Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti 1
Tampere University of Technology. Department of Pervasive Computing. Report 1

Essi Isohanni & Maria Knobelsdorf

Students' Engagement with the Visualization Tool VIP in Light of Activity Theory

TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti 1
Tampere University of Technology. Department of Pervasive Computing. Report 1

Essi Isohanni & Maria Knobelsdorf

Students' Engagement with the Visualization Tool VIP in Light of Activity Theory

Tampere University of Technology. Department of Pervasive Computing
Tampere 2013

ISBN 978-952-15-3045-6
ISSN 2323-9174

Students' Engagement with the Visualization Tool VIP in Light of Activity Theory

Essi Isohanni

Department of Pervasive Computing, Tampere University of Technology

essi.isohanni@tut.fi

Maria Knobelsdorf

Institut für Informatik, Universität Potsdam, Germany

knobelsdorf@cs.uni-potsdam.de

This article takes the first steps towards building a theory to explain how students interact with program visualizations when learning programming. First, we present the findings of a previous study we conducted to investigate how students voluntarily and regularly engaged with the program visualization tool VIP in a three-month programming course. Then, we interpret the empirical results of the study using Activity Theory. Finally, based on the interpretation, we propose two research hypotheses about students' long-term engagement with VIP. These hypotheses also set guidelines for future research on visualizations and teaching with visualizations, but most importantly, they offer a sustainable basis for further work on the theorization of students' interactions with visualizations.

Keywords: program visualization; activity theory; student engagement; qualitative research; grounded theory

1 Introduction

Execution of computer programs can be visualized in a simple manner by drawings or even using arts and crafts materials like reported in an experiment by Hundhausen (2002). However, as programming teachers have the skills to automatize this procedure a lot of effort has been put on technical development of software on this purpose. In the past 30 years, many different visualization tools were developed to illustrate program execution graphically with the purpose of aiding students to learn programming, e.g. Jeliot 3 (Ben-Bassat Levy et al., 2003) and Ville (Rajala et al., 2007). Nowadays, these tools represent an important field of research in Computer Science Education.

To date, the research on the educational effectiveness of program visualization tools has been "markedly mixed" as stated by Hundhausen et al. (2002). They also report that visualizations are not used widely even in cases where their educational effectiveness has been empirically proven. These problems have been addressed in many ways: there are studies mapping the problems of using visualizations in teaching, usability studies seeking to improve the quality of the visualizations, and studies on how students use the visualizations. Despite this work, what is missing is a theory to explain how students interact with the visualizations when learning programming. Creating such theories is important because ”[i]t is the long-term goal of much of science to generate theories that can offer stable explanations for phenomena that generalize beyond the particular” as stated by the Committee on Scientific Principles for Education Research in the rationale of their scientific principle ”link research to relevant theory” (Shavelson and Towne, 2002).

In addition to allowing us to generalize our results, a theory explaining the interactions of students with visualizations can also possibly provide us help in understanding the “markedly mixed” results of the previous visualization studies, explain why visualizations are not used despite their empirically proved educational effectiveness, and, above all, act as a research frame-work that makes future studies on program visualizations more easily comparable to each other.

This article incorporates theory in two different ways: 1) The aim of the article is to take a first step towards building a theory that explains students’ interactions with program visualization tools. 2) We start this theory building by interpreting the results of our empirical study using Activity Theory (AT) (Kaptelinin and Nardi, 2006) as a research framework. AT is a psychological theory proposing concepts that describe how a human being interacts with the environment using artefacts or social entities as mediators for the learning process involved. From this perspective, visualization tools can be understood as tools that mediate a student’s learning process with the proposed visualization and therefore change the programming activities involved.

The structure of this paper is as follows: we first present a previous empirical study on how students use the program visualization tool VIP. Section 2 presents the background and context of the study and Section 3 the study setup. Results of the study are presented in Sections 4–6. First, in Section 4, we present students’ working patterns when they use VIP, then in Section 5 we describe the long-term engagement with VIP, and last, in Section 6, we exemplify this all by telling the stories of two students who used VIP during a programming course. In Section 7, we interpret the results using AT. Based on the interpretation, we propose two research hypotheses that form an understanding of the use of program visualizations in learning programming in general. The hypotheses are a starting point for developing a theory of how students interact with visualizations when learning programming. The paper concludes ideas for future work in Section 8.

2 Background and Context of the Study

In this section, we present the background and context of our qualitative empirical research where we studied how students engaged with the program visualization tool Visual InterPreter (VIP) (Virtanen et al., 2005). This study was a result of several other studies about VIP conducted at Tampere University of Technology (TUT), which we will introduce in the next subsection providing our scope of research interest. In Section 2.2, we will present related work in this research field. Finally, in Section 2.3, we introduce the chosen research design and the research questions we intended to answer with our study.

2.1 Previous Studies on Student Engagement with VIP

According to Hundhausen et al. (2002), visualizations in general have not succeeded in becoming a part of mainstream programming education. This was also the case at TUT where VIP was developed and used to better support the students’ learning progress.

The tool had been integrated into the materials of an introductory programming course, and the students were left to choose themselves whether they use the tool for their weekly course assignments (Lahtinen, 2006). The majority of students in the course did not use VIP regularly, but we observed that there was a constant minority of students who used the tool regularly and enthusiastically by choice (Lahtinen et al., 2007a).

Testing whether VIP is beneficial for learning, a controlled experiment with a test group and a control group was designed, and this experiment showed that the students who used VIP for their course assignments spent more time preparing for the exercise session and also showed better learning results than those who did not work with the tool. Although the results were statistically significant, it was not possible to conclude whether the better learning results originated from the pedagogical impact of VIP's visualizations or from the fact that the visualizations made the students study for a longer time (Ahoniemi and Lahtinen, 2006). In addition, subsequent surveys revealed that the most frequent users of VIP were also the mid-level performing students and not those who have the greatest difficulties in learning (Lahtinen et al., 2007b). It was also revealed that the students who used VIP were less likely to drop out than students who did not use it (Lahtinen et al., 2007a).

The described results indicated that VIP indeed seems to be supportive and beneficial for learning, but it remained unclear why the majority of students were not using VIP despite the fact that it was promoted by the teacher and why, at the same time, a minority of students were constantly using the tool. We wondered how this minority of students become engaged with the tool on their own and also in the long term. At this point of our investigation, our theoretical understanding of learning programming was determined by constructivism, which we extended later by Activity Theory (see Sec. 7.1). According to constructivist learning theories, learning means that students construct their own individual understanding by building upon prior knowledge and skills (Illeris, 2002; Ben-Ari, 2001). From this point of view, learning is a highly active, self-directed, and meaningful process. Therefore, we assumed that students' engagement by choice with VIP must have been beneficial for them in some way, otherwise they would not have used it voluntarily. Furthermore, in the learning process a program visualization tool represents an entity that students can individually incorporate into their knowledge construction of programming concepts and skills. In consequence, understanding the educational effectiveness of a tool like VIP allows us to begin to understand how students become engaged with VIP in the long term and how they seem to benefit from this engagement. In the interaction between student and visualization tool, our focus lies on the students and how they incorporate VIP in their regular learning activities. We regard this as complementing our understanding of why other students do not become engaged with VIP. Thus, it seems sensible for us to investigate this issue further in order to develop a general understanding and knowledge background about student engagement with visualizations.

2.2 Related Work

In the previous section, we introduced and discussed our research interest, which focuses on students' engagement with visualizations. With the term *engagement*, we understand all kinds of activities students perform voluntarily with the visualization tool being mostly motivated by their own beliefs that the activity with the tool is beneficial for them. A *long-term* engagement includes for us a longer process of weeks and months as it corresponds to the duration of a typical programming course, which is contrary to the notion of a single-session that lasts for a maximum of few hours.

The research field of visualizations particularly emphasizes student engagement. This research focus goes back to a meta-study by Hundhausen et al. (2002) comprising 24 studies on visualizations and their educational effectiveness. The main result of this meta-study was the insight that students' utilization patterns of visualization tools have a much greater impact on their learning success—and therefore, on the tool's educational effectiveness—than the quality of the visualizations. As a consequence, Hundhausen et al. suggested investigating the educational effectiveness of visualizations focusing specifically on student engagement.

Following the meta-study (Hundhausen et al., 2002), Naps et al. (2003) explored the role of visualizations and the corresponding student engagement in CSE and proposed an Engagement Taxonomy (ET) with a general research framework for further inquiry. The ET defines different levels of engagement, for instance, the level *responding* means answering questions concerning the visualization presented by the system; meanwhile, the level *viewing* describes non-active involvement. The different engagement levels describe single situations or activities. Their research framework proposes hypotheses contrasting these engagement levels (e.g. "*Responding* results in significantly better learning outcomes than *viewing*"), indicates these hypotheses can be tested, and recommends a classical experimental study that is based on three steps: pre-test, use of materials, and post-test. A great number of controlled studies have been conducted following this research framework; to summarize them Urquiza-Fuentes and Velázquez-Iturbide (2009, Sec. 3) presented a review where they analyze 33 evaluation studies of VTs with regard to the different levels of the ET. All these studies emerge from the general question how to better engage students with visualizations testing different pedagogical approaches with regard to the ET levels.

Beside the ET, other kinds of studies have been conducted to investigate student engagement with visualizations. We will summarize these studies below:

- Sorva (2012) studied students' perceptions on using the visualization tool UUhistle in learning programming and reports six qualitatively different understandings. The category describing the richest understanding of learning through the visualizations explained how students learned about program execution using the tool as a learning aid. The categories describing simpler understandings showed that the students perceive the visualization tool, for example, mainly as a source for copying and pasting code from example programs or as a tool where you manipulate graphical presentations, or a tool for learning what a computer does.
- The visualization tool Jeliot has been studied intensively for over a decade. An overview of this work is provided by Ben-Ari et al. (2011) in a summary report. We briefly summarize those Jeliot studies that are closest to our research interest:
    - Moreno and Joy (2006) followed the use of the tool Jeliot to verify two research hypotheses related to students' expectations for Jeliot and how the visualizations are comprehended. In their study, Moreno and Joy found two categories describing the use of the tool: Jeliot as a learning aid and Jeliot as a debugger. They reported that the students were using Jeliot mainly as a debugger rather than making an effort to understand the animations. This result indicates that students indeed construct their individual understanding of how a tool like Jeliot might support their learning activities.

    - Lattu et al. (2000) used both interviews and observational methods to investigate students' experiences using Jeliot. The authors reported a list of different kinds of purposes students developed to use Jeliot that they discovered in the field. The authors' emphasized testing the applicability of Jeliot and how students responded to it. Their objective was to record the drawbacks of the tool in order to improve further tool development.
    - Kannusmäki et al. (2004) used students' written reflections, discussion board messages, and feedback emails to study how students used Jeliot and what features they would like to see included in the tool. The authors reported four different patterns of completing programming exercises. Jeliot was used in two of these patterns. Like Lattu et al. (2000), they also concentrated on the usability of Jeliot.

  This work on Jeliot gives interesting insights into students' use of visualizations and hints to how they might benefit from using them. However, none of these studies focused particularly on how students engage with visualizations in terms of learning processes. These studies rather investigated the student perspective on the tool's usability and have only a minor attempt to capture and reconstruct students' learning processes and the specific activities involved with.
- Romero et al. (2005) studied the role visualizations played in debugging activities. The authors presented a detailed process analysis on how students used visualizations when debugging software in single use sessions. Their results covered different forms of students' debugging strategies and demonstrate how diverse and individual students' interaction with a visualization can be.
- Lönnberg et al. (2011) identified students' different approaches to debugging concurrent programs with the visualization tool Atropos. Just like in the previously mentioned study, they also concentrated on debugging activities instead of the student engagement of the tool in general, limiting the investigation scope also to single use sessions.

Similar studies describing how students use visualizations have been conducted for other visualization tools too (see, for example, Hundhausen and Brown, 2007, 2008; Kehoe et al., 1999).

The reported studies as well as the different engagement levels of the ET focus on single interaction situations. The engagement with a visualization tool is hardly considered to be part of a long-term process that goes on for weeks, months, or even years (Stasko and Hundhausen, 2004, pp. 224). Since learning programming is a long-term process, we assume that the benefits of a visualization tool depend rather on a long-term engagement than on a single use-situation. The studies that investigated students' long-term use of visualization tools, like Ben-Bassat Levy et al. (2003); Korhonen et al. (2002); Laakso et al. (2005), focused on students' performance and outcome after using the visualization tool. They report that the visualization tools they used had positive effects on students' learning; in the study of Laakso et al. (2005) especially for less talented students and in the study of Ben-Bassat Levy et al. (2003) for the mediocre students. In these studies, the authors investigated students' exam results statistically without considering the preceding engagement process. Thus, it is not possible to explain the exact reasons of why the visualizations were beneficial for these particular groups of students.

In summary, research in the field of student engagement has focused mostly on single-session interventions investigating either how to engage students with visualizations with regard to the ET or on usability and debugging aspects of a visualization tool. We found little evidence for possible explanations for why and how students voluntarily and regularly used visualizations on their own in the long-term and how they believe they benefited from this usage. As a consequence, we have conducted our own study where we investigated this issue further.

