A Study of the Difficulties of Novice Programmers

Essi Lahtinen
Tampere University of Technology
Institute of Software Systems
P.O. Box 553
FIN-33101 Tampere, Finland
essi.lahtinen@tut.fi

Kirsti Ala-Mutka
Tampere University of Technology
Institute of Software Systems
P.O. Box 553
FIN-33101 Tampere, Finland
kirsti.alamutka@tut.fi

Hannu-Matti Järvinen
Tampere University of Technology
Institute of Software Systems
P.O. Box 553
FIN-33101 Tampere, Finland
hannu-matti.jarvinen@tut.fi

ABSTRACT

Categories and Subject Descriptors
K.3.2 [Computers and education]: Computer and Information Science Education

General Terms
Human Factors, Languages

Keywords
Programming, learning, teaching, difficulties, novices

1. INTRODUCTION

Programming is not an easy subject to be studied. It requires correct understanding of abstract concepts. Many students have learning problems due to the nature of the subject. In addition, there are often not enough of resources and students suffer from a lack of personal instruction. Also the student groups are large and heterogenous and thus it is difficult to design the instruction so that it would be beneficial for everyone. This often leads to high drop-out rates on programming courses.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ITiCSE'05, June 27–29, 2005, Monte de Caparica, Portugal.
Copyright 2005 ACM 1-59593-024-8/05/0006 ...$5.00.

2. RELATED LITERATURE

Robins et al. [7] provide a comprehensive review on the research relating to programming education. Another good source of information is an older collection of research papers on novice programmers, edited by Soloway and Spohrer [8]. These papers provide several viewpoints on the characteristics and common misconceptions of novice programmers that should be considered when designing approaches for programming education.

These sources conclude, for example, that novice programmers are typically limited to surface knowledge of programs. They often approach programming "line by line" rather than using meaningful program structures. The knowledge of novices tends to be context specific, and they also often fail to apply the knowledge they have obtained adequately. They may know the syntax and semantics of individual statements, but do not know how to combine them into valid programs [9]. Hence, it is important to combine both concept knowledge and strategies for their use in the learning process.

Several approaches for CS1 courses have been presented, e.g. Fincher [1] surveyed "syntax-free", "literacy", "problem-solving" and "computing as interaction" approaches. The most common discussion topic in the literature of today seems to be whether imperative [2] or object-oriented [3] approach should be the first. Whatever the approach, at some point the students have to learn the basic structures of the programming languages such as loops, variables, recursion, and parameter passing. Several typical misconceptions related to language constructs are presented by Soloway...
and Spohrer [8] as well as by Pane and Myers [6]. For example, there are often misconceptions related to variable initialization, loops, conditions, pointers and recursion. Students also have problems with understanding that each instruction is executed in the state that has been created by the previous instructions.

In addition to the typical misconceptions presented in the literature mentioned, a recent survey by Milne and Rowe [5] ranked object-oriented programming concepts according to the level of difficulty. They had only 66 respondents in their survey, but it provided interesting information about the difficulties of the students today, with the present programming languages and programming environments.

3. SURVEY DESIGN AND METHODOLOGY

The earlier research on this area has often been carried out with older programming tools and languages [8], or concentrated mainly on certain language concepts [5]. We wanted to study the present situation with Java and C++ courses, and find out perceptions also on the different programming phases, learning situations and materials on the courses. This way we would not only gain ideas for topics that needed instruction but also information of the preferred material usage situations.

The web-based questionnaire had three different sections: background, course contents and learning aspects. The questions can be seen in Table 1. The same questions were asked on another form from the teachers as their perceptions on students’ difficulties. The goal was to compare the differences in the conceptions of students and teachers.

The first section contained the general information of the respondent (year of studies, experience in programming before university, computer skills, programming languages used). The purpose of this section was to be able to compare whether the background has impact on the learning difficulties. It was known in advance that most of the respondents had been studying programming either in C++ or Java so we expected a possibility to study the impact of the programming language.

