Learning Programming by Programming: a Case Study

Marko Hassinen*
Department of Computer Science,
University of Kuopio
P.O. Box 1627
70211 Kuopio, Finland
marko.hassinen@uku.fi

Hannu Mäyrä
Department of Computer Science,
University of Kuopio
P.O. Box 1627
70211 Kuopio, Finland
hannu.mayra@uku.fi

ABSTRACT
Programming is a challenging field of computer science for both to teach and learn. Although studied extensively, a definite method for teaching programming is yet to be found. In quest of finding success factors in both elementary and more advanced programming courses, this paper discusses some findings made studying exam success and home assignment activity in programming courses. Our claim is that there is no shortcut in learning to program, but extensive practise and sufficient time to become familiar with programming concepts is needed.

Keywords
Learning programming

1. INTRODUCTION
Programming is probably the most challenging field of computer science to teach [2]. Studying proramming involves not only learning new things but more importantly learning how to apply this new knowledge to solving problems. The ability to see how things are connected to each other and derive new constructs from existing ones is utterly important for a programmer. Terms like knowledge, understanding and expertise can be used to differentiate the abilites needed to program a computer.

While knowledge can be gathered by reading books or attending lectures, understanding requires deeper processing of the knowledge and, in most cases, hands-on practise [1]. We claim that only through adequate practise and training can expertise be obtained in the field of programming. To support this claim we evaluate results of two programming courses and study the importance of practise in students success on these courses. In the light of our findings we discuss two methods for measuring learning on programming courses, namely the traditional paper exam and a novel approach where exam assignments are pure programming assignments carried out with a computer. Both of these methods are contrasted with a study of how student activity in completion of practical assignments during courses affect their success in exams.

2. METHOD
The material was collected at University of Kuopio from elementary programming course for first year computer science students in 2001 and 2004 as well as advanced network programming courses for third and fourth year students in 2002 and 2005. On the elementary programming course the programming language was Java, and on the network programming course there were several programming languages, such as Java, PHP, and Perl. The material contains a total of 226 records, 153 from the elementary course and 73 from the advanced course.

The students were given assignments which they were expected to solve by a certain date. Excercise sessions were held on these deadline days, where each student marked which assignments he or she had solved. Students who solved more than 50% of the given assignments were given an extra point to the final grade and students who finished more than 75% received two extra points to the final grade. The grade scale was from three to twelve. A threshold of 30% was kept as a requirement for taking the final exam. A student not meeting the threshold had to pass an additional exam to compensate for the missing exercises. As solved assignments gave students extra credit, marking solved assignments was controlled by random checks.

Individual learning results were evaluated with exams. The students could pass the course either by taking two small exams, one in the middle of the course, another one in the end, or by taking the final exam. The smaller exams had two formats, either a traditional paper exam or a computer exam, in which a student was given a programming task to solve and a computer to do it with. The expected result was a computer program solving the given task. The students were given the opportunity to take either the paper exam or the computer exam or both, in which case the better result was considered in the overall evaluation.

Feedback was collected at each course. On the elementary courses, feedback was collected twice, first time in the middle of the course and a second time in the end of the course. On the more advanced network programming course feedback was collected at the end of the course. The feedback form consisted of free form questions with an open question for any comments they wanted to give to either the lecturer or the teachers responsible for the excercise sessions.

3. RESULTS
Figures 1 and 2 depict how completed assignments reflect to the final grade. Figure 1 contains all 226 observations and their average plotted on the gray curve. Figure 2 shows the grade distribution averages of the two courses on separate curves. The material contains all students who got a grade from the course. However, students who failed to pass the exam are not included. Also, there is no information about failed exams of students who failed the first exam, but passed a subsequent retake. Unfortunately

