ABSTRACT

Previous visualization research has focused mainly on specific tools. This paper presents an alternative approach that focuses on the students and what use they typically make of the tool. Taking students who voluntarily and regularly use the program visualization tool Visual InterPreter (VIP) for their programming assignments, we conducted a qualitative empirical study to observe how these students go about solving assignments with VIP. In the data analysis, we reconstructed students’ usage activities with VIP and their successful as well as inconclusive use of strategies, which we then conceptualized to students’ working patterns.

Categories and Subject Descriptors

K.3.2 [Computing Milieux]: Computers and Education—Computer and Information Science Education

General Terms

Human Factors

Keywords

Program visualization, Qualitative student-oriented research

1. INTRODUCTION

Every semester more than 400 students attend the introductory programming course at Tampere University of Technology (TUT). The course has high dropout rates, which are mainly caused by students’ learning difficulties. To better support the students’ learning progress, the program visualization tool Visual InterPreter (VIP) was developed and has been used in the course since 2004 [15].

While the majority of students in the course do not use VIP regularly, there is a group of students that use it enthusiastically for their programming assignments. We observed that the students who make regular use of VIP are less likely to drop out of the course [7]. Although it is unclear whether using VIP leads to learning success or whether successful students choose to use VIP, the finding does indicate a certain coherence. Without focusing on a potential causality, it is obvious that from the student perspective there must be a benefit in the use of VIP; otherwise students would not use it. However, we do not know the nature of the benefit or why and how these students use VIP. Consequently, we do not know whether other students might also benefit from using VIP.

The research to date on VIP has focused on establishing its general educational effectiveness. However, such a research approach is general, tool-oriented, and fails to recognize the different ways in which students respond to VIP. A simple yes/no question about a phenomenon as complex as learning is almost impossible to answer, because students’ learning success depends not only on the way the learning environment is designed but on many additional factors such as students’ interests, preferences, and learning styles. We argue that a student-oriented research approach that conducts a detailed investigation into students’ use of VIP will provide more sustainable information than the general results obtained so far. Students respond differently to teaching environments and what helps one student does not necessarily help another. Yet information about students’ usage should help to improve both teaching and the design of VIP, which should in turn support students’ learning and lead to lower dropout rates.

According to Hundhausen et al. [3], visualizations in general have not succeeded in becoming a part of mainstream programming education. As one solution to this problem, Stasko and Hundhausen [13] suggest that further development of visualization tools should be based on student-oriented research instead of the technical visions of the developers or programming experts. Researchers should study how students use visualizations, and should develop tools and materials according to the discovered needs. For a more detailed introduction with a special focus on related work and literature on visualization research, see Lahtinen [6].

In this paper we present a qualitative empirical study whose purpose is to learn what frequent users of VIP are actually doing with it. We focus on students who voluntarily and regularly use VIP for their programming assignments. A research setting was created to observe students’ activities and interview them about this while they were solving typical programming assignments with VIP (see Sections 2 and 3). In the data analysis, we reconstructed students’ usage activities with VIP and their successful as well as inconclusive strategies of use; these we conceptualized to students’ working patterns.
working patterns (see Section 4). Then they were compared to the way VIP was introduced during the course and connected to suggestions for teaching and design of visualization tools (Section 5). The paper concludes with ideas for further research (Section 6).

2. THE RESEARCH APPROACH

We are interested in how students go about solving typical programming assignments using the visualization tool VIP. What are they doing with VIP when the lecture is over? Do they use it in the way they were shown during the lecture? Our research objective is to see behind the curtain and observe students’ activities with VIP. Because this is the first student-oriented VIP study, and because we know very little about what is relevant and important in this context, our research starts with open questions.

We follow the Natural Inquiry paradigm as explained by Guba&Lincoln ([8] p.36ff) and in the Grounded Theory approach by Strauss and Corbin [2] (though without using the methodological implementations of Grounded Theory). We approach the research field in an explorative study where we observe and interview students while they are solving typical programming assignments using VIP. This means that we work with single cases and analyze their similarities and differences in order to develop a data-driven conceptualization of the cases. Therefore we rely on the following qualitative criteria: authenticity by reflecting the researcher’s subjectivity and preconditions; indication of the procedure’s appropriateness by describing and discussing relevance and limitations of the research approach, choice of methods or sampling strategies; and data-driven theory-formation using codified methods and checking internal coherence [8, 14].