The goal of the second section was to find out difficulties in learning the course contents. It was divided in two parts: the issues in program construction (e.g., using the program development environment or dividing the functionality into procedures) and the programming concepts. The questions asked the respondents to grade programming aspects and concepts on a five-point scale from very easy to learn (1) to very difficult (5). In addition, there was an option don’t know.

The third section contained questions about learning programming. The goal was to find out what kind of learning situations and materials the students find most effective. The scale for responses was the same as before, from 1 to 5, varying from the student feeling that he/she was learning never in that kind of situations to learning always. The questions about the materials 1 stood for practically useless and 5 for very useful.

The questionnaire was presented for students who had experience of 1-2 programming courses of their BSc and MSc programs in 6 universities. It was advertised also to the teachers of the same courses. Responses were received from Fachhochschule Furtwangen (FHF, Germany), Reykjavik University (RU, Iceland), Tampere Polytechnic (TPU, Finland), Tampere University of Technology (TUT, Finland), Bucharest University of Technology (UTCB, Romania), and Ventspils University of Technology (VENTA, Latvia). The questionnaire was available for 10 days.

4. RESULT ANALYSIS

In total, 559 students and 34 teachers answered the survey. The number of students’ responses from different universities are shown in Figure 1. The results of the sections Course contents and Learning and teaching programming are presented in Table 1. The Background information section is analyzed in the following.

More than half (58,6%) of the students taking part in the survey already had experience in programming before studying at the university. Almost half (40,6%) of the ones that had experience in programming, believed that their programming skills were at least moderate. This shows that students in a programming class often may have very different experience levels, which makes it difficult to design the teaching so that it would be challenging and interesting for everyone.

Majority of the students had been learning the basics of programming using C++ as the programming language. There were also some students who had used Java, and a small minority had used Pascal or other languages. The percentages of different languages being used are presented in Figure 2.
4.1 Course contents

The respondents perceived as the most difficult issues in programming understanding how to design a program to solve a certain task (I3), dividing functionality into procedures (I6) and finding bugs from their own programs (I7). These are all issues where the student needs to understand larger entities of the program instead of just some details about it.

The most difficult programming concepts were recursion (C4), pointers and references (C6), abstract data types (C9), error handling (C11) and using the language libraries (C12). Again, error handling requires understanding the program comprehensively. Using the language libraries requires independent searching of the information, which can make it difficult for the novices. Recursion, pointers and references, and abstract data types are abstract concepts and thus cognitively complex to understand without a similar phenomenon in the daily life for comparison.

The teachers’ opinions on the most difficult course contents were almost the same as the students’. In addition, the teachers perceived understanding programming structures (I3) difficult in issues about programming. In programming concepts, almost all the questions had the mean value above 3, and the most difficult issues were the same according to teachers and students. Teachers perceived systematically everything in the course contents more difficult to learn than the students. Figure 3 shows the differences on programming concepts.

4.2 Learning situations and materials

The students seem to be very self-confident, because they rated studying alone (S4) more useful than lectures (S1), and working alone on programming coursework (S5) more useful than exercise sessions (S2) and practical session (S3). Learning by doing was considered to be effective too, because exercise sessions (S2) were rated more useful than lectures (S1), and practical sessions in computer rooms (S3) even higher. Similarly, programming by themselves (S5) was rated more useful than studying by themselves (S4).

Example programs (M4) were considered as the most useful type of material both by the students and the teachers. The rest of the material forms were considered equally useful by the students. The teachers valued interactive visualizations more than the rest of the
4. Correlations

When analyzing the correlations of different programming issues and programming concepts from the students’ responses, it was found that the issues relating to understanding programming structures (I3), learning the programming language syntax (I4), understanding how to design a program to solve a certain task (I5), and dividing functionality into procedures, functions and/or classes (I6) all have a strong positive correlation with each other (0.534 < r < 0.637, p < 0.01). The student either learns all of these easily or has problems with all.

These four aspects seem to form some kind of a core of understanding programming, because they also correlate strongly (0.406 < r < 0.600, p < 0.01) with understanding most of the programming concepts (C1-C9). The other programming issues had clearly weaker correlations with learning the programming concepts.

