experts remained in the third group as well. This confirms the hypothesis, that visualization tools are only useful if used in higher levels of engagement. The extended study is presented in Kaila et al. [11].

We have also studied e.g. the effects of cognitive load of learning to use the visualization tool (see Laakso et al. [12]), using the visualization tool collaboratively (Rajala et al. [13]) and the course long use of such tools (Kaila et al. [10]). In addition to all the quantitative studies, we have also conducted qualitative research about the tool: 114 students answered the survey after collectively using ViLLE to answer to more than 10,000 exercises. The answers showed that most of the students think that ViLLE is highly beneficial for their learning; some of them even preferred ViLLE over traditional learning methods, such as lectures or demonstrations. The study is presented in detail in Kaila et al. [14].

In addition to all the previous studies, we wanted to find out whether ViLLE can be used to help the transition from one programming language to another. The syntactical similarities should be easy to visualize, and as ViLLE supports a variety of programming languages, it should be useful in emphasizing and building links between the similarities of the languages. The study and the results are reported in the following sections.

V. Research

The purpose of this study is to find out how effectively students' knowledge of basic programming concepts in one programming language can be transferred to another language with a short programming session utilizing visualization exercises.

A. Method

This study was experimental design in which the treatment was identical to all participants. The only difference was that the pre- and post-test were in different programming language; python and java, respectively. The goal was to transfer student's existing knowledge from one programming language to another.

B. Materials

At the beginning of the study participants answered to a Python pre-test. This was composed of four program tracing exercises, a structure previously used in the collaboration study by Rajala et al. [13], covering conditional statements, loop structures and methods. Additionally, the students were asked about their previous programming experience in Java.

In the tutorial held between the pre- and post-test the similarities and differences between different programming concepts in Python and Java were briefly explained. After each explanation of a concept the tutorial included links to one or more visualization exercises in ViLLE covering the explained topic. The programming concepts covered in the tutorial were: variables, output, conditional statements, while- and for-loops, and procedures and functions. The students viewed the interactive exercises in parallel view mode by default to help them to grasp the similarities of the two programming languages.

After the tutorial, students answered to the post-test. The test had the same structure as the pre-test, only this time the programming language used was Java. Additionally, the students were asked if they had found ViLLE useful in transferring their knowledge from Python to Java, and how useful they found it in learning the various concepts introduced in the tutorial.

C. *Participants*

The participants were students who attended the course held in the high school of Kupittaa, a specialized institute having a focus on teaching information technology and media. This instance of the course was held in the spring semester of 2010, and according to the curriculum it aimed at enhancing students' programming skills, taking them from structured programming into the direction of object-orientation [15].

Following the suggested completion order in the curriculum, this course is participants' third or fourth programming course (first one in Java). Students attending this course are on their second or third year of studies. Due to the courses more advanced contents for high school level, the number of participants was only 10. A small sample size as this isn't ideal for providing statistical data, but it serves well as an introduction to the research setup, that we intend to reimplement with larger group of participants.

D. *Procedure*

The students had previously taken an introductory programming course in Python. Due to this course's focus in the OO paradigm and Python's impediments in completely fulfilling all the principles behind it [5][6], the course was held in Java. In order to transfer their current programming knowledge in Python to Java a two hour programming session was arranged. The session was held during the first lecture of the course.

At the beginning of the session, participants answered to a Python pre-test measuring their current programming skills. The students were given 15 minutes to complete the first phase. In the second phase, the students were introduced to the similarities and differences between the basic constructs and syntax in Python and Java with a 45 minute tutorial. Finally, in the last phase students had 30 minutes time to answer to a post-test. As previously described in the materials section, the contents and the structure of this test were similar to the pre-test, only this time the programming language was Java. Each exercise was graded on scale from zero to ten, totaling 40 as the maximum score for both tests.

VI. RESULTS

In this section we present the results of the study in order to find out an answer to the research question "Can ViLLE be used to transfer the basic knowledge attained in one programming language into another, via a short interactive tutorial?"

The pre-test results are presented in Table 2. Mean and standard deviation are presented for each question and the total of all questions.

TABLE II. PRE-TEST SCORES

Question	Mean (N=10)	Std. Dev. (N=10)
Q1	8.00	3.77
Q2	1.50	3.38
Q3	6.00	4.52
Q4	7.20	3.68
Total	22.70	9.95

The results for the post-test are presented in Table 3. The post-test followed the structure of the pre-test, the only difference being the programming language used. The question numbers correspond to similar question in the pre-test. Mean and standard deviation for all questions, as well as the total of all questions are presented.

TABLE III. POST-TEST SCORES

Question	Mean (N=10)	Std. Dev. (N=10)
Q1	9.00	2.11
Q2	2.00	4.22
Q3	5.20	5.10
Q4	7.10	3.81
Total	23.30	11.03

To find out whether there is a difference in pre- and post-test results, a pairwise t-test was used between corresponding questions and the total values. The results are presented in Table 4.

TABLE IV. PAIRED SAMPLES TEST BETWEEN THE PRE- AND POST-TEST

Question	Pre mean (N=10)	Post mean (N=10)	p-value
Q1	8.00	9.00	0.42
Q2	1.50	2.00	0.34
Q3	6.00	5.20	0.22
Q4	7.20	7.10	0.96
Total	22.70	23.30	0.80

As seen in the table, no statistically significant differences can be found between individual questions or in the total of all questions.