There are many different theories about novice programming, debugging strategies with visualizations, interactions between person and digital artifacts, etc. Every theory has a certain focus, describing certain aspects of the huge research field of programming with visualizations. We decided to approach our study without any preconceptions. Aside from our ‘intuitive’ understanding about programming (as computer scientists and teachers) we leave the data to set the focus. Then in a second stage the results of this study can be interpreted using theoretical approaches.

Before describing the methodology of this research approach, in the next two subsections we introduce the programming course, the visualization tool VIP, and how the teacher presented and taught the tool during the course. We summarize and conclude this section with our research questions.

2.1 Course Setup and VIP

The course is a 3-month first-year programming course that focuses on imperative programming with C++. The only prerequisite is basic computer literacy. In addition to a normal program development environment, the students are also encouraged to use the visualization tool VIP during the course. The teacher (author Isolammi) introduced VIP through lectures based on her conceptualizations of VIP. The visualizations are available for students on the course web site for voluntary use. The course material integrates the visualization examples by including links and references to them in all the study materials that the students have to use [5]. This establishes a tight connection between the visualization examples and the other study materials used in the course, but the students are left to decide whether to use the visualizations and the VIP tool, and their decision does not directly affect their grade for the course.

The visualization tool 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 step by step by presenting pictures of the contents of the variables. It is possible to execute the program 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. 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 [15].

The VIP main window 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 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.

2.2 Instruction on VIP

At the start of the programming course, VIP can be used to follow the execution of simple example programs. This is the first and simplest way the teacher illustrates the use of the tool. However, this quickly becomes uninteresting for the students. The teacher then demonstrates that VIP can also be used when implementing small programming assignments, where students can write some program code in the code editor and run it to see how it works. This procedure includes testing in order to see if it works as expected and debugging in order to see what is happening and why in case of failure. In addition, the teacher points out that the students can still also use VIP to study new programs as they learned in the beginning of the course.

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 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 are advised to follow how the execution proceeds while testing the program.

The strategies taught both for testing and for debugging a program in VIP are similar to the single stepping strategy of Romero et al. [11]. They describe a debugging strategy where the student steps through program code by looking at the visualized data structures or output. The students are instructed to use the multiple views of VIP and not revert to traditional debugging strategies such as hand-simulation and causal reasoning, which are described by Katz and Anderson [4]. Hand-simulation means that the student mentally

\[\text{See a screen shot in http://www.cs.tut.fi/~vip/en/}\]
executes the program as the computer would and looks for inconsistent behavior. Causal reasoning means that the student gets information about the error by looking at the output of the program and then decides what might be causing the behavior. While these traditional debugging strategies were also explained during the programming course, they were not connected to the use of VIP, and the teacher did not explicitly name them.

2.3 Research Questions

Summarizing the previous subsections, the research questions of this study are:

1. What kinds of activity are students performing when using the visualization tool VIP to solve a typical programming assignment?
2. How do students’ activities differ from what was taught during the course?

By activity we mean a process of doing something, especially when dealing with a problem. Here, the problem is a typical programming assignment and we are interested in what a student is doing with VIP in order to solve it. We assume that students’ use of VIP is not limited to the way it is taught in the lectures. Therefore, we want to compare the results of research question 1 with the use of VIP as it was taught.

3. METHODOLOGY

From the methodological point of view it would have been best to conduct an ethnographic field study, following students for the whole semester and observing their use of VIP. Due to limited research resources it was not possible to implement such an approach, so instead we designed a two-stage data collection method.

3.1 Data Collection

In the first stage we wanted to collect data that identifies students who use VIP often and who know that they benefit from using it. For this purpose, we collected log files of students’ VIP usage during the whole semester. In addition, we asked the students to complete weekly questionnaires (with both multiple choice and open-ended questions) where we asked, for example, about their backgrounds, details of how they learn programming, and reasons for using the visualization tool. In the second stage we used the collected background information to identify students who used VIP often in their own study sessions. We invited 17 students to a combined observation and interview session at the end of the course. Half of these did not reply or did not want to attend the interview, so this session was attended by only eight of the invited students, which is still a suitable number for a study.

