Student's diversity problem in programming courses

MICHAL BLAHO, MARTIN FOLTIN, PETER FODREK, JÁN MURGAŠ
Institute of control and industrial informatics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Ilkovičova 3, 812 19 Bratislava
SLOVAK REPUBLIC
michal.blaho@stuba.sk, martin.foltin@stuba.sk

Abstract: Ability to produce computer program is one of the necessary skills of today's engineers. Several courses at early stages of study at the Faculty of Electrical Engineering and Information Technology of Slovak University of Technology in Bratislava teach how to program in some basic programming languages necessary for engineers praxis. Problem of these courses is student's diversity in programming skills. Many of students don't have programming experience at all and some students know much more than course can offer. Course setting that can be beneficial to all students is very challenging problem to solve.

Key-Words: - Programming skills, learning, students diversity, students opinion, operating systems, programming languages

1 Introduction

Demand for high quality engineers is big today. For example Germany needs more than 30000 engineers [1,2,3]. Modern engineers must have good theoretical knowledge and practical experience. One of many skills that are necessary in praxis for our students is know how to do computer programming.
Many teachers agree, that many students have problems dealing with the learning of programming [4,5]. Teaching style of programming is usually individual for each student, therefor is almost impossible to choose right style for the programming course. Great help in this problem are many information sources that students can use for computer programming during learning process [6].
What we can do to improve out teaching of programming is to incorporate modern learning strategies like collaborative learning [7,8]. We try some of them with different success in our teaching praxis. We present more practical rather than scientific perspective in this paper.

2 Students diversity

Students from different parts of the country study at the universities. They have been studying at various secondary schools with various focuses on topics. We have more than 250 secondary grammar schools (focus on general knowledge) with about 90 000 students and more than 500 secondary technological schools (focus on engineering) with about 180 000 students in Slovak republic.

The one of the common problems in early terms on faculties is the student's diversity. Because they studied at the different schools they have different basis of mathematics, physics, computer programming, technology, etc. The teacher's job is to reduce gap between students knowledge necessary for study at the faculty. We focus mainly on operating system usage and programming skills in our study.

2.1 Operating systems

Almost every student basically from primary school starts dealing with a computer. In our praxis (industrial informatics) variety of operating systems is used, for example Unix based systems in embedded systems or real-time control. It was in our interest to find out what operating systems students had contact with before study at faculty.
The major operating system on market is Microsoft Windows. It is also well known between students and 99.7% answered that they are familiar with this operating system. Open source Unix based operating systems are growing in popularity because they are free and have near same functionality as Windows. Students know these systems and 46% are familiar with Unix based operating systems. Apple operating systems are known for support for students and study process. In our country they aren't much spread because the expenses but 10% of the students have seen this operating system. Other operating systems used 3% of the students.

ISBN: 978-1-61804-021-3

2.2 Programming experience

Students come to university from different schools and have different knowledge background as we mentioned before. We have found this fact in our courses (mainly in computer programing) when some students can understand lectures and practices easily and some have problems. The difference is often enormous. We were curious how students are prepared from secondary schools and if they had programming after all. If they have been learning computer programming by them self is another question.

2.2.1 Computer programming at school

Teaching informatics in secondary schools is certainty. But the question is, if the schools learn how to design computer programs. On the question, if students had course of computer programming 90% answered positive. The rest 10% haven't got any programming courses yet.

This doesn't mean, that 90% of the students understand principle of computer programming well and are good at algorithm design understanding. Many of them had various teachers or programming topics. Some secondary schools prefer different programming languages then others. We try to find out in our questionnaire what programming languages they learned in secondary school.

The most popular language for teaching programming at secondary schools is Pascal. About 71% students have learned this language. Delphi is similar to the Pascal, which is introductory to objective programming and has better graphical user interface capabilities. Delphi is familiar to 10% of the students. Modern programming courses (mainly at universities) starts programming courses with C or C++. About 25% of students learned those languages so they have good chance to pass exams. Technological secondary schools teach low-level programming language Assembly. More than 30% of students learned this language that is used for programming microcontrollers in industrial informatics. More and more secondary schools start to learn more and more popular web technologies like Html, Php or Javascript. Students like this technologies because they are relatively easy to learn. About 21% have been learning web technologies at school. The most popular and used objective language Java has learned only 1% of students.

2.2.2 Programming in free time

You don't necessary learn what you want to learn at secondary school. Many students are curious and therefor learn some programming languages by them self. These students often achieve better results and have deeper understanding of programming languages rather than students who learn at school. They wanted to learn, but they don't need it, which make the difference.

As a contrast to the school programming free time programming has different distribution of students that learned programming language. Near half of the students (42%) learned web technologies in their free time. The next favorite programing languages for the students are C/C++, near 27%. Pascal is also popular but not as much (11%). Java is the last known popular language with 10%. Other mentioned languages in previous part have less than 10%.

