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.ala-mutka@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

Programming is related to several fields of technology, and many university students are studying the basics of it. Unfortunately, they often face difficulties already on the basic courses. This work studies the difficulties in learning programming in order to support developing learning materials for basic programming courses. The difficulties have to be recognized to be able to aid learning and teaching in an effective way.

An international survey of opinions was organized for more than 500 students and teachers. This paper analyses the results of the survey. The survey provides information of the difficulties experienced and perceived when learning and teaching programming. The survey results also provide basis for recommendations for developing learning materials and approaches.

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.

ITiCSE'05, June 27–29, 2005, Monte de Caparica, Portugal.

Codewitz (www.codewitz.net) project aims to develop solutions that would benefit teaching and learning programming. The main purpose of the project is to develop web-based visualisations of programming concepts for use in classroom and for supporting independent learning. Unlike several approaches that concentrate on algorithm animation, e.g. [4], this project aims at improving students' learning introductory programming concepts and language structures, e.g., variables, loops, and conditional statements. For developing international co-operation, it was decided to organize a large survey among partner universities to study the present difficulties in learning programming. The results of this study could be used both as a basis for developing new visualisations as for developing learning approaches for programming courses generally.

The organization of the article is as follows. Section 2 contains an overview of the related literature. The survey design and methodology will be introduced in Section 3 and the analysis of the results in Section 4. The results will be discussed further and related to other studies in Section 5. Section 6 contains the conclusions.

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*. In 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.

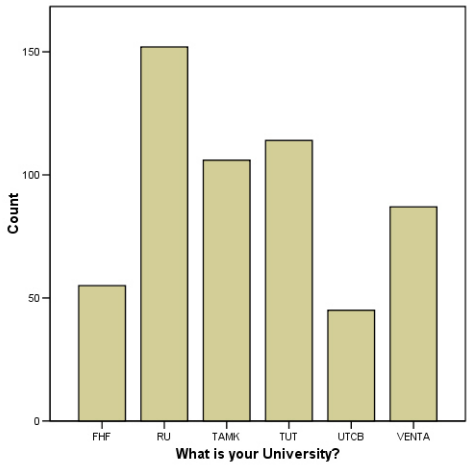

Figure 1: Number of students' responses by university.

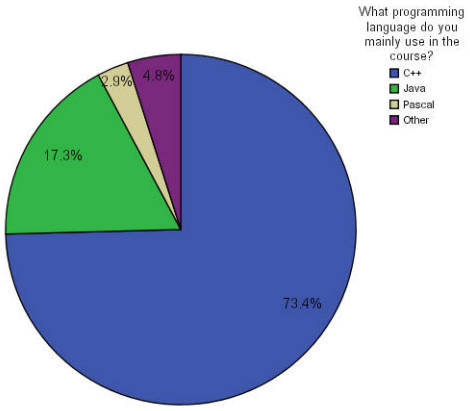

Figure 2: Proportions of different languages in the survey.

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.

Table 1: Results on sections course contents and learning and teaching.