3.1.1 The Interviews

The interviews 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. Although the usability laboratory was not the realistic learning environment the students were used to, we assumed that the students would show some typical behavior after a three-month period of familiarization with programming assignments and VIP.

The interview was semi-structured and consisted of three phases. First there were general, easy questions about the interviewee’s opinions on learning programming. Then the interviewee 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, which were left to the end so that they would not affect the way the student used the visualization tool in the programming assignment. Each interview lasted 30-60 minutes depending on how much the student talked and how long it took to complete the programming assignment.

The main difficulty with the interviews was that the interviewer (author Isohanni) was also the teacher of the programming course. Although a different person was grading the exams, it is still uncomfortable for students to solve difficult problems while being observed and questioned by the teacher. In addition, Finnish people tend to be shy and tight-lipped, so it was difficult to get some of the students to share their thoughts. Thus the interviewer had to pose questions while the student was working in order to be sure of what was happening.

3.1.2 The Assignments

The programming assignments were designed to present something interesting in all the VIP windows at some point during the execution of the program. This means that both the algorithm and the data structures have to be challenging for the student. The topics (see Table 1) were chosen so that they were related to but not the same as the ones covered during the course.

Four programming assignments were created to provide the appropriate difficulty for each student. We chose the assignment individually for each student based on the background information we collected about the students. Assignments that were too simple were not suitable for our purposes, and due to limited resources, it was not possible to require the students to write a new, complicated program. Thus three of the assignments included program code that the student was required to modify and complete.

3.2 Data Analysis

Table 1: Assignments used in the interviews

<table>
<thead>
<tr>
<th>The Assignment</th>
<th>The Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Sorting the content of an array in an ascending order</td>
<td>Array of integers</td>
</tr>
<tr>
<td>2) Implementing a move of the king on a chess board</td>
<td>Two-dimensional array of characters</td>
</tr>
<tr>
<td>3) Marking an X in a simple coordinate system (ASCII graphic)</td>
<td>Two-dimensional array of characters</td>
</tr>
<tr>
<td>4) Calculating some percentage values into a statistic</td>
<td>Array of records</td>
</tr>
</tbody>
</table>

Due to constraints on the length of this paper, only the data analysis of the assignment part of the interview and observational session is described here. The interviews were
transcribed and their data coding was supported by the coding software MAXQDA. The data analysis was based on qualitative content analysis which is a methodology from qualitative social research used to systematically analyze textual data, see Mayring [9].

In our study, we started by addressing research question 1 and went through the interview transcripts marking relevant text samples for all students’ activities in the data and creating codes for them. Besides students’ actions with VIP, we also coded anything else of interest that appeared in the data. We summarized the codes looking for similarities and differences, explicating them to categories. 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 define the strategies students use. All this resulted in a conceptualization of students’ working patterns that is presented together with the category system in Section 4.

The interviews were conducted in Finnish. Because one of the authors does not understand the language, the immediate work with the data was performed by only one author who explained and summarized the first open coding process in English. In qualitative content analysis, at least the final data coding should be done by more than one person in order to measure inter-coder reliability. This criterion is useful during the analysis process because the researchers make many decisions about and interpretations of the data. By considering different perspectives in this process, more reliable results are obtained. Due to limited resources, it was not possible for a third researcher to repeat the coding or to translate the data. Therefore, during the whole data analysis process we instead followed the peer de-briefing approach to enhance credibility [8].

4. RESULTS

The results contain three parts that build upon one another and represent different levels of abstraction: the category system, the use situations, and the working patterns. The category system developed from a data-driven analysis focusing on research question 1, that is, focusing on what activities the students perform while working on the assignment. The categories group and structure the different aspects of students’ activities and help to reconstruct students’ experiences of working situations that describe a series of activities carried out while using VIP. In the further analysis and interpretation process we identified differences and similarities between the use situations and conceptualized their characteristics to working patterns. In the next subsections these results will be presented in detail, starting with the category system.

