Changes in Computer Science Teaching at Warsaw University

Janusz Jabłonowski

Abstract: *The paper describes the ongoing changes in the program of computer science teaching at the Institute of Informatics at the Warsaw University (department of Mathematics, Informatics and Mechanics). We describe the current curriculum, the reasons for changing it and the proposed new curriculum. The author of this paper was engaged in the process of devising the new curriculum at all stages as the member then head of the institute committee for the new curriculum and recently also as the institute's vice-director responsible for teaching.*

Key words: *computer science, teaching of computer science, curriculum in computer science, curriculum in software engineering.*

INTRODUCTION

The problem of teaching is well known to be extremely difficult. Even if we agree which output the teaching process should have – i.e. what knowledge and capabilities a graduate should posses (a question which if far from being trivial), we do not have an easy, fair and unbiased benchmark for deciding which teaching approach is better. Ideally to compare two teaching approaches we should apply them to two representative groups of students and compare, when they will be ending they professional life (i.e. when they will retire), their professional achievements. Apparently this is not possible (at least because of those students which happen to be taught in the worse way). And the answer we would have gained in such an approach would be, that forty years ago one teaching approach was better, what does not necessarily mean that it would be of any value now (let us think about the computer science curricula from forty years ago).

The bad news is that despite the fact that we are not in a position to definitely decide which teaching approach is better (or good), we do have to teach computer science or software engineering (and of course many other subjects not discussed in this paper). The situation is still worsening by the fact that the subject we are dealing with is quite new (compared for example with mathematics, medicine or bridge engineering) and changing rapidly. Therefore even if we are satisfied with current teaching organization we are forced to look for new solutions which will assure (or at least we hope they will) the same good level of teaching in the future.

Such a process of devising new curriculum – not because of drawbacks of the current one but because of the demands of future – is currently going on at the Institute of Informatics at the Warsaw University. In this paper we first state the reasons for carrying out the changes, then we describe the current curriculum, next we explain the rationale for the new curriculum and present its contents. We conclude with some final remarks. In this paper due to the lack of space we present only the changes at the bachelor level although the Master stage has also undergone significant alterations.

The author of this paper was engaged in the process of devising the new curriculum at all stages as the member, then the head of the institute committee for the new curriculum and recently also as the institute's vice-director responsible for teaching.

It is obvious that such a demanding and complicated task as proposing new academic curriculum requires a group of people engaged in it for a longer period of time. I would like to mention here those persons who actively participated in the process of constructing the new curriculum. First of all prof. K. Diks and prof. A. Tarlecki who were playing the leading role in work of our group, then members of the institute committee: prof. J. Tyszkiewicz (who for some time was also the head of the committee), prof. D. Niwiński, P. Chrząstowski, K. Ciebiera, M. Engel and all members of the research and teaching staff of our institute who participated in numerous discussions and prepared the curricula and syllabi for all subjects.

We will start teaching with the new curriculum in the academic year 2007/2008. This change will be synchronised with the introduction of a new curriculum by the Institute of Mathematics at our faculty.

The process of devising new curricula is of course not peculiar only to our institute, the same is being done at the European level (let us just mention the two Thematic Networks [1] and [2]) or in the USA (for example the new ACM curricula recommendations [7]).

THE NEED FOR CHANGES

As was already mentioned the areas of computer science and software engineering are evolving and expanding very quickly. This causes the need for amending the teaching contents in these subjects. Of course some small changes may be carried out on the fly when they are needed, without causing changes in other subjects. But with time the changes accumulate, so it starts to be necessary to revise the entire teaching program. Some subjects needs changing of their syllabi due to new achievements in the corresponding areas. But there are more demanding sources of changes. New subjects arise, which earlier did not exist or were not considered important enough to have their separate lecture in the curriculum. For example object-oriented programming which is currently the dominant paradigm for programming, although was invented in 1960s (Simula-67) was not especially popular for the next twenty years and as a result also not taught at many universities. Nowadays it is not possible to consider a software engineer education complete without good knowledge of object orientation. As other examples we can mention networking, webservices, security issues etc. Hence it seems quite natural for such a rapidly changing area of knowledge to examine the curriculum every year to analyse if there is a need for more structural changes and to perform such changes at least once a, let us say, ten years.