During the post-test, the students were also asked whether they found ViLLE useful in transition between programming languages in general, and concerning given programming topics. The later questions were answered in scale of 1 to 5, where 1 represents not useful at all, and 5 very useful. The results of the survey are presented in Table 5.

TABLE V. USEFULNESS OF VILLE IN TRANSITIONING BETWEEN LANGUAGES

Concept	Mean (N=10)	Std. Dev. (N=10)
Variables & Output	4.20	0.42
Conditional statements	4.00	0.47
While-loops	3.90	0.57
For-loops	3.80	0.63
Methods	3.90	0.88

As seen in Table 5, the students found ViLLE very useful when transitioning from one language to another. Moreover, the binary question about ViLLE's usefulness in general was unanimously answered with positive opinion.

Cronbach's Alpha was used to ensure the reliability of variables in the procedure. The values calculated for pre-test (0.530) and post-test (0.645) indicate medium to high reliability.

VII. DISCUSSION

Reviewing the results reveals that there are no statistically significant differences between the scores in pre- and post-test. This indicates that the students' ability to effectively solve basic program tracing exercises remained at the same level, even though the questions in the post-test were presented with syntactically heavier programming language. It's notable, that the students had no previous experience with Java.

The observed effect takes place in all tested areas of programming. Thus, it seems that the transition from one programming language to another is possible with a short, interactive tutorial, as long as the tutorial and especially the exercises are designed to underline the syntactical differences and structural similarities of the languages. Hence, the answer to our research question is positive. Remarkably, the students only had 45 minutes to use the tutorial with ViLLE exercises. Hence, it seems that with the right tools, the transition can be made in relatively short time.

Students' opinions seem to confirm the effect of ViLLE: the binary answers about the usefulness in general were unanimous and positive. Moreover, ratings given for usefulness of the tool in specific areas were relatively high (mean >3.8 in scale of 1 to 5). This correlates with our previous findings: students seem to think, that ViLLE is highly beneficial for their learning (Kaila et al. [10]).

VIII. CONCLUSION

We conducted a study about the possibility to use interactive tutorial with visualization exercises to ease the transition from one programming language to another. The results indicate that by using such interactive tutorial – even for relatively short time – the students can build program comprehension skills for syntactically much heavier language. This supports the findings in Laakso [16] that automatic assessment with immediate feedback can be used to enhance learning of programming concepts. It is much more important to understand the ideas behind those concepts than to learn the syntactical notation. In addition, this study supports and extends our previous findings: ViLLE can be used effectively to teach basic programming concepts.

REFERENCES

[1] Laakso, M-J., Kaila, E., Rajala, T. & Salakoski, T. 2008. Define and Visualize Your First Programming Language. In Proceedings of ICALT 2008 - the 8th IEEE International Conference on Advanced Learning Technologies. July 1st - July 5th, 2008. Santander, Cantabria, Spain.

[2] McIver, L. The Effect of Programming Language on Error Rates of Novice Programmers. School of Computer Science, Monash University.

[3] Wexelblat, R. L. "The Consequence of One's First Programming Language." In Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems. Palo Alto, California, 1980. pp. 52-55.

[4] McIver L. & Conway D. 2011. Seven Deadly Sins of Introductory Programming Language Design. Department of Computer Science, Monash University. Available online: http://www.csse.monash.edu.au/~damian/papers/PDF/SevenDeadlySins.pdf. Fetched: 25.11.2011.

[5] Grandell et. al. 2006. Why Complicate Things? Introducing Programming in High School Using Python. Eighth Australasian Computing Education Conference (ACE2006).

[6] Oldham, J. 2005. What happens after Python in CS1?, Consortium for Computing Sciences in Colleges. Centre College. Danville, KY.

[7] Miller B. & Ranum D. 2006. Freedom to Succeed: a Three Course Introductory Sequence Using Python and Java. Consortium for Computing Sciences in Colleges. Luther College, Decorah IA.

[8] 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.

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

[10] Kaila, E., Rajala, T., Laakso, M.-J. & Salakoski, T. 2010. Long-term Effects of Program Visualization. In 12th Australasian Computing Education Conference (ACE 2010), January 18- 22, 2010, Brisbane, Australia.

[11] Kaila, E., Laakso, M.-J., Rajala, T. & Salakoski, T. 2009. Evaluation of Learner Engagement in Program Visualization. Appeared in 12th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2009), November 22 – 24, 2009, St. Thomas, US Virgin Islands.

[12] 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.

[13] Rajala, T., Kaila, E., Laakso, M.-J. & Salakoski, T. 2009. Effects of Collaboration in Program Visualization. Technology Enhanced Learning Conference 2009 (TELearn 2009), Taipei, Taiwan.

[14] 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.

[15] Kupittaa ICT high school curriculum (201X). Available online: http://www05.turku.fi/ah/ol/2010/1117018x/Images/1004679.pdf. Fetched: 28-11-2011.

[16] Laakso, M.-J. (2010). Promoting Programming Learning. Engagement, Automatic Assessment with Immediate Feedback in Visualizations. TUCS Dissertations no 131