4.1 The Category System

In what follows we do not explain every category, but focus on those categories that are relevant to an understanding of the other aspects of the results. These are the four main categories use of study materials, student focus, interference, and cognitive activities. We will now describe these categories with selected subcategories (see also Table 2).

When solving the programming assignment, the student uses both the visualization tool (e.g., pressing the buttons in the user interface or reading the content) and other study materials such as the assignment description and paper and pen. Therefore, the main category use of study materials contains all the activities related to the material, while the main category student focus captures the student’s focus within the material used. The visual attention strategies of people using a visualization tool reveal that it is typical for the user to make connections between two or more of the windows [1]. Thus the same data can be coded multiple times with the code student focus. The subcategories of use of study materials and student focus are summed up in Table 2 and are self-explanatory.

Interactions that interfered while the student was working were gathered in the main category interference. This main category includes, for example, the non-programming-related subcategory computer problems and emotions which covers moments when the student was expressing desperation about solving the assignment. The subcategories hint with using VIP and help with programming are very closely related to solving the assignment. The interviewer needed to interfere in these situations because sometimes the student started spending a lot of time on something non-essential. The interviewer then guided the student so that the situation would proceed. Sometimes the student also asked for help with programming since it was faster to ask the interviewer than to start browsing the course materials that were also available.

These first three main categories specify what is happening physically in the learning environment. On the other hand, the main category cognitive activities captures what the student is thinking or reasoning about his or her activ-

<table>
<thead>
<tr>
<th>Use of study materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading</td>
</tr>
<tr>
<td>Writing</td>
</tr>
<tr>
<td>- Writing input for the program</td>
</tr>
<tr>
<td>- Editing code</td>
</tr>
<tr>
<td>- Writing new functionality</td>
</tr>
<tr>
<td>Programming activities</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Student focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>One subcategory for each window of VIP</td>
</tr>
<tr>
<td>Assignment description</td>
</tr>
<tr>
<td>Paper and pen</td>
</tr>
<tr>
<td>SSH-client</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interruptions</td>
</tr>
<tr>
<td>- Computer problems</td>
</tr>
<tr>
<td>- Interacting with the interviewer</td>
</tr>
<tr>
<td>- Practical arrangements</td>
</tr>
<tr>
<td>- Clarification of the assignment</td>
</tr>
<tr>
<td>- Help with programming</td>
</tr>
<tr>
<td>- Hint with using VIP</td>
</tr>
<tr>
<td>- Emotions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cognitive activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzing the assignment’s code</td>
</tr>
<tr>
<td>Reasoning about the implementation</td>
</tr>
<tr>
<td>Analyzing the program’s execution</td>
</tr>
<tr>
<td>Analyzing why something went wrong</td>
</tr>
<tr>
<td>Designing a test case</td>
</tr>
<tr>
<td>Planning for next activity</td>
</tr>
<tr>
<td>Making associations</td>
</tr>
<tr>
<td>Enlightenment</td>
</tr>
</tbody>
</table>

Table 2: Main categories and selected subcategories of the category system
2.1. The visualization properties of VIP are only used in the window of VIP and the code editor as described in Section 2.1. The visualization properties of VIP are only used in the main window. Therefore, we defined a use situation as a series of actions the student performs in the VIP main window. Each use situation only presents a short segment of the whole assignment session. It is easy to mark off the boundaries of one use situation in the transcripts because we have the category student focus.

In total, 35 use situations were reconstructed from the data, with each assignment interview containing 2-7 use situations. The reconstructed use situations contain three parts: 1) initial state, 2) action, and 3) final state. Because we cannot present the reconstructions of all of use situations, we here outline three different use situations: one where the student is stepping through correctly-working program code line by line; one in which the student runs the program code quickly using the run function of VIP and finds an error in the code; and one where the student finds an error but does not analyze the reason for the error in VIP.