2.3 Research Design and Research Questions

When developing the research design of our study, we decided not to consider the ET. The ET was developed mostly normatively in order to describe possible engagement levels that can be tested in experiment studies. Our research interests, instead, focus on students that are already engaged in using VIP regularly and in the long-term. It would have been possible to investigate which level of the taxonomy our students' engagement corresponded to and to connect this to the hypotheses discussed by Naps et al. (2003). Though, approaching the field with predetermined classifications and hypotheses about possible forms of engagement which are only little grounded in students actual activities with a visualization tool can lead to the situation in which other possible forms of student engagement might be ignored or not investigated any further. Since we are particularly interested in all forms of student engagement with VIP, we did not consider the ET for us as a conceptual framework.

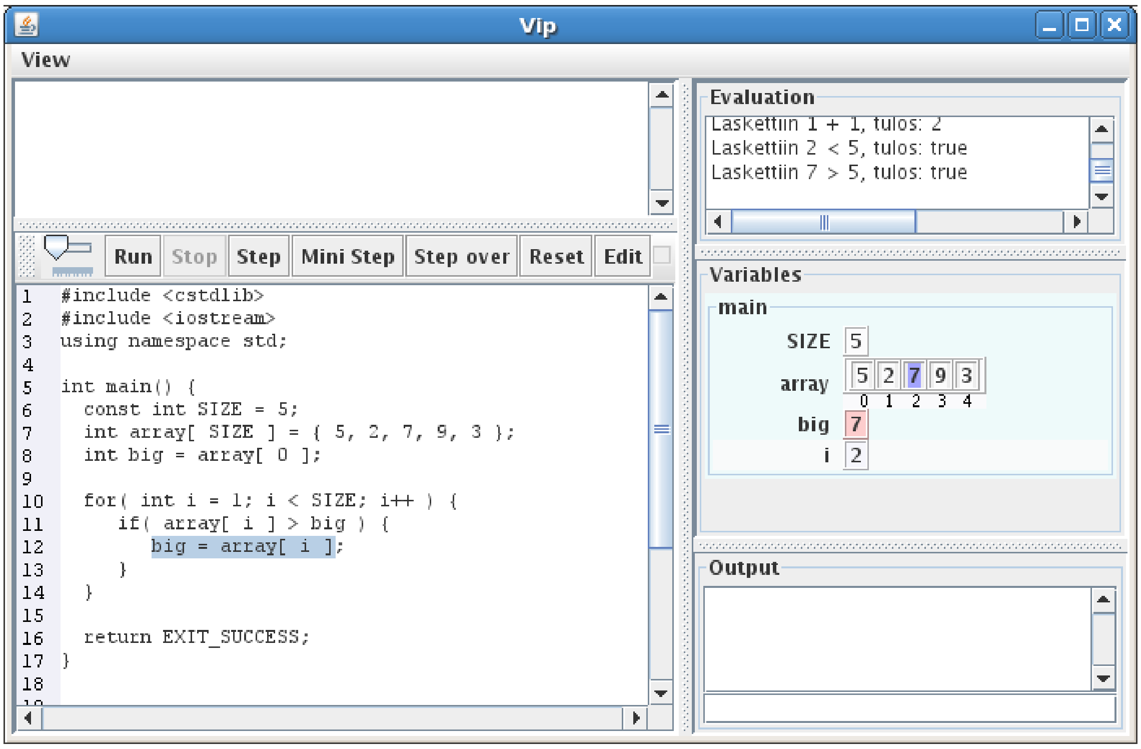
Because very little is known about how students really learn programming with visualizations in their everyday life, we focused on an explorative research approach. We started our investigation with the following very open research questions designed to address the mentioned minority of students at TUT that use VIP regularly and enthusiastically by choice:

(1) How do these students engage with the visualization tool VIP in the long term?
(2) How do they benefit from the use of VIP?

For this purpose, we designed and conducted an explorative study following the *Naturalistic Inquiry* paradigm as explained by Lincoln and Guba (1985, p.36ff) and utilizing the *Grounded Theory* approach as defined by Corbin and Strauss (2008). In our study, we worked mainly with single cases and analyzed their similarities and differences in order to develop a data-driven conceptualization of student engagement with visualizations. The purpose was to provide information about how visualization tools should be developed and used as educational resources so that students can benefit from them. Knowledge on how students interact with the tools will help both visualization tool developers and teachers using the tools to gain understanding on how to improve their work. In the next section, we introduce our study in detail.

3 Research Study

In this section, we introduce our two-part research study. This study was previously presented in Isohanni and Knobelsdorf (2010) and Isohanni and Knobelsdorf (2011). In the next subsections, we first describe the course setup of the introductory programming course at TUT and the visualization tool VIP. Then, we give a summary of the data collection and analysis of our research study.



```
#include <cstdlib>
#include <iostream>
using namespace std;

int main() {
  const int SIZE = 5;
  int array[ SIZE ] = { 5, 2, 7, 9, 3 };
  int big = array[ 0 ];

  for( int i = 1; i < SIZE; i++ ) {
     if( array[ i ] > big ) {
        big = array[ i ];
     }
  }

  return EXIT_SUCCESS;
}
```

```
Laskettiin 1 + 1, tulos: 2
Laskettiin 2 < 5, tulos: true
Laskettiin 7 > 5, tulos: true
```

```
SIZE  5
array 5 2 7 9 3
      0 1 2 3 4
big   7
i     2
```

Figure 1 A screenshot of VIP.

3.1 *Course Setup and VIP*

The introductory programming course at TUT is a three-month, first-year C++ course. Here, the students were required to complete four programming projects and an exam, but weekly exercise sessions and corresponding pre-assignments were voluntary. VIP was introduced by the instructor of the programming course (author Isohanni) during the lecture, and students were encouraged to use it in multiple ways following the model of integrating visualizations into the course content presented by Lahtinen (2006). The visualizations were available for students on the course web site for voluntary use. The course material integrated the visualization examples by including links and references in all study materials. This established a tight connection between the visualization examples and the other study materials used in the course, but the students were still left to decide whether to use the visualizations and the VIP tool, and their decision did not affect the grading process by the teacher.

3.1.1 The visualization tool VIP

VIP supports imperative programming in C++ and is based on an interpreter that uses a simple subset of C++. Given regular source code files, the interpreter executes them and automatically illustrates the program execution. With VIP, the program can be executed either by stepping or by running, the latter with an adjustable speed. Of course the program code needs to compile correctly in order for its execution to be illustrated. (Virtanen et al., 2005)

The VIP main window (see Figure 1) is composed of five smaller windows. 1) *The code window* shows the program code that is executed and illustrates the execution by highlighting the relevant line of code. 2) *The variables window* draws pictures of the variables and data structures and highlights parts of the pictures as the values are changed or referenced. 3) *The evaluation window* is activated whenever the code window executes an expression. It shows the values of the operands, the operators, and the resulting expression. 4) *The input-output window* prints the output and accepts input from the user. 5) When needed, *the view window* shows comments that explain to the user what is happening. *The VIP code editor* opens in a separate window. Using the compile button, the user can go back from the editor to the main window for the visualization.

3.1.2 Teaching Programming with VIP

In the lecture, the teacher introduced VIP and showed how it can be used to follow the execution of example programs. The teacher also demonstrated how VIP can be used to complete small programming assignments: this procedure included testing and debugging written code.

Testing and debugging with VIP is different from performing these activities in a traditional program development environment because, in addition to the program code and its output, the programmer also sees multiple presentations of the state of the program. For example, when testing a program in VIP, the student can follow the execution of the algorithm line by line and at the same time see the content of the data structures. Thus, errors can be recognized immediately, not just when the program produces a wrong result. For this reason, the students were advised to follow how the execution proceeds while testing the program.

The strategies taught both for testing and for debugging a program in VIP were similar to the single stepping strategy by Romero et al. (2005). They explained that when performing single stepping, the student steps through program code by looking at the visualized data structures or output. The students were instructed to use the multiple views of VIP and not revert to traditional debugging strategies such as hand-simulation and causal reasoning, which were described by Katz and Anderson (1988). Hand-simulation means that the student mentally executed the program as the computer would and looked for inconsistent behaviour. Causal reasoning means that the student received information about the error by looking at the output of the program and then decided what might be causing the behaviour. While these traditional debugging strategies were also explained during the programming course, they were not connected to the use of VIP. In addition, the teacher did not explicitly name any of the debugging strategies.

3.2 Data Collection Methodology

From a methodological point of view it would have been best to conduct an ethnographic field study, following students for the whole semester and observing their long-term engagement with VIP. However, due to limited research resources it was not possible for us to conduct such a study, so instead, we designed a two-stage data collection and analysis approach that permitted us to enter the research field nevertheless. In Isohanni and Knobelsdorf (2010, p. 89-90) and Isohanni and Knobelsdorf (2011, p. 34-35) we presented the data collection and analysis in detail. Here, we will provide a brief summary.

3.2.1 Data Regarding the Long-Term Engagement

In the first stage, we collected data on how students use VIP in the long term during the whole programming course. For this purpose, we collected both the log files of students' VIP usage and asked the students to complete weekly questionnaires. The questionnaires contained both multiple choice and open-ended questions which inquired about their backgrounds, asked detailed questions about how they learn programming, and asked about their reasons for using the visualizations.

VIP contains an integrated automatic logger module. Every time VIP is launched, the logger module will send the log information to the server through the Internet. The log files contain all the operations a student performs on the VIP user interface, i.e., run, step, edit, or compile. The operations are marked with both a user name and IP address. Additionally, there is a timestamp for each operation. Every time a new version of the code is accepted for the visualizations, the logger stores the source code file in the log information as well. In the second stage, we used this collected background information to identify students who used VIP often in their own study sessions and had 8 students attend a combined observation and interview session at the end of the course.

3.2.2 The Interviews

The observation and interview sessions were performed approximately two weeks before the course final exam in a usability laboratory with a computer. A video camera recorded the computer screen used by the student and the voice dialog between student and interviewer. The session consisted of three phases. First there were general, easy questions about the interviewee's opinions on learning programming. Then, the participant was asked to complete a short programming assignment on the computer and explain his or her actions. The interviewer followed the student's progress and occasionally asked questions about what the student was doing. Finally, there were some further questions related to the visualization tool. Both interview parts were semi-structured. The session lasted 30−60 minutes depending on how much the student talked and how long it took to complete the programming assignment. For further details see Isohanni and Knobelsdorf (2010, p. 89).

3.3 Data Analysis

The data analysis followed a two-stage process in which we first investigated the eight students' observations and interview session data. The interviews were transcribed and the first stage of data analysis was based on qualitative content analysis by Mayring (2000), which is a methodology from qualitative social science research used to systematically code and analyze textual data. The result of this data-driven analysis was a system of categories and subcategories that evolved during the multi-phase creation process. The categories helped to reconstruct students' activities with VIP and to identify characteristics that defined utilization strategies the students showed. For further details, see Isohanni and Knobelsdorf (2010, p. 90-91). This first part of the data analysis resulted in a conceptualization of students' working patterns that is presented in Section 4.

In the second part of the data analysis, we analyzed the log files of five of the eight interviewed students in order to capture the "whole story" of their long-term engagement with VIP. For analyzing the log files, we chose a grounded theory (GT) approach as introduced by Strauss and Corbin (1995). Since the analysis of the interviews was performed prior to the analysis of the log files, we were also able compare the reconstructed behaviour with the results gained in the analysis of the interviews. Such comparisons were used in validating the assumptions we had to make during the log file analysis. This second part of the data analysis resulted in a conceptualization of four different use phases of VIP that are presented in Section 5.

4 Students' Working Patterns Using VIP

The results of the observation and interview session contain three parts that build upon one another and represent different levels of abstraction: *the category system, the use situations*, and *the working patterns*. In Isohanni and Knobelsdorf (2010, p. 90-93), we presented these results in detail. Therefore, in the next two subsections, we will summarize them briefly. Building upon this, we will present a grouping of the working patterns in Subsection 4.3, relating this to our research questions in Subsection 4.4.

4.1 Use Situations

The *category system* developed from a data-driven analysis focuses on what activities the students performed while working on their assignments. This represented a first abstraction of the data. The categories grouped and structured the different aspects of students' activities. When we analyzed the transcripts of the student sessions further, we found that certain sections of them were more relevant for our research questions than others; this helped us to focus on *use situations*.

A *use situation*, representing the second level of abstraction, referred to a series of actions the student performed in the VIP main window. Each use situation only represented a short segment of the whole assignment session. In total, 35 use situations were reconstructed from the data, with each assignment interview containing 2−7 use situations. The reconstructed use situations contained three parts: 1) initial state, 2) action, and 3) final state. Next we exemplify two such use situations.

4.1.1 Example 1

Initial state: The student was writing a program that sorted the content of an array in ascending order. He had just started to accomplish the assignment by writing a new piece of code that was supposed to find the smallest integer in the array. He was uncertain if the code worked properly and also did not have a clear idea of what he should do next. Thus, he decided to open the main window of VIP to visualize the execution of the program.

Action: He started to execute the program line by line using the step button. He was analyzing the program's execution very carefully, looking at both the code window and the variables window. He was pressing the step button non-stop and watching the representations change on the screen thinking aloud what should happen next.

Student: Oh, it only reaches that line now. I was ahead in the execution myself.

After following the execution for some time, he decided that he can make the program run faster because the program is repeating the same thing he had already seen and he wanted to see the end result more quickly. He stopped the fast running when the program was close to the end and changed back to stepping again line by line. He started analyzing the program's execution again. He looked at the variables window:

Interviewer: What do you see from VIP now?