2.2.3 Comparison

If we compare what were students learned at the secondary school and what they learned in their own free time, we will see what languages and programming paradigm they know best. Secondary schools learn procedural programming (Pascal, C/C++) and web technologies. Student focus mostly on web technologies but also in procedural and objective oriented programming (Java).

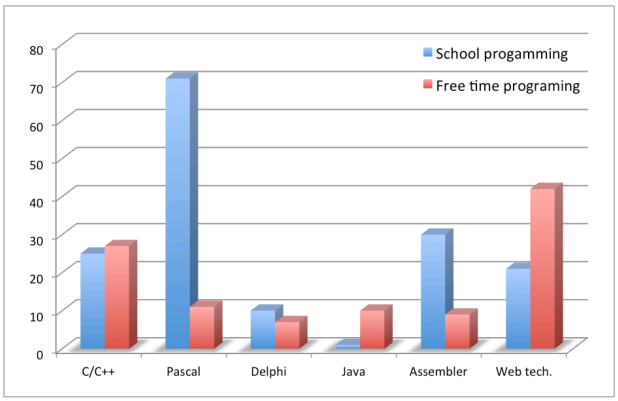

Fig.1 Comparison of learning styles

3 Programming at faculty

As we mentioned before, students with different knowledge of programming are coming to the faculty. The first few terms can be hard for students that hadn't programming courses at the secondary school.

3.1 Problems in courses

We wanted to know how students see difficulty of the programming courses at the university. We ask them, how big problems they had in computer programming courses.

ISBN: 978-1-61804-021-3

Answers were divided into five groups by problems degree. Major problem had almost 10% of the students. They wasn't capable understand most of the lectures or practices. Above average problems had 21% students. Average problems had almost 35%. These problems are usual on every course. Problems beyond average had 20%. About 13,5% of the students hadn't any problems in programming courses. These students came to faculty prepared for programming courses.
The distribution of answers is standard Gaussian distribution as you can see on figure 2. For this reason we asked this question universally and it applies to all programming classes at the faculty.

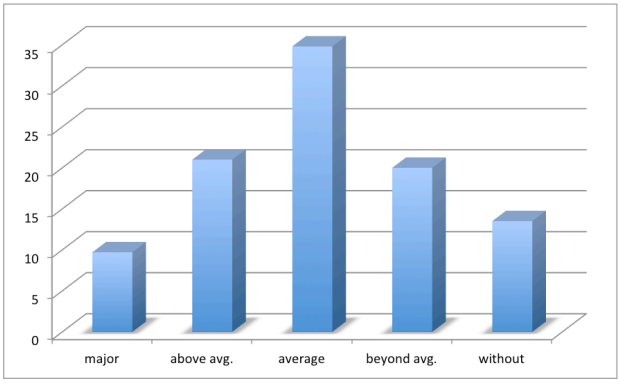

Fig.2 Degree of problems

3.2 Flipped Gaussian distribution

Dehnadi and Bornat [5] point out that student can be divided into two general groups of students. On group of students is extremely difficult to teach programming. The other group is much easier to teach. This group was successful in programming and found it easy. This can be plot on distribution figure of exams. One peak is in right half where are students with no problem with programming. Other peak is on left where are students with serious problems in programming learning and fail at exam test.

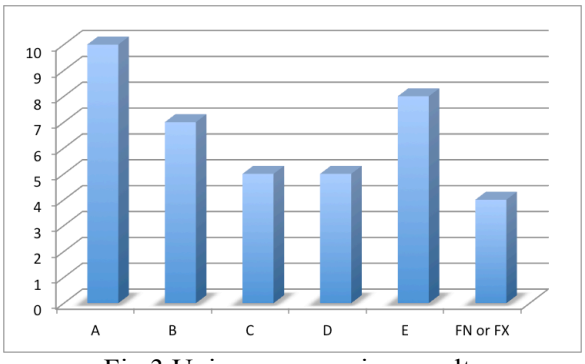

Fig.3 Unix programming results

Exam results from some courses confirm Dehnadi and Bornat theory. In Unix programming course (figure 3, FN or FX means absence at exams or failure) students result have two peaks. Java programing course (figure 4) have three peaks of results. Figures show, that students had in fact problems with passing course or fail at exams.