Question	Code	Students			Teachers		
		N	Avg	Std	N	Avg	Std
THE COURSE CONTENTS							
What kind of issues you feel difficult in learning programming?							
Using program development environment	I1	553	2,43	0,99	33	2,61	0,90
Gaining access to computers/networks	I2	536	2,11	0,95	32	1,97	0,78
Understanding programming structures	I3	556	2,92	1,02	33	3,27	0,67
Learning the programming language syntax	I4	555	2,75	1,01	33	2,70	0,73
Designing a program to solve a certain task	I5	555	3,12	0,98	33	3,97	0,73
Dividing functionality into procedures	I6	543	3,10	1,09	31	4,06	0,63
Finding bugs from my own program	I7	549	3,28	1,03	33	3,91	0,77
Which programming concepts have been difficult for you to learn?							
Variables (lifetime, scope)	C1	541	2,10	0,97	34	2,41	0,70
Selection structures	C2	552	1,98	0,90	34	2,38	0,70
Loop structures	C3	551	2,09	0,97	34	2,79	0,91
Recursion	C4	512	3,22	1,03	31	4,06	0,96
Arrays	C5	526	2,79	1,15	33	3,24	0,71
Pointers, references	C6	518	3,59	1,04	32	4,44	0,56
Parameters	C7	513	2,60	1,09	32	3,47	0,76
Structured data types	C8	496	2,90	1,03	31	3,45	0,81
Abstract data types	C9	499	3,02	1,10	31	4,06	0,81
Input/output handling	C10	519	2,96	1,04	32	3,75	0,88
Error handling	C11	481	3,33	1,01	32	4,13	0,79
Using language libraries	C12	465	3,04	1,09	32	3,88	0,71
LEARNING AND TEACHING PROGRAMMING							
When do you feel that you learn issues about programming?							
In lectures	S1	543	3,01	1,01	33	3,21	1,02
In exercise sessions in small groups	S2	510	3,44	1,10	32	3,84	0,99
In practical sessions	S3	514	3,77	1,03	31	4,35	0,75
While studying alone	S4	546	3,79	1,06	31	3,42	0,72
While working alone on programming coursework	S5	539	3,98	1,09	33	4,00	0,79
What kind of materials have helped/would help you in learning programming?							
Programming course book	M1	515	3,35	1,03	33	3,30	0,88
Lecture notes/copies of transparencies	M2	539	3,39	1,05	34	3,47	0,71
Exercise questions and answers	M3	523	3,33	1,07	34	3,62	1,02
Example programs	M4	551	4,19	0,86	34	4,24	0,65
Still pictures of programming structures	M5	490	3,15	1,00	30	3,70	0,75
Interactive visualizations	M6	315	3,33	1,03	27	4,07	0,87

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* (I5), *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 excercise 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

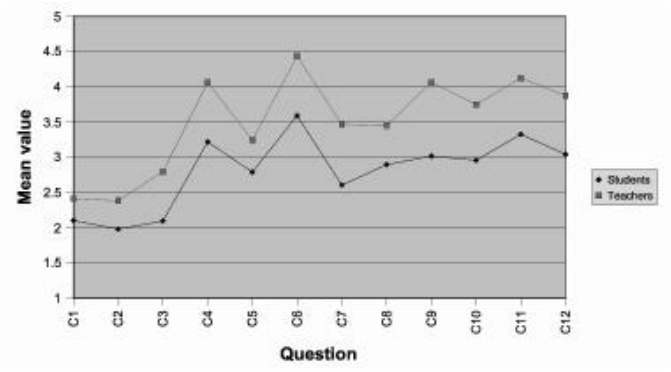

Figure 3: Difference of students' and teachers' responses concerning the programming concepts.

materials, but it can derive from the fact that the survey was carried out among teachers that are interested in developing visualizations.

The learning situations were seen differently among the teachers than among students. The teachers thought that the most effective learning situations were *practical sessions in computer rooms* (S3), *exercise sessions in small groups* (S2) and *working alone on coursework* (S5). Either the teachers seem to think that the students need guidance more than the students themselves or the teachers consider their teaching more effective than it actually is, because they rated all the guided learning situations higher than the students.

4.3 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 completelyq. 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

[1] S. Fincher. What are we doing when we teach programming? In *Proc. of the 29th ASEE/IEEE Frontiers in Education Conference*, pages 12a4–1–12a4–5, November 1999.

[2] C. Hu. Rethinking of teaching objects-first. *Education and Information technologies*, 9(3):209–218, 2004.

[3] M. Kölling. The problem of teaching object-oriented programming. *Journal of Object-Oriented Programming*, 11(8):8–15, 1999.

[4] A. Korhonen and L. Malmi. Algorithm simulation with automatic assessment. In *Proceedings of the 5th annual ITiCSE conference*, pages 160–163, 2000.

[5] I. Milne and G. Rowe. Difficulties in learning and teaching programming - views of students and tutors. *Education and Information Technologies*, 7(1):55–66, 2002.

[6] J. Pane and B. Myers. Usability issues in the design of novice programming systems. *School of Computer Science Technical Reports, Carnegie Mellon University, CMU-CS-96-132*, 1996.

[7] A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003.

[8] E. Soloway and J. Spohrer. *Studying the Novice Programmer*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[9] L. E. Winslow. Programming pedagogy – a psychological overview. *SIGCSE Bulletin*, 28(3), September 1996.