Student: Well. . . I see that it [my program] works correctly. At least temp [a variable]. And I need to swap that [the value of the variable smallest] to the beginning [of the array]. I suppose.

Final state: By the end of this use situation, the student had used VIP to verify that his program works correctly so far. He also had an idea of how to proceed with programming.

4.1.2 Example 2

Initial state: The student's task was to make modifications to a program that was handling a simple coordinate system. He was working with the assignment for while and had already corrected a couple of errors in his code. He wanted to test his program to see if it was working correctly now.

Action: He started to execute the program fast. The program required typing input coordinates, and he typed values that were supposed to be legal. He expected the program to print the coordinate system, but the program printed a message saying that the input was illegal.

As a consequence, VIP stopped running the program further and all representations in the windows were static. The student started analyzing the program's execution using these static representations. His objective was to understand what happened when the program was running. He looked mainly at the code window but also used the evaluation window and the variables window. He kept talking and explaining what might have happened, and it sounded as if he was running the program in his head exactly the way VIP ran it.

Student: So it [the control] was on this line of code here [pointing in the code window] moving pretty fast and... It went to this scope [pointing at the code window] because here... [muttering] [looks at the evaluation window] The size of the coordinate system... [looks at the value in the variables window] Ummmm...

After this, he was reasoning not only what happened during the run but also why it happened. He concluded that the less-than-sign of the if-structure's condition (`coord.size() > y`) had to be exchanged with a greater-than-sign. He did not explain the reasons for this conclusion to the interviewer and only made the change in the code editor.

Final state: By the end of this use situation, the student had used VIP to test his program with one input and found an error. He also found the reason for the error and how to correct it.

Further in the analysis and interpretation process, we identified differences and similarities between the *use situations* and conceptualized their characteristics to *working patterns*, which constituted the third level of abstraction and the final results of the data analysis.

4.2 Working Patterns

Two major characteristics emerged when grouping the use situations: the student's use objective and the student's use strategy for achieving that objective. We combined them to a *working pattern* which we understand as a model that conceptualizes the use situations. Altogether, we found three different use objectives: 1) debugging, 2) testing, and 3) exploring. Among them, we identified eight different use strategies. When we describe the working patterns, we switch to present tense since the working patterns are a model that conceptualize students' behaviour when using visualizations.

Subsection 4.2.1 will first introduce the working patterns on the level of the three use objectives to give an overall understanding. Then, the use strategies are described in detail in Section 4.2.2. In Section 4.3, we start forming a general understanding of the working patterns for further interpretation by grouping them according to specific similarities.

4.2.1 Use Objectives

The objectives reconstructed from the use situations are:

1) Debugging: The students find an error in the program and use VIP to locate it and to understand the reason for it in order to be able to correct it (the use situation presented in Section 4.1.2 is an example of this). The use situations related to this objective resulted in three working patterns.
2) Testing: The students want to see what happens during the run of the program. The objective is to test whether the program works correctly (the use situation presented in Section 4.1.1 is an example of this). If an error is found but the student makes no attempt to debug the program, the objective of using VIP is testing. The use situations related to the objective testing resulted in four working patterns.
3) Exploring: The students start working on the assignment by running the program code that was provided, to see how it works. They do not have a clear idea of what the program should do before running it. The objective is to become familiar with the program. Use situations related to the exploring objective resulted in one working pattern.

4.2.2 Use Strategies

The use strategies were grouped in the following subsections according to the corresponding use objective. Every use strategy is composed of different activities students performed in their use situations. Seven of the working patterns we noted are summarized in Figures 2 and 3. The rectangles in the figures correspond to one activity. A dotted rectangle indicates that the corresponding activity was not always performed in a use situation of a student although he or she exposed all other activities that represent the particular working pattern.

With the category system, we captured students' activities of their use situations. We have not presented the category system in its entirety (as it is described better in Isohanni and Knobelsdorf (2010, p. 90-91)) but we provide some examples to make the working patterns understandable. The subcategory *Running program in VIP* from the category *programming actions* means either using the run or the step function of VIP.

Category	Example
Analyzing the program's execution	"The [input] coordinates are correct, but it should have moved this to that position."
Analyzing why something went wrong	"I started thinking that there might be something interesting with the indexing here... That the coordinates (3, 3) actually refer to somewhere else than there [that square]."

Table 1 Examples of two categories.

From the category *cognitive activities*, we introduce two subcategories: The students had different approaches when reasoning through the execution of their program in VIP. The subcategory *analyzing the program's execution* includes students' thoughts about what is and was happening, as well as what should have happened during the execution of the program, without analyzing the reasons. Therefore, the other subcategory *analyzing why something went wrong* includes talking about the reasons for the program execution. Table 1 introduces text excerpts to illustrate these two subcategories.

Debugging

The first working pattern with the objective *debugging* is **Dynamic debugging in VIP**, illustrated with a normal, black arrow on the left in Figure 2. The students execute their programs in VIP and exploit the visualizations during the execution both in finding the error and in analyzing the reason for it. Students can do this by running the program once and changing their perspective of analysis, reasoning first about what was happening and then about why it happened. Alternatively, students can run the program twice, observing it in different ways each time. For instance, the first execution might be very fast using the run function while the second might be slower using the step function.

The second working pattern with the objective *debugging* is **Static debugging in VIP**, illustrated as the gray path in the middle of Figure 2. Here, the program is not run in VIP after the students find an error. Instead, the students analyze the reason for the error by looking at the static representations in VIP's windows once it has stopped (the use situation in Section 4.1.2 exemplifies this). This means that students are actually performing the traditional debugging strategy called hand-simulation introduced in Section 3.1.2 and using VIP's window as support.

The third working pattern with the objective *debugging* is **Immediate reasoning**, illustrated with the dashed arrow in Figure 2. Here, the students understand the reason of the error immediately after seeing what happened during the run of the program and there is no further need to work with VIP.

Testing

The working pattern **Abandoning VIP** has the objective *testing* and is illustrated with the dotted arrow on the right in Figure 2. In these cases, the students find an error in their program, but instead of debugging it, they totally stop using VIP. Therefore, the objective for using VIP here is testing. After that, they can move to the editor window to think about how to correct the error or give up on the assignment.

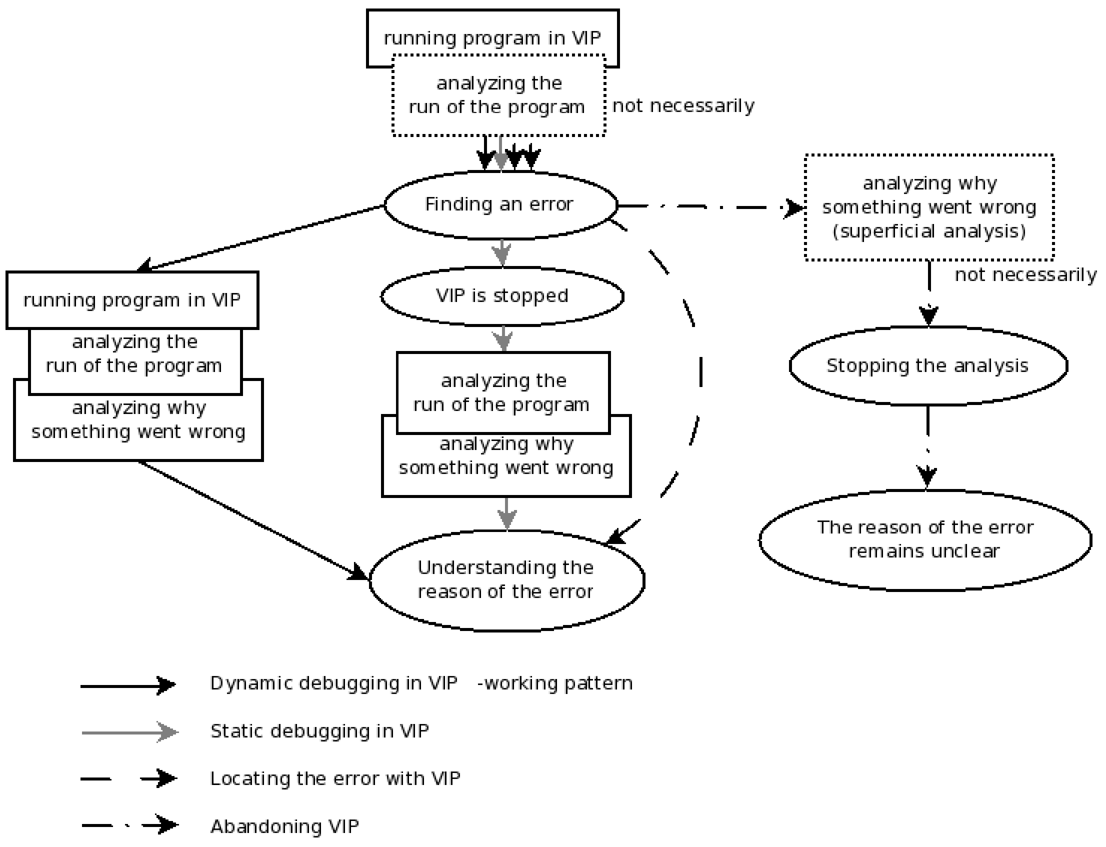

running program in VIP
analyzing the run of the program
not necessarily
Finding an error
analyzing why something went wrong (superficial analysis)
not necessarily
running program in VIP
analyzing the run of the program
analyzing why something went wrong
VIP is stopped
analyzing the run of the program
analyzing why something went wrong
Stopping the analysis
The reason of the error remains unclear
Understanding the reason of the error
Dynamic debugging in VIP -working pattern
Static debugging in VIP
Locating the error with VIP
Abandoning VIP

Figure 2 Working patterns where the students are dealing with an error in the program.

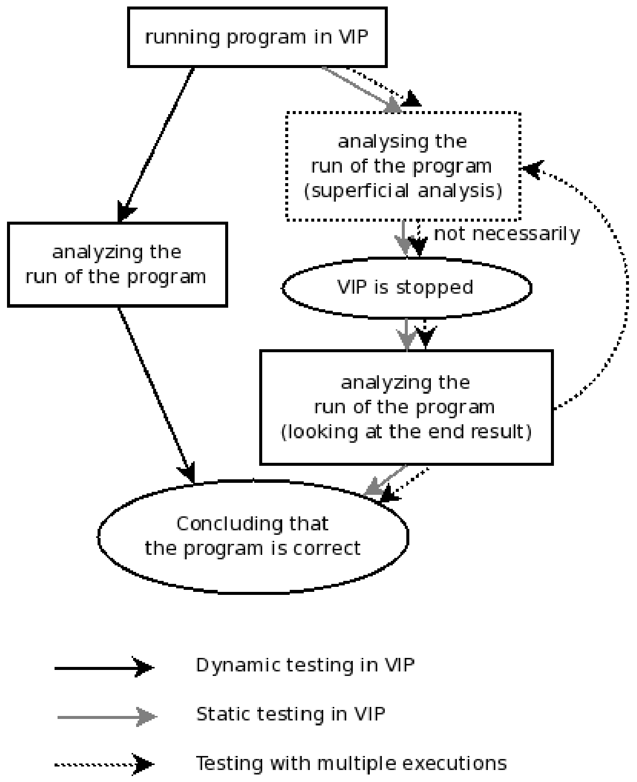

running program in VIP
analysing the run of the program (superficial analysis)
not necessarily
analyzing the run of the program
VIP is stopped
analyzing the run of the program (looking at the end result)
Concluding that the program is correct
Dynamic testing in VIP
Static testing in VIP
Testing with multiple executions

Figure 3 Working patterns where the students are testing their programs without finding an error.

The rest of the working patterns with the objective *testing* describe situations where the students do not find errors in their program during testing it and thus only concentrate on analyzing the run of the program. They are illustrated in Figure 3.

The working pattern **Dynamic testing in VIP** is illustrated with a normal, black arrow on the left in Figure 3. In this working pattern, the students execute their programs in VIP and analyze the run of the program during the execution exploiting the visualizations at the same time as in the use situation in Section 4.1.1. The analysis concentrates on how the algorithm works and what happens in the internal state of the program during the execution.

As distinct from the previous, the working pattern **Static testing in VIP** illustrated as the gray path in the right of Figure 3 describes testing based on a superficial output analysis. Here, the students execute the program fast either observing superficially or not observing at all and then make their conclusion about the correctness of the program by relying on the end state they see in VIP's window. Most often the analysis concentrates only on the output window of VIP.

The last working pattern with the objective *testing* is called **Testing with multiple executions**. It is illustrated with the doted arrow in Figure 3. Literally, the students execute their program multiple times with different inputs and the objective seems to be to achieve some sort of test coverage. Typically, the observation of the execution in this case is superficial as described in the working pattern **Static testing in VIP**.

Exploring

The students were provided a section of program code with the assignment description and they start working by running the code to see how it works. They do not study the assignment description too much before starting exploring the code and the objective is to become familiar with the program.

No variation in *exploration* strategy emerged from the data. All of the students were analyzing the run of the program during the execution, and in addition, the output of the program can be analyzed after the execution is finished. Also, multiple executions can be used.

4.3 Grouping the Working Patterns

In the further data analysis, we noticed the strategies the students were applying in certain working patterns capture a similar way of using VIP even if the use objectives of the patterns are different. For example, the use objectives in the working patterns **Dynamic debugging in VIP** and **Dynamic testing in VIP** are different. But, the strategy of both of these working patterns describe a similar way in which students follow a program's execution in VIP and at the same time analyze the program's behaviour exploiting VIP's visualizations.