*Corresponding author

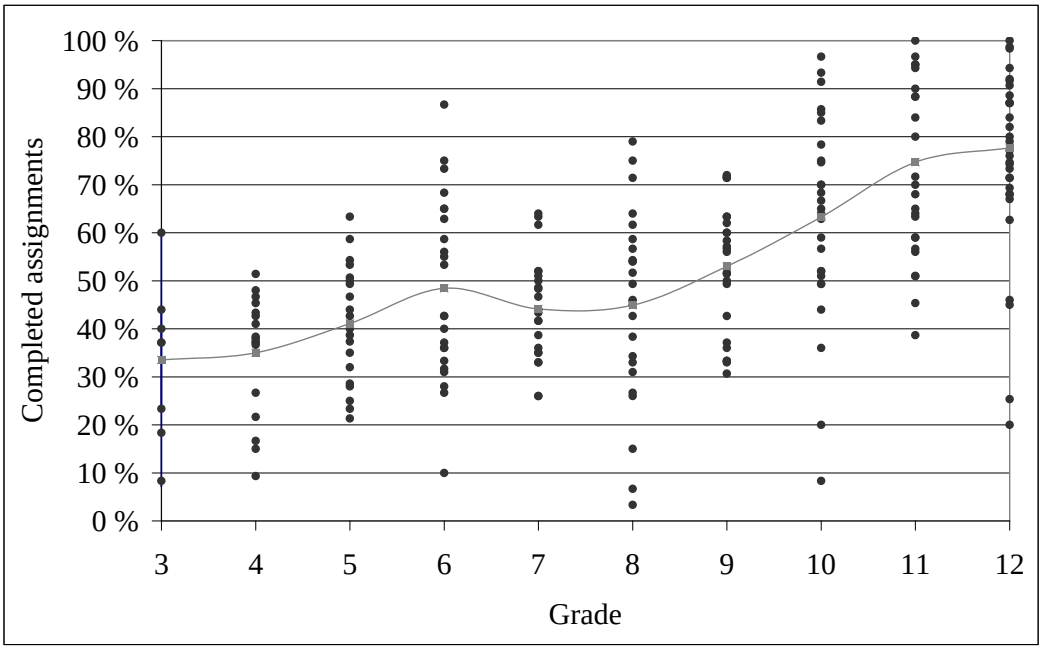

Figure 1: Distribution of completed assignments in Programming I and Network programming courses