4.2 Use Situations

When we analyzed the transcripts further, we found that certain sections of the interview were more relevant for our research questions than others. For example, when working with VIP, the student constantly switches between the main window of VIP and the code editor as described in Section 2.1. The visualization properties of VIP are only used in

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzing the assignment’ code</td>
<td>“I suppose this is the vector whose content will be printed. […] And there it prints the chess mark.”</td>
</tr>
<tr>
<td>Reasoning about the implementation</td>
<td>“I started thinking that I can’t write an else-statement here. […] Can I make it an else-if? No…”</td>
</tr>
<tr>
<td>Analyzing the program’s execution</td>
<td>“The [input] coordinates are correct, but it should have moved this to that position.”</td>
</tr>
<tr>
<td>Analyzing why something went wrong</td>
<td>“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].”</td>
</tr>
<tr>
<td>Planning for next action</td>
<td>“And then we can test smart things like... coordinates that do not exist.”</td>
</tr>
<tr>
<td>Making associations</td>
<td>“So I’ll change everything back to what it was in the good old times.” [=some minutes ago]</td>
</tr>
<tr>
<td>Enlightenment</td>
<td>“Now it’s starting a new iteration of the loop because 1 is 3... [reading the evaluation window and looking at the variables window in between]”</td>
</tr>
<tr>
<td>Enlightenment</td>
<td>“Of course! Because the indexing starts from 0 and not from 1.”</td>
</tr>
</tbody>
</table>

Table 3: Examples of the cognitive activities

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning for next action</td>
<td>“I need to swap that [the value of the variable temp] again. He looks at the values of the variables in the variables window. At the end, there is this exchange:</td>
</tr>
<tr>
<td>Making associations</td>
<td>“What do you see from VIP now?</td>
</tr>
<tr>
<td>Enlightenment</td>
<td>“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.</td>
</tr>
<tr>
<td>Final state</td>
<td>“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.</td>
</tr>
<tr>
<td>Final state</td>
<td>“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.</td>
</tr>
</tbody>
</table>

4.2.1 Example 1

Initial state: The student is writing a program that sorts the contents of an array in ascending order. He is starting the assignment. He has written a new piece of code that is supposed to find the smallest integer in the array. He is uncertain if the code works properly and also does not have a clear idea of what he should do next. Thus he decides to go to the main window of VIP to visualize the execution of the program.

Action: He starts to execute the program line by line. He is analyzing the program’s execution very carefully, looking at both the code window and the variables window. He is pressing the step button non-stop and watching the representations change on the screen. He is not just watching how the execution proceeds in VIP but also thinking what should happen next.

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

After following the execution for some time he decides that he can make the program run faster because the program is repeating the same thing he has already seen and he wants to see the end result more quickly. He stops the fast running when the program is close to the end and changes back to stepping line by line. He starts analyzing the program’s execution again. He looks at the values of the variables in the variables window. At the end, there is this exchange:

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: The student has used VIP to verify that his program works correctly so far. He also has an idea of how to proceed with programming.

4.2.2 Example 2

Initial state: The student’s assignment is to make modifications to a program handling a simple coordinate system. He is working with the assignment and has already corrected a couple of errors in his code. He wants to test his program to see if it is working correctly now.
Action: He presses the run button in VIP, and VIP starts executing the program fast. The program asks him to type input coordinates, and he types values that are supposed to be legal. He expects the program to print the coordinate system, but the program prints a message saying that the input was illegal. VIP has stopped running so all the representations in the windows are static. The student starts analyzing the program's execution using the representations. His objective is to understand what happened when the program was running. He looks mainly at the code window but also uses the evaluation window and the variables window. He keeps talking and explaining what just happened, and it sounds as if he is running the program in his head exactly the way VIP ran it.

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] Ummm... After this he is talking about not only what happened during the run but also why it happened. He concludes that the sign > in condition of the if structure coord.size() > y has to be changed to <. He does not explain the reasons for this conclusion to the interviewer but only makes the change in the code editor.

Final state: The student has used VIP to test his program with one input and found an error. He also found the reason for the error and knows how to correct it.

4.2.3 Example 3

Initial state: The student is modifying a program that handles an array of records. The program reads some values that are stored in the array, calculates the sum of these values and stores the sum in another position in the array. She has corrected one error from her program and wants to see if the program is now working correctly.