For further interpretation, we analyzed the different ways of incorporating VIP's visualizations in the strategies of the working patterns. For this reason, we grouped those working patterns together that captured the "same amount of attention" a student paid to VIP visualizations while following the programming execution in VIP. We are aware that this grouping criterion is not sharp enough to form new categories of VIP engagement, but our objective is to point out the different "levels of visualization engagement" we observed in the working patterns and to give an overall understanding of the different ways students can take advantage of visualizations. The different groups are as follows:

- The first group, **Using VIP fully**, covers the working patterns **Dynamic debugging in VIP**, **Dynamic testing in VIP**, and **Exploring**. In all these patterns, the students' full attentions are focused on VIP, and they use the control functions and multiple different windows of VIP to visualize the execution while analyzing their program code (example in Section 4.1.1), i.e., practically all features of the visualization tool support their work. We can say that they use VIP fully as a visualization tool, and thus named this group **Using VIP fully**.
- The next group, **Using VIP partly**, covers the working patterns **Static debugging in VIP**, **Static testing in VIP**, and **Testing with multiple executions**. In all these patterns, the students incorporate a limited set of VIP's functions. They can, for instance, deploy the static presentations showing in VIP when the execution is stopped (example in Section 4.1.2) or use VIP just like any other programming environment that executes a program and displays its output. VIP could offer them a dynamic representation of the program execution but they do not use it. Therefore, we named this group **Using VIP partly**.
- Then, we have two working patterns where the students do not use VIP's visualizations at all or use them very little: **Immediate reasoning** and **Abandoning VIP**. **Immediate reasoning** captures the situation where the students are able to create a correct program and do not need any support for analyzing errors. **Abandoning VIP** on the other hand, captures the situation where the students leave their programming task unfinished because they are unable to accomplish the task with the support of the visualizations. Since the reasons behind this low engagement were very different, we distinguished both patterns with a group of their own. The group containing the working pattern **Immediate reasoning** was named **No need for VIP** and the group containing the working pattern **Abandoning VIP** was named **Unfinished use of VIP**.

Merging the groups together, we have presented the students' different engagement levels using VIP visualizations to support their programming tasks (see Table 2).

Working pattern	**Group**
Immediate reasoning	No need for VIP
Dynamic debugging in VIP Dynamic testing in VIP Exploring	Using VIP fully
Static debugging in VIP Static testing with VIP Testing with multiple executions	Using VIP partly
Abandoning VIP	Unfinished use of VIP

Table 2 Working patterns and their corresponding groups.

4.4 Discussion of the Results

The earlier studies on the use of visualizations (see Section 2.2) included a study by Sorva (2012) which described students' perceptions of what it is to learn programming through a visualization tool. The findings by Sorva were connected to each other logically forming a structure describing simpler and richer understandings of using the visualization tool. This structure can also be compared with the grouping of the working patterns.

The behaviour described Sorva's category of richest understanding of learning through the visualization tool is similar to what we described in the group **Using VIP fully**. Both of these describe how the students exploit the visualizations when working with the program code. In Sorva's category of a simpler understanding named as "learning to recall code structures" the students are paying very little attention to the visualizations and instead see the visualization tool as a platform for encountering example code. This compares to some behaviour described in our group **Using VIP partly**. Sorva's work is very similar to ours. It just seems that the perspectives of these two studies are slightly different. Our attention is more on students' actions and Sorva concentrates on what the student is learning. The findings of these two studies could complement to each other.

4.5 Relating the Results to Research Questions

Our first research question focused on how students engaged with VIP in the long term. In this subsection, we will relate the results to this research question.

The working patterns describe different ways of how students interacted with VIP when learning programming. We were able to distinguish different use objectives and use strategies in students' programming sessions. Students use VIP to support debugging, testing, or exploring their program code. With eight different working patterns we described how students use VIP and its visualizations to accomplish their different use objectives.

By grouping these working patterns, we captured different levels of engagement that students achieve when they interact with the visualizations of VIP. The group **Using VIP fully** subsumed working patterns where the students engage with VIP in a comprehensive way to aid their thinking when solving a programming problem, meanwhile **Using VIP partly** included patterns that present a partial engagement with the visualization. In the remaining two groups, no engagement with the visualization was displayed.

The working patterns and especially their grouping reveal and describe different levels of engagement with the visualization that is the result of the students' long-term uses of VIP during the whole programming course they attended. Thus, they are a part of the answer to the first research question "How do students engage with VIP in the long term?"

Beside the observational and interview session, we also collected and analyzed data about student engagement during the whole course. In the next section, we will describe the corresponding results. In Section 6, we will return to the working patterns and present two single case studies that describe how a student's engagement with VIP can possibly develop in more detail.

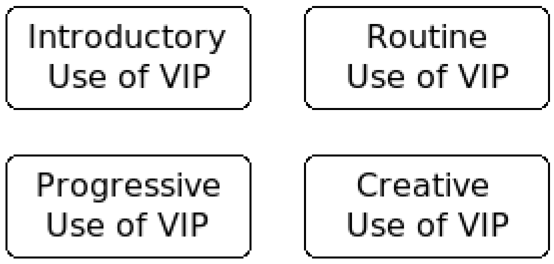


Figure 4 The phases of using VIP.

5 Long-Term Engagement with VIP

From the log files, four phases emerged that cover the development of student engagement with VIP when learning to program. These phases are illustrated in Figure 4. Each phase summarizes a student's activities and, when possible according to the log file analysis, also the purpose and development of the activities. Therefore, the phases serve as an analytical tool that captures the development in a student's log file for further interpretation. Thus, we describe them using the present tense similarly to the working patterns.

In the next subsections, we introduce and describe each phase and the concepts that relate to it in a generalized way. In Section 6, we concretize the generalized perspective by giving examples in the field in context of two use stories that present how two students used VIP and benefitted from it during the whole programming course.

5.1 Introductory Use of VIP

For students who enter VIP for the first time with the intention of learning about the functions of VIP only and not studying programming, we call the first phase of the engagement **introductory use of VIP**. In this phase, students use the controls of VIP in a tentative way (e.g. they can try out the speed setting to determine how fast the maximum and the minimum speeds are).

The user interface and functions of VIP are so simple that some students do not perform such an introductory phase at all. For the ones who did, it lasted 1-2 sessions. However, this phase did become interesting during the analysis of the students' development because it can affect the further development of the student engagement.

5.2 Progressive Use of VIP

During the **progressive use of VIP** phase of the engagement, students use VIP to learn programming, but the personal manners of using the tool have not developed yet. Mainly, students are following the instructor's recommendations on how VIP can be used (see Section 3.1.2) but experimenting with other use purposes might occur as well. Students can continue testing the controls of VIP and they can learn to use them more efficiently. They are developing their personal habits of learning to program in VIP during this phase through testing different kinds of activities.

If the development does not end after a couple of sessions, this phase is intermediate. Once VIP becomes a familiar environment for the student, the development leads either to a **creative** or to a **routine use of VIP**.

5.3 Creative Use of VIP

After having learned to use VIP efficiently in the earlier phases, the students are able to create their personal way of using VIP as beneficially as possible. When this happens, the students enter the **creative use of VIP** phase, in which they take an active role in choosing the VIP examples to be used in learning and creating their own use purposes of VIP. According to the constructivist theories of learning, choosing the study materials and methods according to one's own experiences is part of personal knowledge construction (Illeris, 2002).

In this phase, the purpose of using VIP varies between sessions more than in the other ones. Typical use purposes of VIP in this phase are debugging program code that was written in another program development environment; using VIP as a program development environment when writing part of a larger programming project; investigating how new features of the programming language work; and revising and editing VIP examples whose topics are related to one's own programming project. Next, we will describe several characteristic behaviours that were connected to this phase.

5.3.1 High commitment to the visualization

In the log files we observed moments when the students were executing their program code in VIP slowly by clicking the step button several times with clear pauses between the clicks. Relying on students' use behaviour during the observational and interview session (Isohanni and Knobelsdorf, 2010), we interpreted that while clicking the step button slowly they follow what is happening during the execution on the screen. Judging the log files, we were not able to precisely determine which of the VIP windows was being followed because there were changes in all five windows simultaneously. However, all the information presented in VIP was related to how the program execution proceeded. Therefore, we interpreted that, at this point, the students focus on following the visual presentations of the program structures or the progress of the program execution. This behaviour was named high *commitment to the visualization*. In general, *high commitment* was perceived in each student's personally preferred activities performed during the **creative use of VIP** phase when students use the visualization tool in a personalized way. It occurred in other phases of using VIP as well, but it was not recurrent then.

5.3.2 Different Purposes of Using VIP

Accomplishing their programming projects during the course, the students were using VIP in multiple different ways. The use purposes that were central in the **creative use of VIP** phase were named *debugging program code, using VIP as a programming environment*, *revising programming concepts*, and *investigating programming concepts deeper*. They characterize the phase very well since such use purposes had not been presented by the instructor. The instructor did not suggest completing the programming projects in VIP since they were unreasonably large to be mastered in VIP code editor.

However, the students solved this problem by investigating only small parts of their code in VIP at a time. We will briefly describe these four use purposes below.

When using VIP to *debug program code* the students copy and paste program code between the VIP code editor and another programming environment they are working in parallel to using VIP. They follow the execution of a particular part of the program using the step button slowly (*high commitment to visualization* occurs in such cases). After the error is detected, the students correct it either directly in the VIP code editor and repeated the visualized execution or move back to the other programming editor in use. When *using VIP as a programming environment* the students start developing new functionality to their program in the VIP code editor. They repeat a similar cycle of editing the code and visualizing the execution as during using VIP to *debug program code*. It is also possible that they start writing a program from scratch in the VIP code editor directly and behave in the same, recurring way. These two use purposes can also be combined in a session. When *revising programming concepts* or *investigating programming concepts deeper*, the students execute example programs prepared by the teacher, or write and edit small code snippets related to the topic of the programming project. In a way, they use VIP instead of the written course materials.

The use sessions where these purposes were detected lasted for multiple hours, so it was possible to follow the development of students work during a longer period and thus detect how it was related to working on the programming project of the course. During the data analysis, we had to make several assumptions about these concepts. The log files only contained information on the use of VIP so we could not know what the students were doing in the other programming environment involved. We assumed that students' motivation for using VIP was their work on the programming project elsewhere. This assumption was based on the facts that during the multiple-hour-long session, the students were looking at multiple code examples that were closely related to the topic of the project and making modifications that were related to the assignment of the programming project, and that this all happened when the deadline of the programming project was approaching.

5.4 Routine Use of VIP

In this phase of development, students adopt a certain procedure of using the visualization tool and keep repeating it without variation or change. Such a routine can be to open VIP weekly in order to perform the same task, to complete all the assignments on a certain list, or to repeat the same manners of using VIP in each use session over and over again. We call this stage of engagement the **routine use of VIP**. For an observer, this kind of engagement with VIP might seem to lack motivation and meaningful reasons. But according to the students' comments in the questionnaires, they see VIP as a useful and beneficial tool for programming, even in this **routine use of VIP** phase.

In the following two subsections, we outline two students' experiences of using VIP in the described routine form. Both subsections describe a case of one student.

5.4.1 Omit adjusting the visualization

In the **progressive use of VIP** phase, this student was adjusting or learning to adjust the program execution in VIP by using the controls in various ways. However, in the **routine use of VIP** phase she always followed the visualization using the same settings (e.g. using the function run instead of swapping between run and step according to the context). For example, the student used one speed of execution only during one whole VIP session or changed the speed very rarely. Following visualization for a long time in only one speed can be monotonous for the user; the interesting execution parts are not distinguished from the routine parts and recognizing the important functions on the screen requires great concentration. It appears that this student repeated the program code execution in VIP quickly between modifications just to check whether the execution was possible or to see the result of it (i.e. the output of the program). Obviously, using a traditional compiler that executes the program much faster than VIP would have been much more practical. Hence, it is doubtful whether using the visualization in this way would be beneficial for the students, even if she claimed so in the questionnaires.

With regard to this use behaviour, we can only assume why the student judged VIP to still be beneficial and useful. Since the teacher advised using VIP for programming, she might have believed it was beneficial without really experiencing this.

5.4.2 Willingness, but no patience to follow visualization

This student opened VIP regularly every week the evening before he attended the exercise session of the programming course. Every time, he completed an assignment for the exercise session using VIP. Mostly his VIP sessions were short; he just used the visualizations one time to see that everything worked as it should. It seemed that the assignments were so easy that he could have solved them without using VIP, but he still persistently decided to open VIP every time.

According to his answers of the weekly questionnaires, he was convinced that program visualizations were useful in general. However, he also sometimes mentioned that the assignment was so easy that on that particular time he did not really need the visualizations. This explains why the sessions were often quick. The way in which he worked gave an impression that he is perfectionistic and wanted to solve the assignments as flawlessly as possible. This seemed to lead him to using VIP regularly even if he was able to solve the assignments as well without VIP.

5.5 Discussion of the Results

The earlier studies on the use of visualizations (see Section 2.2) did not distinguish different phases of using the visualization tool in the long term but different ways of using the tools are identified in some studies. In our category system, these were placed in the category *Different Purposes of Using VIP* (see Section 5.3.2). We compare this category to the findings presented by others.