The core programming issues correlate also with the rest of the programming concepts, but not as strongly. These concepts include handling input and output (C10), error handling (C11), and using language libraries (C12), i.e., issues that are usually not part of the core of the programming language. These concepts do not typically belong to the main topics on a programming course.

There were no significant correlations between the learning situations or materials and the course contents. The correlations in teachers’ results were also not significant.

4.4 Comparison between different languages

There were some significant statistical differences between the languages in the course contents. The teaching language did not seem to affect the learning situations. However, because different universities used different programming languages, it is possible, that the circumstances in the universities also affect the differences of the languages.

C++ was found to be more difficult than Java. Selection structures (C2), arrays (C5), pointers and references (C6), and parameters (C7) were perceived significantly more difficult when learning in C++ than in Java (p < 0.05).

Understanding the programming structures (I3) was significantly (p < 0.05) more difficult in other languages than in C++, Java or Pascal. However, there was no field in the questionnaire to reveal which other languages were meant here. Using the language libraries (C12) was easier in Java than in Pascal.

5. DISCUSSION

When interpreting the results, it is important to bear in mind that the responses are subjective opinions of the people who answered the students do not always see their difficulties completely. However, the number of responses is so large that the respondent group can be seen to represent the programming students and teachers of these universities well.

The survey results concerning the programming concepts confirm that the most difficult concepts to learn are the ones that require understanding larger entities of the program instead of just details, as also found in several articles in Soloway and Spohrer [8]. The results support also the notions made by Milne and Rowe [5]: abstract concepts like pointers and memory handling are difficult to learn. The results also showed a group of topics (e.g., input and output, language libraries) that should probably have more attention, since understanding them was not related to understanding the recognized “core” of programming.

However, the biggest problem of novice programmers does not seem to be the understanding of basic concepts but rather learning to apply them. Robins et al. [7] suggest that teachers should focus more on combination and use of these features, especially on the underlying issues of basic program design. In the results of the survey both students and teachers agreed that the practical learning situations were the most useful. Even if the theory is very important in learning programming, students also need practical experience to understand the concepts. The more practical and concrete the learning situations and materials are, the more learning takes place. Learning by doing should be a part of the studies all the time.

One of the problems in teaching programming seems to be that the students overestimate their understanding. The teachers think that the course contents are more difficult for the students than the students themselves. The reason for the different perceptions can be that the students do not realize all the difficulties they have, but the teachers do, for example, when assessing exams. Also, the teachers know the concepts deeper and they are able to see that the students do not have a full understanding of the issues students themselves think they understand completely [5]. Thus the students and the teachers see the need for different kinds of learning situations and materials differently. This can be seen as a possible source for problems in students’ motivation.

Since learning problems are often connected to more advanced issues than individual concepts, learning materials could be directed to develop program generation, modification and debugging skills. If small examples, emphasizing few concepts at a time, could be developed to support students’ active programming skills, they would also better engage the student in the learning situation. Since success in creating a functional program is a major positive force on students’ traditional programming work, materials should have more problem-solving nature instead of only representing concepts.

For future work, the questionnaire could be designed so that it would be possible to study the impact of the programming language and the environment used. In this survey it was only possible to see that the different languages have impact on learning some of the
programming concepts. Following the development of the difficulties on the same group of students or individuals in a long-term research could reveal more detailed information.

6. CONCLUSIONS

Programming is not difficult only because of the abstract concepts. Students have also problems in different issues related to program construction. It is important for the learning that the students do programming by themselves. With carefully designed materials and approaches teachers can guide students knowledge and skill construction.

The survey studied the students’ and teachers’ perceptions of the difficulties in learning programming. The results provide an extensive amount of data on perceived difficulties related to programming concepts and program construction. The survey gives also information on students’ perceptions of the most useful material types and learning situations. These results can be used when designing materials and approaches for basic programming courses.

7. ACKNOWLEDGMENTS

We would like to thank Minerva (Codewitz) project for funding. We would also like to thank Fachhochschule Furtwangen, Reykjavik University, Tampere Polytechnic, Bucharest University of Technology, and Ventspils University of Technology for helping with collecting the information.

8. REFERENCES