But there were also other reasons for carrying such changes just now. Poland has recently joined European Union what caused many changes also in Poland's educational system. One of these is the strict division of formerly 5-years, homogeneous study into the common in Europe (and not only Europe) 3+2 study model. This change is also in accordance with the Bologna agreement [4]. At our department the division into two stages was introduced already few years ago, but know we are forced (as almost all universities in Poland with exception for some special areas as medical studies) not only to divide our study into two stages but also to make both stages accessible for candidates from other universities. This also means, that we have to prepare our students to leave our study after three years and to start work as software engineers or to apply for study at other universities (not necessarily in Poland). This is in fact very demanding task – both stages of the study must be self-contained, we cannot leave some subjects to be taught later (at the Master level).

Therefore we had to prepare new curriculum. Let us stress once more, that the need for changes was not caused by our dissatisfaction with the current curriculum. On the contrary. Our graduates have no problems with finding jobs (in fact most of our students gets employed already during their study – which has some virtues but of course also many drawbacks for the teaching process). Our students have also many significant achievements in international competitions (just to mention the first place at the 27th Annual ACM-ICPC World Finals [5] or currently for many months the first place at the TopCoder competition in the school category [6]).

THE CURRENT CURRICULUM

Here we provide an excerpt from the current curriculum. More detailed description (in polish) may be found at the department site [3]. The meaning of subsequent columns is as follows:

- Subject - name of the subject.
- Sem. - semester number.
- Lect. - the number of lecture hours per week.
- Exc. - the number of exercise hours per week.
- Lect. - the number of laboratory hours per week.
- Exam - "y" if there is an examination.

If there is no examination at the end of a semester, then other form of assessment is used (e.g. students have to write a program which is evaluated by the supervisor). There are two semesters a year, each consisting of 15 weeks. Some additional subjects (as language courses) are omitted. We also treat here all elective subjects in the same way, whereas in reality there are two kinds of those subjects (depending on the subject pool, from which they are selected). Because of the lack of space we present here only the first 6 semesters from the entire curriculum.

Subject	Sem.	Lect.	Exc.	Lab.	Exam
Discrete Mathematics I	I	2	2		y
Mathematical Analysis I	I	2	2		y
Linear Algebra I	I	2	2		y
Introduction to Programming	I	2	2		y
Fundamentals of Set Theory	I	2	2		y
Discrete Mathematics II	II	2	2		y
Mathematical Analysis II	II	2	2		y
Linear Algebra II	II	2	2		y
Programming Methods	II	2	2		y
Programming Methods – lab.	II			2	
Logics	II	2	2		y
Object-oriented Programming	III	2	2		y
Object-Oriented Programming – lab.	III			2	
Numerical Methods	III	2	2		y
Algorithms and Data Structures	III	2	2		y
Databases	III	2	2		y
Databases – lab.	III			2	
Concurrent Programming	IV	2	2		y
Concurrent Programming – lab.	IV			2	
Computer Architecture and Low Level Programming	IV	2		2	y
Software Engineering	IV	2	2		y
Software Engineering- lab.	IV			2	
Elective I	IV	2	2		y
Automata Theory, Languages and Computations	V	2	2		y
Operating Systems	V	2	2		y
Operating Systems – lab.	V			2	
Program Semantics and Verification	V	2	2		y
Elective II	V	2	2		y
Elective II – lab.	V			2	
Team Programming Project I	V			1	
Computer Networks	VI	2		2	y

Team Programming Project II	VI			1	
Elective III	VI	2	2		y
Elective III – lab.	VI			2	
Elective IV	VI	2	2		y
Elective V	VI	2	2		y