Kannusmäki et al. (2004) studied the ways in which students solved exercises and presented four different patterns of working that the students performed during their observations. The visualization tool Jeliot was used in two of these patterns. In the other one, students used Jeliot both for writing code and visually testing/debugging it and in the final one, the code is written in another environment and Jeliot was used only for visually debugging and/or testing the program. Both of these ways of working were also found in our study. In our category system, they were named *using VIP as a programming environment* and *using VIP to debug program code* (Section 5.3.2). We did not limit our study to the ways in which students solved exercises using VIP, so we also found other kinds of use purposes for it. We assume that Jeliot could also be used for the other use purposes we found, but they were not covered in the study of Kannusmäki et al. (2004) due to their focus on the exercise solving only.

Also Moreno and Joy (2006) describe the use of Jeliot in their work. They provide two categories: ”Jeliot as a learning aid” and ”Jeliot as a debugger.” They failed to describe these categories on a general level and instead relied on multiple quotes of students' comments. Thus, it is difficult to compare the findings of Moreno and Joy (2006) to the results of Kannusmäki et al. (2004) or our study very precisely. It seems that the category ”Jeliot as a debugger” is somewhat similar to the ways of working described in the previous text paragraph and the category ”Jeliot as a learning aid” has some similar characteristics as the categories *revising programming concepts* and *investigating programming concepts deeper* of our category system (Section 5.3.2).

Regarding these comparisons, we concluded that all these studies have found similar ways of working with the visualizations and complement the knowledge provided by each other. This gives the impression that the results of these studies are valid and describe how students act with visualizations since different studies using different visualization tools revealed similar behaviours of the students.

5.6 Relating the Results to Research Questions

In this subsection, we return to our first research question on the students' long-term engagement with VIP. The working patterns and their grouping in Section 4 described different levels of students' engagement with VIP that the students performed at the end of the programming course. The use phases reveal what kind of stages can lead the student to this behaviour.

The use phases can be used to describe a certain development in students' long-term engagement with VIP. This development can start with **introductory use of VIP** and become more intensive in the **progressive use of VIP**. The **creative use of VIP** phase represented the strongest personal development in the engagement with VIP, while **routine use of VIP** indicated that the use of VIP continued despite the engagement with the visualization not being strong. Since we did not group the phases to a particular order, they rather describe different episodes of the overall engagement during a long-term process.

Together with the working patterns and their grouping, the phases describe the students' interactions and engagement with VIP and thus give an answer to the research question 1. The use phases and the working patterns are a thorough description of the nature of the engagement where as the deeper reasons for the engagement will be analyzed in Section 7 where we return to these results one more time relating them to the interpretation with the activity theory. Before that, we will exemplify the use phases in the following section by relating them to the stories of two students.

6 Two Stories of Using VIP

The four phases described in the previous section combined together describe a certain development a student undergoes when using VIP. From a qualitative research perspective, we present this development in a descriptive form that we call *a use story*. By a *story* we mean the product of one student's reconstructed development by the researchers using the developed structure in the previous sections based on both the log files and the interview and observation sessions. Being the product of the data analysis,

Introductory Use of VIP
Progressive Use of VIP
Creative Use of VIP

Figure 5 Phases Tomi reaches during his use story of VIP.

the *story* is told from the researcher’s perspective not by the student him/herself.

Multiple kinds of different combinations of the phases are possible in one VIP use story. In this section, we present the different stories of two students. The two students have been chosen because their development in using VIP was very diverse and went through different phases. Both stories begin with the students’ first interactions with VIP and summarize their VIP sessions during the 12 weeks of the programming course. The stories are divided into subsections according to the phases described in the previous section. In the last subsection of the stories, we also describe the interview and observation session of each student. Hence, this subsection shows the end result of this long-term development and how the working patterns described in Section 4 capture the students’ behaviours.

6.1 Tomi’s Story

Tomi was a first-year student of information and knowledge management. This was his first programming course in TUT but he had been programming a little before entering the university. He had written Basic programs of less than 400 lines of code. He was particularly creative in the way he used VIP as he had developed his personal style of using it in the **creative use of VIP** phase (see Figure 5).

6.1.1 Tomi’s Introductory Use of VIP

On the third week of the programming course, the next day after the lecture where VIP was introduced, Tomi opened VIP for the first time. In half an hour he opened 23 different example programs in VIP. The main motivation of the session seemed to be to investigate the functions of VIP control buttons; for instance, he pressed all the control buttons that appeared in the user interface of VIP. Thus, we distinguished this as the phase **introductory use of VIP**. However, it is also possible that he investigated some programming concepts at the same time since he opened so many different example programs. Thus, he might have already proceeded to the next phase of using VIP during this lengthy introductory session.

6.1.2 Tomi’s Progressive Use of VIP

The phase **progressive use of VIP** certainly began in Tomi’s next VIP session later during the same day as he was preparing his homework assignment for the exercise session of the programming course and considered using VIP for it. He followed the teacher’s recommendation, opened VIP, and looked at the template the teacher had set up in VIP for completing the homework assignment but closed it soon. Instead, he completed the homework assignment using a normal compiler and handed it in. He was searching for his personal style of using VIP and figured out that despite the teacher’s advice, he did not need VIP for the small homework assignment that he was easily able to complete using a normal program development environment.

In the next week, Tomi started testing his own use purposes of VIP: The deadline of the first programming project was approaching and Tomi was working on it. He opened a small example program in VIP and erased the code in the VIP code editor. He copied and pasted his whole solution to the programming project in VIP and executed it twice with the step function of VIP. Most likely, he came up with the idea as he faced a problem when working on the project. In any case, the teacher did not suggest using VIP for debugging in this way.

It took him a while to go through the code and *high commitment to visualization* can be observed in his log files: As his program execution continued, it also slowed, and we assumed he was following each step more and more carefully. Next, he found an error (he had used the logical OR operator instead of the logical AND operator). After correcting the error, he executed his program again using the step button. His activities were an iteration of interacting with the visualization and correcting the code. *High commitment to visualization* was observed every time he executed the code in VIP.

Tomi also tried to use VIP for a pre-assignment, set up in VIP by the teacher one more time. His way of using VIP here differed a bit from the ordinary style: Instead of opening the pre-assignment directly in VIP, he opened the same small example program he used in the previous session and copied and pasted the code from the assignment there. The *high commitment to visualization* did not occur in this session, and in the questionnaire, he stated “maybe” when asked whether VIP was useful in solving the pre-assignment. The whole session seemed like testing VIP. In addition to testing whether VIP was useful in solving a pre-assignment he was also testing the run-function of VIP. He tested the different speed settings: first the maximum value, then the minimum value just for 3 seconds, and then a value in between for a longer time. In contrast to previous sessions, in this session he was not using the step-function at all. This indicates that there was nothing especially interesting he wanted to follow very slowly. In his normal sessions he always executed program code he had chosen himself and thus there was something he wanted to follow slowly, whereas this time he was executing program code suggested by the teacher.

6.1.3 Tomi’s Creative Use of VIP

In the previous phase, Tomi tested using VIP like the teacher recommended and according to the needs he faced in his own programming sessions. In the following week he entered a new phase, **creative use of VIP**, and started using VIP only where he found it most beneficial−in his own programming projects. He had developed his personal manners of using VIP and stuck to them now.

That week he was working with VIP intensively. He opened it on three sequential days just before the deadline. One of these sessions was longer than usually and he spent two hours using VIP and his normal program development environment in parallel. He corrected multiple errors from his program and seemed to use VIP like a visual debugger.

To exemplify what Tomi was exactly doing during these long VIP sessions, we will next deliver a detailed description of Tomi’s typical way of working with VIP, which was a result of development he had gone through during the two earlier use phases of VIP. This example shows that Tomi learned to use VIP very fluently and to observe exactly the part of the execution he was interested in. Additionally, it exemplifies the concept of *high commitment to visualization* and *using VIP as a programming environment* (see Subsection 5.3.1).

Tomi was debugging his solution for the second programming project in the programming course. At the beginning of the session he copied and pasted his program code in the VIP editor and started to execute it. He had learned to use the functions of VIP very fluently and was alternating between the step and run functions and changing the speed of run frequently. For example, when he had figured out that the problem was in the end of the program code, he first executed the program using run with a very high speed, then turned the speed down slower as the execution approached the erroneous part, and eventually changed to use the step function assumedly to be able to follow very carefully. He used the step-function for 11 minutes and executed the code so slowly that he only presses the step-button twice per minute. We do not know what was happening during such 30-second-long breaks, but we may assume that he would not bother to click the step-button for 11 minutes if he was not following carefully how the execution proceeds.

Tomi also tried to use VIP when solving the third programming project of the course, but this time he was not successful. He had problems with the syntax related to records, which was the new topic for this project. Thus, he concluded that records did not compile in VIP. This was odd because he should have known they do compile in VIP as he had earlier looked at examples handling records in VIP. He found some other way of working out the assignment and completed the programming project without the help of VIP. This was the last time Tomi used VIP in his own study sessions.

6.1.4 Tomi's Observation Session

In the interview, Tomi explained that VIP was his “problem killer.” The interviewer gave him an assignment to write a program that sorts the content of an array in an ascending order, which was a challenging task at this stage of his studies. The interviewer helped Tomi with designing the idea of the algorithm to get him started a bit faster since there was not too much time for the observation. Tomi started to work in his normal program development environment. He had written a new piece of code that was supposed to find the smallest integer in the array. He was uncertain if the code worked properly and also did not have a clear idea of what he should do next. He was about to give up, but the interviewer motivated him to keep working. Hinting VIP, the interviewer asked: “What kind of problems do you usually kill with VIP then?”, which encouraged Tomi to copy and paste his program code into VIP and proceed with the assignment.

In the beginning of the VIP session, Tomi showed an astonishing ability to use it. He started with the working pattern *Dynamic testing in VIP* and during the use of VIP learned, that his program worked correctly. He was interested in following the execution carefully which helped him decide what to do next with his program. As he proceeded in writing his solution he also started making errors. The first two errors were handled by *Dynamic debugging in VIP*: He used VIP both in finding the error and in analyzing the reason for it. It seemed that he had a clear understanding of the state of his program and was able to master the assignment well. Because everything worked smoothly, it was easy for him to efficiently use VIP, and he used working patterns from the group *Using VIP fully*.

However, the correction of the second error did not work as he supposed it would, and Tomi became confused. The program was moving the integers in the array in a way that he did not expect at all. He followed the execution and found out that there

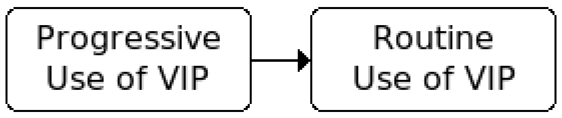


Figure 6 Phases Liisa reaches during her use story of VIP.

was an error, but then he just stoped working with VIP, and the reason of the error remained unclear for him (*Abandoning VIP*). He just sat down thinking and did nothing, so the interviewer tried to speed up the situation by giving a hint with programming. He switched his problem solving strategy to trial-and-error and kept using the working pattern *Abandoning VIP*. Every now and then, he is analyzed the execution of the program in VIP (the dotted rectangles of *Abandoning VIP* in Figure 2) but did not bring the analysis to the end using VIP like he was doing in the beginning of the session. He had somehow lost his understanding of his program and this affected the way he used VIP. In terms of the working patterns, he had moved from *Using VIP fully* to *Not able to use VIP*, which was a radical change.

He was lucky with his trial-and-error strategy and completed the program by chance. At the end of the session, he tested the solution by running it through in VIP using the working pattern *Static testing in VIP*. At this point, he had corrected all the errors, so it was easier for him to understand the state of the program again. Thus using VIP was easier for him again. However, his motivation to use VIP was not too high anymore since he just wanted to finish the assignment and observation session. This probably explains why he did not go back to *Using VIP fully* as in the beginning of the session. Instead he shifted to *Using VIP partly*.

The observation conducted as part of the interview conformed to the analysis of Tomi’s log files. In the beginning of the observation, he performed similar slow stepping through his program code in VIP as he had done during his programming sessions completing the programming projects. In the log file analysis, we assumed that he was using the step button slowly because he wanted to carefully follow how the execution proceeded in VIP, and he behaved the same way in the observation. However, it is noteworthy that such a deep analysis exploiting VIP was not a certainty for him. When loosing the track of his program he also had trouble to continue using VIP. As a consequence, he switched to other less effective ways of using VIP.

6.2 Liisa’s Story

Liisa was a first-year student of automation engineering. She had not studied programming before this course. She opened VIP for the first time when the first pre-assignment was available in VIP and used VIP in solving all seven pre-assignments the instructor had set up in VIP during the programming course. With a few exceptions, she opened VIP on Wednesdays, i.e. one evening before attending the exercise session. Her VIP sessions were long because she often applied the trial-and-error strategy when solving the programming problems, which is a time-consuming process.

Figure 6 illustrates how the use of VIP developed in Liisa’s story. She started working with VIP straight away, so the **introductory use of VIP** phase did not appear in her use story. She was a persistent user of VIP, despite the fact that she did not seem to concentrate on using the visualization features even if she claimed they were useful.

6.2.1 Liisa's Progressive Use of VIP

In the first half of her use story Liisa mainly followed the lecturer's instructions on VIP usage in weekly pre-assignments, but tested some other options of using VIP to some extent as well. The phase was identified as **progressive use of VIP** since even though Liisa's personal manners were not very strong, she did seem to be doing some testing to develop them.