Action: She presses the run button and follows the execution in the evaluation window. The program is going through the array in a for-loop. She sees that the calculations are now correct and just waits until the loop is finished. The program reaches the next for-loop and the student changes her focus to the variables window. She looks at the picture of the data structure. After a short while of following the execution she points at the picture in the variables window and says that the program stores the same value in both the fields of the record. It is obvious that this is not correct. Then she presses the edit button and just sits down thinking. She does not look at the visualizations anymore. Neither does she explain why she did not use VIP to analyze the reason for the error.

Final state: The student has found an error in her program and used VIP to see how the program works so she knows what the program is doing wrong. She did not analyze why the program is working like this and does not know how to correct the error.

4.2.4 Characteristics of a Use Situation

Analyzing the differences and similarities between the use situations indicated important characteristics of a use situation that we could use when analyzing them further. For example, in the first and third use situations (Sections 4.2.1 and 4.2.3), the students run the program in VIP as they analyze it. In the second use situation (Section 4.2.2), the analysis happens with VIP stopped. In the first use situation the student does not find any errors in his program whereas in the other two use situations there is an error. The second use situation focuses on analyzing why the error happened and the third one on what happened.

In the first two use situations the students have an assumption of what should happen when the code is run (in the first one the assumption is correct, in the second it is not). However, in some use situations, VIP was used with no assumption of what would happen during the run. In the third use situation we cannot really tell whether the student has an assumption. Other characteristics that we identified from other use situations included, for example, whether the student’s analysis concentrated on program execution (the algorithm) or program output (the end result of the algorithm). By means of these characteristics, we grouped the use situations looking for similarities and differences. These made it possible to summarize the use situations and conceptualize them to working patterns.

4.3 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 combine them to a working pattern which we understand as a model that conceptualizes the use situations. Altogether, we found three different use objectives: 1) exploring, 2) testing, and 3) debugging. Among them we identified eight different use strategies. The next subsection will first introduce the working patterns on the level of the three use objectives to give an overall understanding. Then, focusing on the use strategies, we will present four working patterns that describe how students address an error in their program.

4.3.1 Use Objectives

The objectives reconstructed from the use situations are:

1. Exploring: The student starts working on the assignment by running the program code that was provided to see how it works. He or she does 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.

2. Testing: The student wants to see what happens during the run of the program. The objective is to test if the program is working correctly (the use situations presented in Section 4.2.1 and 4.2.3 are examples of this). If an error is found but the student makes no attempt to debug the program, the objective of using VIP is testing (as in the use situation presented in Section 4.2.3). The use situations related to the objective testing resulted in four working patterns.

3. Debugging: The student finds an error in the program and uses 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 4.2.2 is an example of this). The use situations related to this objective resulted in three working patterns.
4.3.2 Use Strategies

This section introduces the use strategies of three working patterns with the objective debugging. In addition, we also introduce one use strategy with the objective testing. We chose this additional working pattern to be presented because besides the debugging patterns it is the only other working pattern where the student is dealing with an error in the program.

The four working patterns we present here are summarized in Figure 1. The rectangles in the figure present the categories from the category system. The dotted rectangles indicate activities that are not always contained in the working patterns.

The first working pattern with the objective debugging is Dynamic use of VIP, illustrated with a normal, black arrow on the left in Figure 1. The students exploit the visualizations 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 is slower using the step function.

The second working pattern with the objective debugging is Static use of VIP, illustrated as the gray path in the middle of Figure 1. 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 4.2.2 exemplifies this). This means that students are actually performing the traditional hand-simulation debugging strategy in VIP's window.

The third working pattern with the objective debugging is Locating the error with VIP, illustrated with the dashed arrow in Figure 1. Here, the students locate the error immediately after seeing what happened during the run of the program and there is no further need to work with VIP.

The last working pattern, Abandoning VIP, has the objective testing and is illustrated with the dash-dotted arrow on the right in Figure 1. The students find an error in their program, but instead of debugging it, they totally stop using VIP, as in the use situation in Section 4.2.3. 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.