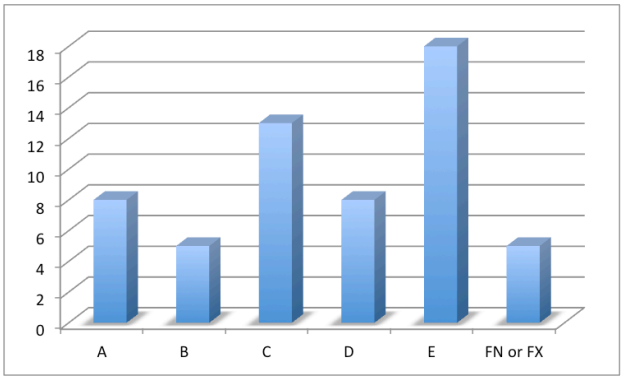

Fig.4 Java programming results

4 Our suggestions

It is hard to suggest universal methodology for teaching computer programming. In our case it look like following topics may help to improve teaching and learning processes. Talented students with practical experience, who are able to pass final exam before start of the course they may be taken individual projects corresponding to their knowledge. It is also possible to use them as consultants for the lecture to improve teaching process or as personal consultants for less prepared students.
This changes practically removes left peak students from figures 3 and 4. It allows redesigning course to better-fit less prepared students requirement. Some of the courses are to be modified to teach team organization and teamwork. Classification may to be done offline not online as now. This means that students upload their programs to server and lectors may study programs at home. This allows them to design more concrete questions for students to find out their knowledge of the program as well as quick modify of the next lecture study material.

5 Students opinions

After questions about secondary school programing and programing at the faculty we asked student what other comments they have to computer programming learning at the faculty.
The first group of students haven't got any programming courses because their specialization didn't offer any, but they still wanted to learn how

ISBN: 978-1-61804-021-3

to program. Some students wanted to have better basis of algorithm designs. Another answers leaded to courses of economy. Students suggested that they have to teach too many of them. They would be replaced by more useful programming courses for their praxis.

The next group of student wanted to modernize the programming courses. Better students can really good see what is used in praxis especially if they begin part-time work meanwhile study. Students want more practical examples rather than theoretical lectures. Another demand was to connect lectures to their courses in their specialization or some visits to real-world companies.

Some students wanted to work in teams on few programming courses. In praxis there people work this way. Another useful technique is programming management and job scheduling. Also modern developing environments would be used during teaching process.

Very surprising were statements that we would take more of originality check of student programs made for classification. Some students demanded harder tasks. Other group wanted to learn more electronics than programming.

6 Conclusion

We wrote about problems in compute programing teaching in this paper. Student's diversity in programming knowledge and computer usage is huge factor. They come from several secondary schools where they learned different kinds of programming languages.

This is one of few factors why some students can't pass some programming courses in first or two years of study. Some exams results show that there are two types of student. One can learn programming easily and other usually fails.

We proposed some ideas how to moderate this fact. Integrate better students into teaching process can shift negative trends in programming teaching. Helping with modernizing lectures or collaboration with no so well prepared students can be beneficial. Individual projects can improve knowledge for better students. Shifting difficulty of courses towards no so well prepared students can increase what they could learn.

Many students at the faculty had interest in improving quality of computer programming learning. In a first day of our questionnaire through Facebook and university information system we had over 500 submissions (almost 20% of students). We can't underestimate student's opinions and their will of improving courses and in end effect us teachers.

Acknowledgement

This work was supported by VEGA agency under contract number 1/0592/10, and KEGA agency under contract number 032STU-4/2011.

References:

[1] J. Kinast, Ch. Reiermann, M. Sauga, Labor Paradox in Germany: Where Have the Skilled Workers Gone? [online], *Spiegel online*, SPIEGELnet GmbH 2007 cit: 11.05.2011 available at <http://www.spiegel.de/international/business/0,1518,490031,00.html>

[2] C. Barry, Germany Needs 34,000 Engineers, *Product Design & Developlemnt* [online] Advantage Business Media, 2010, cit: 11.05.2011, available at <http://www.pddnet.com/news-germany-needs-34000-engineer-112410/>

[3] AFP, Westerwelle: Germany needs foreign workers, *The Local* [online], The Local Europe GmbH, 2010, cit: 11.05.2011, available at <http://www.thelocal.de/national/20100804-28957.html>

[4] S. Garner, The Cloze Procedure and the Learning of Programming, 8th WSEAS International Conference on COMPUTERS, Athens, Greece, 2004

[5] S. Dehnadi, R. Bornat, The camel has two humps (working title) [online], Middlesex University, UK, 2006 cit:11.05.2011 available at <http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf>

[6] C. J. Costa, M. Aparicio, R. Pierce, Evaluating Information Sources for Computer Programming Learning and Problem Solving, Proceedings of the 9th WSEAS International Conference on APPLIED COMPUTER SCIENCE, 2009, pp. 218-223

[7] D. T. D. Phuong, F. Harada, H. Takada, H. Shimakawa, 5th WSEAS / IASME International Conference on ENGINEERING EDUCATION (EE'08), Heraklion, Greece, July 22-24, 2008

[8] J.A. Marin-Garcia, J. L. MAURI, Teamwork with University Engineering Students. Group Process Assessment Tool, Proceedings of the 3rd WSEAS/IASME International Conference on Educational Technologies, Arcachon, France, October 13-15, 2007, pp. 391 - 396