She was learning to use the control buttons of VIP when she used the tool to learn programming. She tried adjusting the speed to be able to follow the visualization and performed some actions to test the controls a couple of times, but mostly, she concentrated on programming, applying the trial-and-error strategy.

In the first two sessions, Liisa did not show *high commitment to visualization* but did so in the third and fourth sessions. This differentiated this phase from the **routine use of VIP** phase that followed in her use story. Liisa's behaviour in sessions three and four are discussed below.

In the first part of the assignment in Liisa's third VIP session, she needed to explain what the initial version of the program code does in Finnish. Remarkably, Liisa started the program execution slowly and looked at it multiple times. We assumed that she executed it slower than usual because she wanted to answer the question about the program code. The trial-and-error strategy emerged again as she started to modify the program. The second part of the assignment was to modify the code, so we assumed that the trial-and-error strategy was related to this part of the assignment. After this session, she stated in the questionnaire that VIP was useful for her:

> *Liisa: Using VIP, I can understand how the program works and why it works correctly or incorrectly.*

The first part of the comment seems to be related to the initial part of the assignment, a purpose of which was to explain how the program works; the latter part, however, gives an impression that she also found VIP to be useful during the trial-and-error strategy, working on the second part of the assignment.

The only time when Liisa used VIP for some purpose other than pre-assignments was in her fourth VIP session, when she revised the course materials before the exam. This was also the only session in which she had constant *high commitment to visualization*. She executed multiple example programs in VIP using both the run and step functions and followed the execution carefully almost all the time. In this session she did not try to modify the programs so the trial-and-error strategy did not emerge.

It seems that in Liisa's normal VIP sessions the trial-and-error strategy prevented Liisa from following the visualizations carefully. Whenever her purpose was just to check whether the code adjustment worked, her motivation to follow how the execution proceeded was quite low. Therefore, when using VIP for other kinds of tasks, Liisa achieved *high commitment to visualization*. This demonstrated a certain development of using VIP in the first half of Liisa's use story.

6.2.2 Liisa's Routine Use of VIP

In the second half of Liisa's use story, the same forms of using VIP in every session begin to appear. She used only the run function when executing her programs and speed changes were very rare. In some sessions there was only one change of speed, and it usually meant turning it up. There were sessions in which she executed her program almost only with the maximum speed. In such cases, it was obvious that she was not following what was happening during the execution but was interested in the result only. Using a normal compiler would definitely have been much faster and more practical, thus it was doubtful whether using VIP in such a manner was of any benefit.

Next, we will provide a detailed description of one of such sessions of using visualization. Liisa started working on her pre-assignment by executing the initial program code with the initial speed setting of VIP; the speed was rather slow thus it was possible to follow the execution. Then, there was a thirty-minute break during which she wrote her assignment solution in the VIP code editor. She turned the speed up very high, but not to the maximum, and executed her program twice, editing a bit between the executions. She turned the speed up to the maximum before the next execution. During the following two hours, she performed a long session of trial-and-error programming and executed her program 24 times using the maximum speed only. This was so fast that she basically only looked at what was printed in the end. When the solution was finally correct, she turned the speed down and executes the final version of her program to see whether everything worked the way it should.

The only times Liisa lowered the speed enough to follow the visualizations was when she executed the initial program code to see how it worked and when she confirmed that the solution was correct. Hence, paradoxically, she followed the execution of correctly working program code only. If there was an error in the program, it would not have been possible to use visualization to detect the error. However, each time when answering weekly questionnaires, she chose the option “VIP was useful.”

6.2.3 Liisa’s Observation Session

In the observation session, Liisa’s task was to modify a program that handled an array of records. She chose to use VIP right in the beginning of the session. The program stored some values in the records in the array, made some calculations with these values, and stored the result in another position in the array. The task seemed to be difficult for her, and it took a long time for her to accomplish writing the first version of the solution. Thus, we were only able to observe her actions when she was correcting two different errors one after the other.

She executed her solution in VIP using the run function and watched what was happening in the screen. It appeared as if she was just sitting down and staring the screen, so the interviewer asked what she was seeing there, and she explained:

I look at [pointing at the code window] where it [the execution] reaches. And [pointing at the evaluation window] what it [the program] is doing.

This was an unusual way of explaining the purpose of VIP's windows since all of the windows–not just the evaluation window–offer a different view to “what the program is doing”. The comment reflects that for Liisa the evaluation window was especially important in following the programs execution.

Liisa found an error in the program when running the solution in VIP but instead of debugging the program in the window of VIP she moved to the code editor of VIP and started looking at the program code to figure out how she could correct the error. This working pattern was named *Abandoning VIP*. She behaved in a similar manner when correcting both of the errors.

In the log file analysis we perceived the contradiction that Liisa did not seem to follow the execution of the program in VIP but still claimed that VIP was useful for her when debugging the program. The observation showed that the assumption that she was not really following the execution was correct: she only followed until she found out that there was an error and stopped following after that. Also her strategy of using VIP in the observation was similar to what we detected in the log files: she repeated fast execution of the program in VIP, detecting that there was a problem, but not knowing the reason for it. This was only repeated twice during the observation but it was like a start of a trial-and-error programming session.

In the observation Liisa uses only one of the working patterns. It was from the group *Unfinished use of VIP*. We expected according to her log files that she was not able to use VIP to analyze the problems of her program which she also demonstrated here. However, the observation session did not really explain why Liisa was not using VIP to analyze the reasons of the errors in her program. She just moved from the VIP main window where the visualizations were exposed to the VIP code editor to think how she could correct the error. It seems like she just never had the idea that she could use VIP to also analyze the reason for the error.

6.3 Concluding Remarks

When following Tomi's and Liisa's stories of using VIP in the long term, it is easy to notice that their developments were quite opposite. Tomi was both developing his personal manners of using VIP and learning to be more fluent with its use whereas Liisa kept repeating the same use purposes of VIP with the same inflexible settings of the tool. Tomi was able to use VIP very fluently to follow and analyze the execution of his program. However, he did not work this way all the time. Liisa instead was mainly using VIP as a convenient way of tracing the values of variables via the evaluation window and very rarely used the visualization capabilities of VIP. These aspects of the stories are further analyzed in the following section where we interpret the results of this empirical study.

7 Interpretation with Activity Theory

In the Sections 4–6, we presented our results that capture general characteristics of how eight students engaged with VIP during an introductory programming course. In order to deepen our insights into understanding these eight students' VIP engagement and to generalize these results, we will interpret our results further in this section. For this purpose, we propose utilizing Activity Theory (AT) as a general framework for understanding the cognitive processes involved in supporting learning with program visualizations. Stemming from the work of Vygotsky (1978) in the 1920s, AT has been very influential in human-computer interaction as the focus in this field of research has shifted towards activities of people using technology over the past twenty years. It has also been used in education research, for example by Lim and Hang (2003), Blin and Munro (2008), and Karasavvidis (2009), but it is practically unnoticed in computer science education (apart from Berglund (2005)) and unknown in the field of program visualization tools for educational purposes. AT provides different concepts that describe how individuals interact with the environment through the use of tools or social entities that mediate the corresponding interaction and engagement.

In the next subsection, we introduce AT and its main concepts. Then, in Subsection 7.2, we will interpret our results with the introduced concepts. In Subsection 7.3, we will suggest hypotheses about student engagement with program visualization software for learning programming.

7.1 Activity Theory

Activity Theory (sometimes called Cultural-Historical Activity Theory) is a psychological theory about the relationships between human beings and their goal-directed activities. The theory has its roots in Russian cultural-historical psychology from the 1920's by Vygotsky, and was developed further by Leontiev and others in the following decades, see for instance Engestrom (1990); Bødker (1989); Kaptelinin and Nardi (2006). There are many similar theories that have been elaborated over the last three decades that have built on these historical foundations that go by the names "situated learning" (Lave and Wenger, 1991), "sociocultural learning" (Rogoff, 2003; Wertsch, 1993), "distributed cognition" (Salomon, 1993), among others. For our purposes in this paper, we focus mostly on the basic introduction to AT in Human-Computer-Interaction provided by Kaptelinin and Nardi (2006).

7.1.1 Mediated Activities

AT emerges from the assumption that people are goal-directed and that it is through activities that individuals try to achieve their goals. Though some activities are "visible" like walking or dancing, others are "invisible" because they happen inside the body like thinking or reasoning. With regard to this, AT differentiates between internal and external activities. The traditional cognitivist notion of mental processes corresponds to what is referred as internal activity in AT. AT emphasizes that internal and external activities are highly connected to each other and cannot be analyzed separately or in isolation from each other.

According to AT, internal and external activities carried out by individuals are mediated by signs and tools that influence individuals' relationships with the world. AT posits that an individual interacts with the environment not directly but by using tools that are meaningful and supportive for the incorporated activities. Language, algebraic notations, and maps are classical examples of such "signs and tools," as are many physical objects such as hammers, coffee machines, or scissors (Kaptelinin and Nardi, 2006, p. 42ff).

Saying that activity is mediated makes the assumption that action cannot be separated from the milieu in which it is carried out (Wertsch, 1993, p. 18). For example, pencil and paper, an abacus, and an electronic calculator all are different tools for summing a set of numbers. The related external activities with these tools are different as are the internal activities which are inextricably bound to the particular tool a person chooses to use for summing numbers. Internal activities or mental processes are derived from external activities, and both are mediated by tools. As Kaptelinin and Nardi point out, "[t]he structure of a tool itself, as well as learning how to use a tool, changes the structure of human interaction with the world" (Kaptelinin and Nardi, 2006, p. 56). Signs and tools, then, are not incidental to activity, nor do they simply enable it. Rather, they are inseparable from activity, serving as the point of contact between person and world.

7.1.2 Internalization

Tools mediate activities because they include specific knowledge and skills for using them. For example, the specific form of a hammer embeds knowledge about the ergonomic properties of the human body as well as physical properties of the external world, such as force and momentum. Nailing activities that are mediated by a hammer will be shaped through the tool's incorporated knowledge and skills. In other words, our understanding of nailing and how to accomplish this activity was mediated by the hammer when we learned to use this tool.

Internal–External Dimension

The process of mediating an activity by a tool is called *internalization*, and this relates to both mediated internal as well as external activities. In the process of internalization, the tool's embodied knowledge, concepts and understandings are internalized into a person's activities. Let's consider for example the task of finding the way in a new city using a map. In the beginning, the mediation is highly visible because it incorporates first the external activity of looking at the map, reading its symbols and pictures, and connecting this information to the environment of the city. This is what Vygotsky distinguishes as *external mediation* (and the tool's function as *external mediator*): the map mediates an individual's external and internal activities. After a while when a person *internalized* the map's concepts, he or she will stop to use the map and does not practice the external activities with this tool anymore. He or she will be able to move through the city due to the internalized concepts and understanding developed through external mediation by the map. But his or her internal activities will still remain mediated because they have been mediated by the map and through the external activities. This is what Vygotsky distinguishes as *internal mediation* (and the tool's function as *internal mediator*). In summary, a tool mediates internal and external activities in the process of internalization (Kaptelinin and Nardi, 2006, p. 43ff).

Individual–Social Dimension

Mediation includes not only the introduced internal-external dimension; tools, which mediate activities, do not simply arise de novo in the hands and minds of individual actors. Rather, they are provided to individuals by the surrounding culture, accreting over time and, passed from one generation to the next. Therefore, tools represent socially distributed cultural entities that implicitly embed collective knowledge of their use in context. Cultural practices of tool use evolve in tandem with the evolution of the tool. For example, just as the materials and form of hammers have evolved over time (Basalla, 1989), so have hammer-mediated activities changed; if the tool changes, so must its use.

According to Vygotsky, an individual typically first performs a particular tool-mediated activity in collaboration with or guided by others that already have certain tool-using skills. With gained experience, the individual transforms activities from what was initially social to one that are performed individually (Vygotsky, 1978). This is an active transaction from the social to the individual dimension of activity, which is part of the internalization process, as well.

7.1.3 Automatization

So far, we have discussed the roles activities play in the processes of mediation and internalization. In this subsection, we will focus on activities themselves. In a further development of AT, Leontiev (1981) introduced a conceptualization of "activity" itself distinguishing it as a composition of actions and operations (Kaptelinin and Nardi, 2006, p. 62ff). In this hierarchical conceptualization, activities are on the highest level, include a person’s motive, and are composed of actions. An action has a specific goal a person wants to accomplish and of which he or she is consciously aware of. Every action consists of operations that are automated steps that occur unconsciously: ”Operations are sensorimotor units that a human being performs in a specific situation, without consciously thinking of them, to perform the actions that he or she is consciously aware of” (Bødker, 1989, p. 177). In the context of learning programming, a student’s activity might involve solving a weekly programming assignment because he or she wants to pass the final exam and is also interested in programming. This activity would contain many different actions. Related to debugging a program code, an action might be finding an input that makes the program behave in an incorrect way; meanwhile, operations might include pressing a certain button in the user interface of the programming environment.

In the beginning, every operation is an action performed very consciously. By practicing a certain action long enough, a conscious action transforms into an operation which, in AT, is called *automatization*. Bødker (1989) describes the learning process of handling actions and activities with regard to Dreyfus and Dreyfus (1986). According to the latter, a novice student will focus consciously on actions and being aware of every little step he or she performs. It might be difficult in this stage to distinguish the different actions, their goals and how they are connected to each other and the overall activity and its motive. In the next steps, a more advanced student “moves from a set of situation-specific operations based on situation-specific practical experience to a wider variety of operations” (Bødker, 1989, p. 179). In time, first low-level actions transform to automated operations and higher-level actions are performed. Finally, the expert student “can conduct the activity operationally with very high-level actions”.