4.3.3 Reasons behind the Working Patterns

We cannot explain exact reasons leading to the working pattern Dynamic use of VIP. Nevertheless, we observed that this working pattern occurred only when students were very aware of which part of the program should be observed and also seemed to have a good understanding of the state of their program.

For the working pattern Static use of VIP we see two different reasons. In the use situation presented in Section 4.2.2 the use of VIP is stopped simply because the program has run until the end and it is not possible to continue using the step function in VIP. Exactly the same thing happens if the program crashes due to a runtime error. Another reason for stopping the run is that it was not worthwhile to continue since the interesting part had already occurred. Instead of using VIP for the analysis by running the program anew from the beginning to see the error again, students decided to analyze the reason for the error using the traditional hand-simulation debugging strategy. To support hand-simulation they use the representations in VIP's windows so that they do not need to calculate for themselves all the values of the variables from the start of the program's execution.

In these cases, it might have been helpful if the student had been able to rewind the run, for instance by using a step-back button. This would allow the student to observe a segment of the execution again. Literature lists vital pedagogical requirements for visualization tools, one of which is the rewind capability [12, 10]. Unfortunately, VIP does not contain this feature. Thus, we can say that the reason leading to hand-simulation in VIP was related to a usability problem of the tool.

The working pattern Locating the error in VIP typically happened because the same or a very similar error had just been solved by the student in an earlier use situation. Thus, it was so familiar to the student that there was no need to work on reasoning about the error.

By contrast, there were multiple reasons leading to the working pattern Abandoning VIP:

- In two use situations, the student did not seem to really understand how his program should work and was just experimenting with random corrections in the program code. It seemed that the lack of understanding prevented the student from analyzing the reason for the error even if the student was sufficiently skilled in VIP to determine what the program was doing.

- In one case, the representations of VIP were not helpful for the student in the further analysis and thus it was stopped. VIP was not helpful because it was an arithmetic error, and VIP does not present arithmetic operations any differently from a traditional programming environment.

- In one case the analysis was stopped because the student immediately recognized that the newest correction was totally wrong and wanted to cancel it and return to the earlier version.

- In four cases we could not reconstruct any reason for stopping the analysis. The students just stopped even if they did not understand what was wrong in their program. The use situation presented in Section 4.2.3 gives an example of this. There the student found the problem in the variables window but was not able to locate the program code that produced this problem. That is, the student is not able to make a connection between the variables window and the code window.

5. DISCUSSION

From the working patterns presented, Abandoning VIP was the only inconclusive one because these students did not understand the reason for the error. It is important to recognize that this working pattern was used by students who use VIP regularly and found it useful for learning. There were multiple reasons leading to this working pattern, including that the student was not able to make connections between the views of VIP and that the student simply did not try to use VIP any more. Can the more general reason for abandoning VIP be that students think of VIP as
Figure 1: Working patterns where the student is dealing with an error in the program.

a tool that shows only what is happening and not why it happens? Possibly this is the case at least for the students whose reasons for stopping the analysis could not be reconstructed. This is a hypothesis that can be taken into account for subsequent teaching.

The working pattern Static use of VIP presents an extremely inefficient way of utilizing the visualization tool. The student is calculating in his head what the tool could calculate for him. Nevertheless, this working strategy is successful because the students ultimately solve the problem. They may even learn more about programming than they would by running the program in VIP because they have to think about all the details. So even if it is an inefficient way to use the tool, it is not an inefficient way to learn programming. It is also not surprising that usability problems lead to inefficient uses of the tool.

5.1 Comparing What is Taught with What Students Do

To answer research question 2, we can summarize that students used VIP for the objectives the teacher instructed them to use VIP for. However, students' use strategies did not really correspond to the instruction. The students started working the way the teacher instructed them to do, mainly running the solution in VIP to find the error. But not all of the students analyzed the program execution during the run. After this, there are many variations of the second phase in the working patterns. The single stepping strategy that was expected by the teacher was used for analyzing the reason for the error in only one of the working patterns (Dynamic use of VIP). We can summarize that instead of single stepping, students could for example use the traditional hand-simulation debugging strategy (Static use of VIP pattern) or simply stop using VIP (Abandoning VIP pattern). In their study, Romero et al. [11] conclude that students with more debugging skill perform more single stepping than the students with less debugging skill, which could explain our observation.