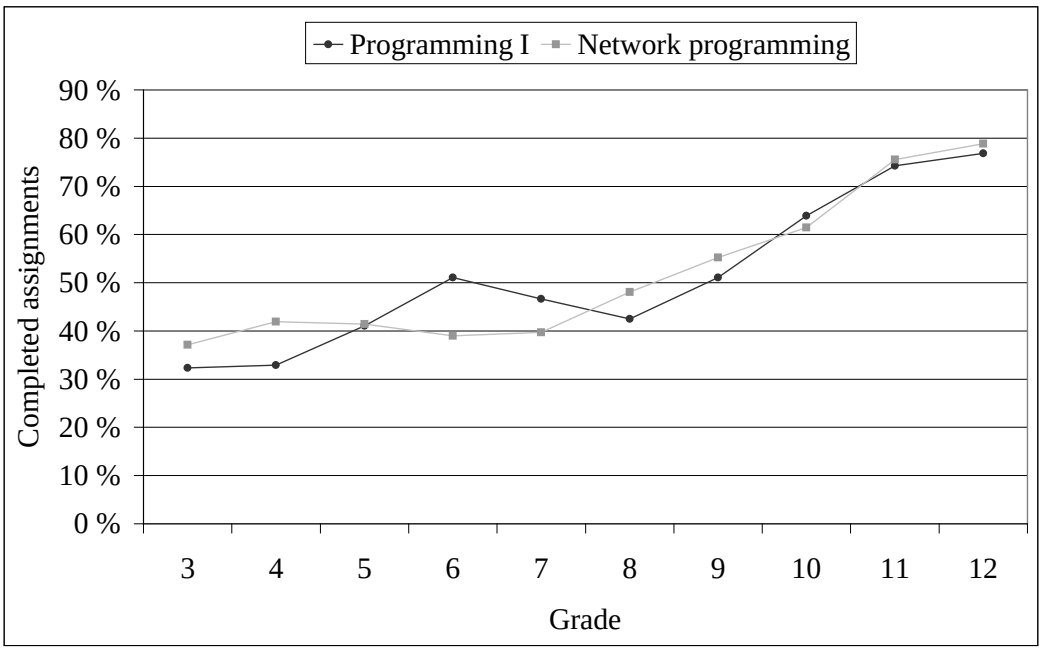

Figure 2: Average on completed assignments per grade

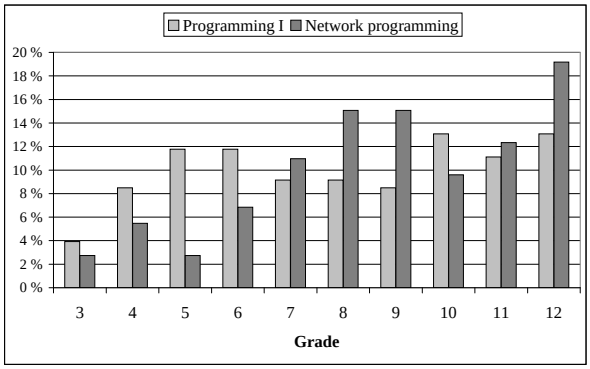

Figure 3: Distrubution of grades

the information on failed exams was no longer available.

From analyzing the material it became obvious that there is a strong positive correlation between the number of completed assignments and exam success. Students who had finished more than 75% of the given task scored excellent marks almost without exception. On the other hand, students who completed just the amount of assignments required to take part in the exam mostly scored low grades. However, the low assignment score had relatively more good exam results than there were poor results in the high assignment score group. This was an expected finding, since there obviously is a very wide range of different levels of knowledge in programming and general computer science among the first year computer science students. Students who already had programming knowledge and experience could pass the course with considerably smaller effort than those who started from the basics.

A very interesting observation can be made from Figures 1 and 2. It seems that both graphs present an s-shape in which there is a clear drop on exercise activity with grades six to eight. It seems that students who scored low grades had been more active in completing their assignments than those who scored mid range grades. This phenomena was present in both the elementary programming course as well as in the advanced programming course. However, the drop was considerably more moderate with the advanced course. A definite explanation could not be found.

Grade distribution is depicted in Figure 3. Extra credits given for those who completed more than 50% of the assignments partly explain the high percentages with the excellent grades.

The first and most evident observation from the results of the feedback was that as students are individuals, there seems to be no way to tailor a programming course that would suit each student optimally. The question about the pace of the course received a nearly equal amount of answers stating that the pace was too fast as those stating a too slow pace. Fortunately, the mid range that were happy with the pace was the majority.

Feedback from students supports also our observation; a large group of students answered that solving home assignments supported their learning experience the most.

4. OPEN QUESTIONS

Our study opened following interesting questions:

- How could the drop in exercise activity in the six to eight grade range be explained?
- What should be the ratio of excercises and lectures on the first programming course? Should this ratio change with more advanced courses?
- How do you motivate students in doing the assigned tasks and to invest the time it takes to complete these assignments?
- What is the impact of providing sample solutions to assigned tasks? Should one do that at all?
- It takes very different amounts of time for the students to finish a given assignment. While some students can solve easy assignments in matter of minutes, it may take hours for others to do the same. Is there any way to bridge this cap or is that even necessary?

5. CONCLUSION

Our findings clearly support the learning-by-doing aspect of studying programming. It can be seen that scoring excellent marks without extensively practising is not possible. In our study exceptions in this were students who had moderate to extensive programming experience already before the course.

6. REFERENCES

[1] R. Bruhn and P. Burton. An approach to teaching java using computers. *The SIGCSE Bulletin*, 35(4):94–99, 2003.

[2] 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.