In the next sections, we will relate the introduced AT concepts with learning programming and the results of our study presented in the last three sections.

7.2 Interpretation of the Results with AT

With regard to Activity Theory, learning programming in an introductory programming course at TUT represents a specific world where our students interact and where their activities take place. This world includes not only the environment like the classroom or the library, but specifically the interaction with other individuals that are met in this world such as professors, teaching assistants, or other students. The specified goals would be for example regularly attending the programming course, doing programming assignments or homework, as well as passing tests or exams (see Section 3.1).

The results described in the previous sections contain four use phases and four groups of working patterns. How can we understand them, the students’ engagement with VIP using AT? In a further process of interpretation, we became aware that every piece of our results might be representing different aspects or stages of an internalization process where VIP can be understood as a tool that mediates students’ activities and actions of learning programming. In the next subsections, we will explain this interpretation, relating our results to the introduced concepts of AT.

7.2.1 The Process of Internalization in the Working Patterns

The grouping of the working patterns presented in Section 4.3 emphasizes the different levels of visualization engagement we observed in the patterns and gives an overall understanding of the different ways students can take advantage of visualizations. Looking at these four groups from the perspective of AT, we interpret them as different stages of using VIP as a mediator.

The group *Using VIP fully* covers patterns where students execute their program in VIP following the visualizations during the program execution (example in Section 4.1.1). Here, the students use all control functions of VIP and multiple different windows. Students analyze the program behaviour, exploiting the visualizations carefully and intensively. This helps them to understand the behaviour of their program as well as the programming concepts involved. These observable, external activities are clearly mediated by VIP and its visualizations. According to AT, we can therefore assume that the corresponding internal activities were mediated by these external activities and VIP, as well.

The group *No need for VIP* captures the situation where the students are able to create a correct program and do not need any support from VIP. They are able to understand what happens during the run of their program without using VIP. We know that these students were using VIP intensively earlier; therefore we can assume that they reached here another stage of the internalization. Since they do not need VIP anymore as an external mediator, we assume that VIP’s visualizations became part of their internal activities. Internalized visualizations might still guide their programming activities, but external engagement with the tool is not needed anymore.

The group *Unfinished use of VIP* captures the situation where the students are unable to use VIP’s visualizations to support them in accomplishing the programming task. Altogether, the students showed no real interaction with the visualizations. We interpret this as a stage where VIP’s visualizations do not function as an external mediator of the students’ programming activities for some reason. In comparison to the group *No need for VIP*, there can also be no internal mediation since the programming problem is left unsolved.

The group *Using VIP partly* covers patterns where the students incorporate a limited set of VIP’s functions. The students can, for instance, exploit only the static presentations of the visualizations (example in Section 4.1.2) or perform superficial output analysis using VIP just like any other programming environment that executes a program and displays its output. Can we assume that these programming activities are mediated by VIP? The students are able to analyze their program, so it is possible that VIP mediates their internal activities in some way. But, the students show a limited amount of external activities mediated with VIP. One explanation could be that this group of patterns captured only the beginning of the internalization process. However, it is also possible that it represents fruitless approaches for possible further internalization. An obvious interpretation of this group is not possible based on our results and must, therefore, be left as future work.

Group	Stage of internalization
No need for VIP	Internal mediation: no external activity nor the tool is needed
Using VIP fully	Internal and external mediation: external activity and visualizations mediate internal activities
Using VIP partly	Beginning of mediation: remains open
Unfinished use of VIP	No mediation

Table 3 Interpretation of working pattern groups using the concept of internalization.

In Table 3, we summarize our interpretation: the different groups correspond to different stages of the internalization process.

7.2.2 The Processes of Automatization and Internalization in the Use Phases

The four phases presented in Section 4.3 describe certain developments in students' long-term engagements with VIP. This development could start with **introductory use of VIP** and become more intensive in the **progressive use of VIP**. The **creative use of VIP** represented the strongest personal development in the engagement with VIP, while **routine use of VIP** indicated that the use of VIP continued despite the engagement with the visualization not being strong. We interpreted these four phases as different stages of the *automatization* process. We also see here further indications for the internalization process, specially related to the social–individual dimension of activities. We will explain this in detail below.

The Process of Automatization

In the **introductory use of VIP** phase, a student's goal is to get to know the user interface of VIP and its components. Moving to the next use phases of VIP, the student's goal focuses on solving programming tasks. By that time, the fluency of using VIP's controls improves, even if the goals stay the same. This could be interpreted as an automatization process. Tomi's story gives an example of a successful automatization process. In the beginning, he was only using the step-function for executing his programs. Then, he switched to only using the run-function and testing its speed settings. As he reached the phase **creative use of VIP**, the use of VIP controls had become fluent for him and he was switching between the run and the step buttons and changing the speed of the program execution in order to observe exactly those parts of the code he was interested in. We interpreted that the first actions with VIP are transformed to operations. Liisa's story, on the contrary, exemplifies a case where automatization did not occur. She did not even try to use the VIP control settings to support her programming activity with the visualizations in the later part of her use story. Non-automated use of the control-functions is difficult when focusing on the same time on the execution of a larger program.

In Liisa's story, we can find two possible reasons for the delayed automatization process. First, she had not paid special attention to learn how to use the controls as there was no **introductory use of VIP** phase in her story. Second, she worked using the trial and error strategy mainly and thus did not particularly concentrate on the visualizations. These circumstances did not support or encourage her to practice the use of the controls.

Therefore, the automatization of low-level actions with VIP seems to be an important basis for the benefit of using VIP and the internalization process involved.

The Process of Internalization

In their first VIP sessions, the students mainly followed the instructions of the lecturer, who set the direction of the use session explaining why and how VIP should be used. However, the responsibility to accomplish this was left to the student. Here, some students rapidly transformed the activity introduced by the teacher to individual forms of using VIP. Tomi's story is an example: first, he looked at the assignment set up in VIP but decided not to work on it in VIP, then he developed new use purposes for VIP which he found more interesting than those recommended by the instructor. The **progressive** and **creative use of VIP** together represent a development where students move from first interactions with VIP being instructed by the teacher towards individual activities. Other students used VIP without experimenting, and their activities remained the same as those introduced by the teacher; they did not transform into individual activities. Liisa's story exemplifies this. The **routine use of VIP** is like the counterpart of the development in the **progressive** and **creative use of VIP**.

With regard to the individual–collective dimension of activities, some of the students transformed the introduced collective activity into an individual activity, indicating that they passed the process of internalization. The concept *high commitment to the visualizations* captures an engagement with VIP where students execute their program very slowly following the visualizations carefully. We can interpret this action as being mediated by VIP. As explained in Subsection 5.3.1, this concept occured mainly in the **Creative use of VIP** phase. This conforms to the idea that students were going through the process of internalization during their use stories when they shifted towards individual activities with VIP.

7.2.3 Discussion of the Interpretation

With regard to what was presented in the last two subsections, we interpret that Tomi was able to use VIP as a mediator while Liisa, instead, only used VIP as a compiler. The most interesting question is *why* some students were able to use VIP as mediator and enter the process of internalization and others were not?

Using Tomi and Liisa as an example, we can argue that these two students were at different stages of the internalization process. Tomi had proceeded to the phase **Creative Use of VIP** where he used VIP according to his own needs in learning and designed his own use purposes. VIP was an external mediator for him, and he was able to use the working patterns from the group *Using VIP fully*. On the contrary, Liisa did not develop such a beneficial way of using VIP and ended up in the phase **Routine Use of VIP** using working patterns from the group *No mediation*. We argue that her internalization process of VIP was delayed exactly like her automatization process was delayed.

One possible reason for Tomi's success and Liisa's failure to enter the internalization process could be the stimulating influence of the automatization process of the VIP control functions. In the beginning of the use story, Tomi performed an **introductory use of VIP** phase whereas Liisa did not. Effort dedicated to learning the basic uses of the tool could pay back later in the ease of concentrating on what is essential−in this case the internalization process.

An additional explanatory factor for the divergent behaviour of these two students could be their gender. Jones et al. (2000) reported that female students tend to carefully follow teachers directions when using science equipment and tools. They found that female students play and tinker only little with the distributed material. Instead male students use tools in inventive and exploratory ways. Similar behaviour has also been detected when studying the use of software tools (Burnett et al., 2011) and programmers' debugging behaviour (Beckwith et al., 2006). Playing and tinkering around may represent important factors for action automatization and individual forms of engagement, which both are first steps toward higher-level actions and internalization.

Another interesting question is why Tomi did not use VIP as a mediator the entire time. His VIP log file and the beginning of the observation session clearly show that he was able to use VIP as a mediator but in the end of the observation he did not do it anymore. It seems that being able to use VIP as an external mediator does not guarantee that one will do so constantly. We see this as an understandable occurrence. There are multiple external factors that influence the learning situation and can distract the learner. In such a case, the student might end up using a working pattern from the group *No mediation* even if he/she was also able to use the working patterns from the higher levels of the grouping.

It is also noteworthy that this interpretation only explicates how students engaged with VIP when learning programming. We do not claim that using VIP makes learning programming effective. For example, it is obvious that the behaviour described in the group *Using VIP partly* demonstrates an inefficient way of using the visualization tool. Still, it does not have to be an inefficient way of learning programming. Working with VIP this way can still lead to positive learning results even if the visualizations are neglected. After all, the most important goal is that the students learn programming, not that they use the visualizations.

7.3 Hypotheses about Student Engagement with Program Visualizations

In this section, we will help to summarize the results' interpretation with AT by presenting two research hypotheses and will provide a discussion of the implications for further research in the field of program visualizations in education.

7.3.1 The Educational Effectiveness of a Visualization Tool

Our empirical results support the interpretation that a program visualization tool can be seen as a tool that mediates students' programming activities. AT lends support to the view that the mediator is advantageous for its user only during the process of internalization. When internalization occurs, students are able to perform programming activities without the tool because it has become an internal mediator and is not physically needed anymore. This leads to our first hypothesis:

> ***The visualization tool is educationally effective mostly during the process of internalization when serving as a mediator for students' programming activities.***

According to our hypothesis, the real benefit of visualizations occurs during internalization when visualizations are internalized to the student's mental model of programming concepts, requiring no further use of the physical tool. This leads to a possible explanation of why students do not use visualization tools although their educational impact has been proven in research studies. The students that have already gone through the internalization process do not need the visualization tool as a mediator anymore so it is natural that they do not use it. In addition, for some students it is possible that the internalization process with the tool has not been stimulated enough and they tend to use the tool only from time to time, not knowing how to support their programming activities with it.

This was also the answer to our second research question. The students benefited from using the visualization tool during the internalization when the tool served as a mediator for students' programming activities.

To date, many research studies about the educational effectiveness of program visualization tools focus on students' performance at the end of a certain time period when students were supported using a tool for programming activities. In such studies, students were probably more stimulated in using the visualizations than in normal programming courses. As a consequence, more students might have entered the internalization process which would explain the tested educational impact of the tool in these studies. Also, there are studies investigating the educational effectiveness of program visualization tools in a single use session as a part of a controlled experiment. Such research setup totally neglects the long-term nature of the internalization process. Considering the students' current stage of the internalization process can be helpful in better understanding the "markedly mixed" results of such studies. Future research studies should take the characteristics of the internalization process into account when investigating and testing the benefit of a visualization tool. Vygotsky also introduced a method called *double stimulation* (Kaptelinin and Nardi, 2006, p. 43-45) and the concept of the *zone of proximal development* (Kaptelinin and Nardi, 2006, p. 48ff) both offering potential alternative research approaches for studying the use of visualizations in the future.

7.3.2 Possible Reasons behind the Internalization Process

Our empirical results support the view that the use phase **Creative use of VIP** captures students' programming activities and actions that were mediated by the visualizations. We can even say that this use phase was the central stage of students' internalization process. In this use phase, students achieved a *high commitment to the visualizations* and used the working patterns from the group *Using VIP fully*.

In order to reach the **Creative use of VIP**, students must master the control functions of the visualization tool and developing personal manners of interacting with the tool. This leads to our second hypothesis:

> ***To be able to achieve a stage where the process of internalization is possible, students must first engage with the visualization tool deep enough in order to automatize lower-level actions with the tool and second, transform collective activities of using the tool to individual interaction with the visualizations.***

This hypothesis suggests directions in course setup and instructions on how visualization tools should be incorporated into teaching programming.

- Surely, it is not enough to introduce the visualization tool once and suppose the students will use it for the rest of the course. Instead, the visualization tool should be promoted by the teacher more intensively over a longer period of time in order to allow enough time for the automatization and internalization processes to happen. Also, using different kinds of visualization tools or materials for each topic discussed in a programming course is not the same as using one tool for a longer period. That would mean that the students had to go through the internalization process with each of the tools to engage with them.
- Students should be encouraged to tinker and play with a visualization tool instead of just following the teacher's instructions on how to use the tool for a concrete programming assignment. This would help students discover their personal preferences on using the tool and, thus, stimulate the transformation to the individual from the collective activity.
- To help also the students who do not tend to tinker, mediation should be stimulated with various types of activities and actions addressing different learning types among students. For example, the assignment of explaining what happens during program execution lead Liisa to experience the high commitment to the visualization when using VIP. Her activities with VIP would have probably been more successful with the stimulation of a cascade of such assignments because experiencing the high commitment to the visualization in multiple VIP sessions could have lead her to enter the internalization process.