As the teacher expected, the traditional debugging strategies do not become useless with a tool like VIP. Instead, the tool can facilitate new variations. In addition to hand-simulation, some students seemed to use the traditional debugging strategy of causal reasoning in parallel with single stepping when analyzing the reason for an error in the working pattern Dynamic use of VIP.

5.2 Suggestions for Teaching and Development of Visualization Tools

Our results indicate possible changes for teaching and also further didactic interventions. The students we observed for this study had studied programming for only 3 months and thus were not yet experts in debugging. Since the teacher was an expert in debugging, she expected the students to be more advanced than they were at this stage of their studies. It is important for teachers to consider the limitations of students' abilities while teaching. In the future, the teacher would not expect the students all to use fine single-stepping strategies but would instead guide the students more. Here, it would be appropriate to discuss debugging strategies generally in the lecture while introducing the use of VIP. The teacher would also emphasize that VIP can be used for understanding the reasons for what is happening in the program, and would provide examples during the lecture.

Finally, the results indicate that VIP design needs to be changed to better respond to students' usability demands. Since students use the hand-simulation debugging strategy despite its inefficiency, the developers of visualization tools could take this into account. Some features to aid hand simulation could easily be added to the tool. For instance, there could be a balloon help or some similar mechanism that would show the value of a variable when the cursor is placed over its name in the code window. Another possibility is that placing the cursor over the name of a variable would highlight its picture in the variables window.

Some students who found an error in the variables window were not able to connect it to the program code that produced it, and so abandoned VIP. Some of them could possibly have been helped by additional features in the visualization tool. For example, if the student was pointing at the pictures of a variable in the variables window it could be helpful for the visualization tool to highlight the line of code that last referenced this variable. Possibly there could
even be a mechanism for the student to choose whether to find the line that last referenced this variable or all the lines that referenced it.

The new features suggested here would help the student in making connections between the program code and the visual presentations of the variables in the other window. This could possibly help when the student is stuck with an error. Also earlier studies about visual attention strategies when using visualization tools [1] reveal that novice programmers have difficulties in making such connections between different windows.

6. CONCLUSIONS

We started this study because of our limited knowledge of the student perspective of VIP usage. We did not know how students who voluntarily and regularly use VIP for their programming assignments benefit from it or why and how they use VIP. In our data collection, we gathered background information about students’ VIP usage, and at the end of the course in an observational interview session we watched how they used VIP to solve typical programming assignments. We reconstructed 35 use situations, conceptualized them to students’ working patterns with VIP (of which we have introduced four in detail), and compared them with the way VIP was introduced during the programming course.

Our observational study was the first one we have conducted to examine students’ use of VIP. Although the study has its limitations, the results clearly demonstrate multiple ways that students use VIP. These results support our assumption that going into the details of students’ usage is much more important than simply testing the general educational effectiveness of a visualization tool.

The research design of the study was purposeful, and useful information for further actions has been gathered. The results of this study concentrated only on the usage of VIP. However, our research design is also a good starting point for a student-oriented research investigation into the use of visualizations more generally. Further research in this direction is needed because the understanding of students’ usage of visualizations is still limited [6].

This was the first study to learn about the usage of VIP and thus its approach was clearly data-driven. In future work we plan to connect our study to theory. Then it will be possible to generalize the results to other tools since the visualization tool used in the study is quite typical (albeit with no back-step ability). Then we will also compare our results to other studies in novices’ debugging processes.

7. ACKNOWLEDGEMENTS

We thank Lauri Malmi, Carsten Schulte, Päivi Kimmunen, and Hannu-Matti Järvinen for supporting this research project and helpful discussions. We specially thank Juha Sorva for his invaluable comments on the first version. Furthermore, we thank the reviewers for their comments that improved the paper, as well. Last but not least, we thank Simon for his thorough and careful proofreading of the final version.

8. REFERENCES