These possible recommendations for teaching programming with visualization tools show that the tool must be treated like any other teaching material. It must be carefully integrated in the course to provide students an environment where the tool's role as a mediator is volitionally incorporated.

7.3.3 Relating the Interpretation to Other Work on Visualizations

When looking at previous work carried out on visualizations (see Section 2.2), we can find multiple possible connections to our interpretation stating that a program visualization tool can be understood as a tool that mediates students' activities and actions when they learn programming.

Moreno and Joy (2006) conclude that students had difficulties in understanding the Jeliot 3 animations. They reason that "The transfer of knowledge from the tool to the student is not successful. Even students who have been explicitly explained the meaning of the animations have problems understanding or applying this later." This reasoning relies on different kind of understanding of learning than the constructivist learning theory we have used as our conceptual framework. Looking at the phenomenon from the AT's point of view, we could also explain that when cognitively constructing their individual programming concepts and skills, the students failed to use the visualization tool as a mediator. Explicitly explaining the meaning of the animations for the students once is not enough to replace the internalization process. Thus, the students failed to use the tool as a mediator.

Lattu et al. (2000) reported that "Jeliot is especially suitable for beginners" and that the students "did not see the visualization relevant for more advanced programmers." We can use the internalization process as a possible explanation for this phenomenon too. According to our interpretation, the benefit of using visualizations occurs during the process of internalization. If we interpret that the beginners were at the state of going through the internalization process whereas the more advanced programmers had already passed it, that explains why the beginners benefited from the use of the visualizations and the more advanced programmers would not benefit from using it.

Kannusmäki et al. (2004) conclude that “students are very sensitive about changing the tool they have used”. We already handled this topic when discussing our second hypothesis. Starting to use a new tool always means going through the process of internalization again. Previous experience of using a similar tool of course helps getting started since learning is a constructive process. Nevertheless, similarly to the explanation of the results by Moreno and Joy (2006), the internalization process cannot be simply replaced by an outsider’s explanation of what the visualizations mean.

It appears that AT also gives new insights into the previous work carried out on visualizations. This supports our interpretation and gives reason to assume that it could also be applicable in the wider visualization research.

8 Conclusions

We applied qualitative analysis methodologies to reveal how students voluntarily and regularly engaged with the visualization tool VIP on their own in the long term. The results are working patterns that describe the use in single use situations and phases that describe the use in the long term. These empirical results were further interpreted using AT as a framework. Based in the interpretation, we propose two research hypotheses that constitute the first step of building a theory about how students engage with visualizations. The theory building is left as future work, but the raw research hypotheses already set guidelines for future research on visualizations and teaching with visualizations. Literature related to AT proposes possible methodologies for studying the internalization process of the visualization tools further. For example, the double stimulation method introduced by Vygotsky (Kaptelinin and Nardi, 2006, p. 43-45) could be used for verifying the research hypotheses in the future. Also other smaller topics for further investigation arose during the interpretation; the third row in Table 3 summarizes behaviour whose exact explanation was not covered here.

The research question 1 concerning how students engaged with VIP in the long term was answered by the working patterns and the use phases. The working patterns describe the forms of the engagement on the level of one use session of VIP, and the phases show possible different developments of the engagement during a whole programming course. The research question 2 regarding how students benefited from the use of VIP was answered by our research hypotheses. The main message encapsulated is that the students’ engagement with the visualization tool and the benefit they gained from using it are interrelated with using the tool as a mediator during the internalization process of programming concepts.

The aim of this study was to map the territory and hence the research questions were open. As a result, this study became an eye-opener for the researcher and teacher of the programming course (author Isohanni). A true understanding on how students work on their own is valuable for a teacher. In this case, the teacher expected that the frequent, voluntary users of VIP would use it pretty much as instructed in the lecture since that seemed to be the only beneficial way to use the tool. Surprisingly, it turned out that even the frequent users of VIP have impractical working patterns like *Static debugging in VIP* and *Abandoning VIP*, and use VIP incoherently like during the phase **Routine use of VIP**. Being an expert in programming, it is easy for the teacher to neglect the novice programmers’ difficulties with the tools. Descriptive studies, like this one, reveal the reality of learning programming, and the theory connection of the study gives a strong basis for the findings so they can also be useful for understanding similar phenomena in other circumstances.

References

Ahoniemi, T. and Lahtinen, E. (2006). Visualizations in Preparing for Programming Exercise Sessions. In Proceedings of the Fourth Program Visualization Workshop, pages 54–59, Florence, Italy.

Basalla, G. (1989). The Evolution of technology. Cambridge University Press.

Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., and Cook, C. (2006). Tinkering and gender in end-user programmers' debugging. In Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI '06, pages 231–240, New York, NY, USA. ACM.

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

Ben-Ari, M., Bednarik, R., Levy, R. B.-B., Ebel, G., Moreno, A., Myller, N., and Sutinen, E. (2011). A decade of research and development on program animation: The jeliot experience. Journal of Visual Languages & Computing, 22(5):375 – 384.

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

Berglund, A. (2005). Learning computer systems in a distributed project course: The what, why, how and where. Doctoral thesis, Uppsala Dissertations from the Faculty of Science and Technology nr. 62, Acta Universitatis Upsaliensis, Uppsala University, Sweden.

Blin, F. and Munro, M. (2008). Why hasn't technology disrupted academics' teaching practices? Understanding resistance to change through the lens of activity theory. Computers & Education, 50(2):475 – 490.

Bødker, S. (1989). A human activity approach to user interfaces. Human - Computer Interaction (Mahwah), 4(3):171–195.

Burnett, M. M., Beckwith, L., Wiedenbeck, S., Fleming, S. D., Cao, J., Park, T. H., Grigoreanu, V., and Rector, K. (2011). Gender pluralism in problem-solving software. Interacting with Computers, 23(5):450 – 460. ¡ce:title¿Feminism and HCI: New Perspectives¡/ce:title¿.

Corbin, J. and Strauss, A. L. (2008). Basics of qualitative research, 3rd Edition. Sage Publica- tions, Thousands Oaks, California.

Dreyfus, H. and Dreyfus, S. (1986). Mind over machme – The power of human intuition and expertise in the era of the computer. Basil Blackwell, Glasgow.

Engeström, Y. (1990). Learning, working and imagining: Twelve studies in activity theory. Orienta-konsultit, Helsinki.

Fincher, S. and Petre, M. (2004). Computer Science Education Research. Taylor and Francis, The Netherlands, Lisse.

Hundhausen, C. (2002). Integrating algorithm visualization technology into an undergraduate algorithmscourse: ethnographic studies of a social constructivist approach. Computers & Education, 39(3):237–260.

Hundhausen, C. D. and Brown, J. L. (2007). What you see is what you code: A live algorithm development and visualization environment for novice learners. Journal of Visual Languages & Computing, 18(1):22 – 47.

Hundhausen, C. D. and Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a cs 1 studio experience: An empirical study. Computers & Education, 50(1):301 – 326.

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

Illeris, K. (2002). The Three Dimensions of Learning. Krieger Publishing Company, Malabar, Florida.

Isohanni, E. and Knobelsdorf, M. (2010). Behind the curtain: students' use of VIP after class. In ICER '10: Proceedings of the Sixth international workshop on Computing education research, pages 87–96, New York, NY, USA. ACM.

Isohanni, E. and Knobelsdorf, M. (2011). Students' long-term engagement with the visualization tool VIP. In Proceedings of the 11th Koli Calling International Conference on Computing Education Research, Koli Calling '11, pages 33–38, New York, NY, USA. ACM.

Jones, M. G., Brader-Araje, L., Carboni, L. W., Carter, G., Rua, M. J., Banilower, E., and Hatch, H. (2000). Tool time: Gender and students' use of tools, control, and authority. Journal of Research in Science Teaching, 37(8):760–783.

Kannusmäki, O., Moreno, A., Myller, N., and Sutinen, E. (2004). What a Novice Wants: Students Using Program Visualization in Distance Programming Course. In Proceedings of the 3rd Program Visualization Workshop, Report CS-RR-407, Department of Computer Science, University of Warwick, UK, pages 126–133, UK.

Kaptelinin, V. and Nardi, B. A. (2006). Acting with Technology: Activity Theory and Interaction Design. MIT Press.

Karasavvidis, I. (2009). Activity Theory as a conceptual framework for understanding teacher approaches to Information and Communication Technologies. Computers & Education, 53(2):436 – 444.

Katz, I. R. and Anderson, J. R. (1987-1988). Debugging: An analysis of bug-location strategies. Human-Computer Interaction, 3:351–399.

Kehoe, C., Stasko, J., and Taylor, A. (1999). Rethinking the evaluation of algorithm animations as learning aids: An observational study. International Journal of Human-Computer Studies, 54:265–284.

Korhonen, A., Malmi, L., Myllyselka, P., and Scheinin, P. (2002). Does it make a difference if students exercise on the web or in the classroom? In Proceedings of the 7th annual conference on Innovation and technology in computer science education, ITiCSE '02, pages 121–124, New York, NY, USA. ACM.

Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A., and Malmi, L. (2005). Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system Trakla2. Informatics in Education, 4:49–68.

Lahtinen, E. (2006). Integrating the Use Of Visualizations to Teaching Programming. Proceedings of the conference Methods, Materials and Tools for Programming Education, pages 7–13.

Lahtinen, E., Ahoniemi, T., and Salo, A. (2007a). Effectiveness of integrating program visualization to a programming course. In Proceedings of The 7th Koli Calling Conference on Computer Science Education.

Lahtinen, E., Jarvinen, H.-M., and Melakoski-Vistbacka, S. (2007b). Targeting program visualizations. SIGCSE Bull., 39(3):256–260.

Lattu, M., Meisalo, V., and Tarhio, J. (2000). How a visualization tool can be used — evaluating a tool in a research & development project. In Proceedings of the 12th Annual Conference on the Psychology of Programming Interest Group, pages 19–32.

Lave, J. and Wenger, E. (1991). Situated Learning: Legitimate Peripheral Participation. Cambridge University Press.

Leontiev, A. N. (1981). Problems of the Development of the Mind. Progress, Moscow.

Lim, C. P. and Hang, D. (2003). An activity theory approach to research of ICT integration in Singapore schools. Computers & Education, 41(1):49 – 63.

Lincoln, Y. S. and Guba, E. G. (1985). Naturalistic Inquiry. Sage Publications, Inc., Newbury Park, California.

Lönnberg, J., Malmi, L., and Ben-Ari, M. (2011). Evaluating a visualisation of the execution of a concurrent program. In Proceedings of the 11th Koli Calling International Conference on Computing Education Research, Koli Calling '11, pages 39–48, New York, NY, USA. ACM.

Mayring, P. (2000). Qualitative Content Analysis. Forum: Qualitative Social Research [Online Journal], 1(2):Art. 20.

Moreno, A. and Joy, M. S. (2006). Jeliot 3 in a Demanding Educational Setting. In Proceedings of the Fourth Program Visualization Workshop, pages 51–59, Florence, Italy.

Naps, T., Rossling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velazquez-Iturbide, J. (2003). Exploring the role of visualization and engagement in computer science education. SIGCSE Bulletin, 35(2):131–152.

Rajala, T., Laakso, M.-J., Kaila, E., and Salakoski, T. (2007). VILLE - a language-independent program visualization tool. In Proc. Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland. CRPIT, 88. Lister, R. and Simon, Eds., ACS. 151-159.

Rogoff, B. (2003). The Cultural Nature of Human Development. Oxford University Press,. Romero, P., du Boulay, B., Cox, R., Lutz, R., and Bryant, S. (2005). Graphical visualisations and debugging: A detailed process analysis. In Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group, pages 62–76.

Salomon, G. (1993). Distributed Cognition: Psychological and Educational Considerations. Cambridge University Press.

Shavelson, R. J. and Towne, L. (2002). Scientific Research in Education. The National Academies Press.

Stasko, J. T. and Hundhausen, C. D. (2004). Algorithm Visualization. In Fincher, S. and Petre, M., editors, Computer Science Education Research, pages 199–228. Taylor and Francis, The Netherlands, Lisse.

Strauss, A. and Corbin, J. (1995). Basics of Qualitative Research, 2ed edition. Sage Publications, Newbury Park.

Sorva, J. (2012). Visual Program Simulation in Introductory Programming Education. Department of Computer Science and Engineering. Aalto University publication series DOCTORAL DISSERTATIONS 61/2012. Helsinki, Finland.

Urquiza-Fuentes, J. and Velazquez-Iturbide, J. A. (2009). A survey of successful evaluations of program visualization and algorithm animation systems. Trans. Comput. Educ., 9:9:1–9:21.

Virtanen, A. T., Lahtinen, E., and Jarvinen, H.-M. (2005). VIP, a visual interpreter for learning introductory programming with C++. Proceedings of The Fifth Koli Calling Conference on Computer Science Education, pages 125–130.

Vygotsky, L. S. (1978). Mind in Society: The Development of Higher Psychological Processes. Harvard University Press.

Wertsch, J. V. (1993). Voices of the Mind: Sociocultural Approach to Mediated Action. Harvard University Press.