THE RATIONALE

If something is working quite good – and the current curriculum is – one have to be very cautious when changing it. As it was already mentioned, one reason for the change was the necessity of splitting our curriculum into two separate parts. So the first question was: which elements of our current curriculum are necessary for a bachelor to be able to start working just after finishing the first three years of our study? The simple answer of this question is: the practical ones. Without practical skills in programming the bachelor education of course will not be complete. But we must be very careful answering this question. Just programming is far not enough. First of all there are some underlying subjects, as databases or software engineering, which are necessary for each programmer.

But the maybe less obvious point is that only strictly computer programming oriented subjects are also not enough. Teaching at university should not only provide the students with some practical knowledge (like how to write a loop over a collection or organize index in a database) but first of all with the capability of learning. No study, course or whatsoever can provide the students with all knowledge they will need in their work. The tools of computer science are changing so quickly, that we should not only show some of them, but first of all we should prepare the students to learn them by themselves, because that is what they will be forced by the overall progress to do during all their professional life. Hence it is so important, to teach them some mathematical subjects, which will train them in logical and systematic way and to look for generality of their solutions and not for some particular properties of used constructs. Many young people tend to believe that knowing the meaning of the fourth parameter of some function – like mcrypt_decrypt in PHP – is the core of computer science. It is vital to show students that such information like mentioned above – although sometimes useful – is of very limited value. When it is needed it is easy to find in contrast for example with the understanding of the way the ciphering algorithm used by the mentioned function works.

Hence we moved only some advanced theoretical topics (like proving the completeness of first order logic) to the Master stage, leaving considerable amount of mathematics at the bachelor level – exactly because that stage has to be practical!

We have proposed a big change in the teaching of programming at the first year. Due to the fact that we have to close a stage in programming teaching within three years we had to move the introductory programming courses to the first semester (instead of two semesters in the current curriculum). The amount of hours for the introductory course remains the same. We are aware of course of the fact that preserving the number of hours while shortening the teaching period is a big change for the students (one has to get used to some things during the teaching process which requires time), but this change was needed to be able to earlier introduce object-orientation, strongly demanded by many of our lectures as a prerequisite for their subjects. It is also true that computer teaching at high schools starts to raise its level (among other things due to many competitions organised each year for pupils) and due to the fact that computers are nowadays more and more affordable our students usually has some programming experience before they start our study.

It is not stated in the curriculum itself but we will continue to use Pascal as the first programming language at our study. The first semester is used for teaching programming in the small and we find Pascal as the most appropriate language for that aim. That what

we need at this stage is just an imperative language with conditionals, loops, procedures and some data types.

It does not follow directly from studying the presented tables but the subject of Object Oriented Programming has undergone significant changes, although the number of hours remains the same. The reason for changes is the shift of it from the third to the second semester. Why is that so important? There are two factors. First is an administrative one, students who do not pass their examinations at the second year can start studying at the third year (they only have to pass the missing examination at the next year). At first year the situation is different, those who have not passed their examinations cannot study further. So the responsibility for the lecturers is much higher and the level of difficulty of presented material must be very well balanced. The other reason is closely connected to the way of presenting object-orientation in current curriculum. To put more emphasis on the ideas and not the tools, the lecture shows object-oriented programming first in Smalltalk (i.e. a pure object-oriented language) and then in C++ (a hybrid language). The languages are so different that it is much easier to show students the ideas behind used notations. In the new curriculum this approach is not possible because we think that three new languages taught during the first year may be too difficult. Therefore we have changed the program of this subject (one of the changes consists in adopting only one object-oriented language for the entire lecture).

Other quite remarkable change concerns laboratories. We divided them into two categories: those which consist in writing one large (according to the academic standards) project - and those are treated as separate subjects - and supplementary ones, which are used only to illustrate the notions shown at the corresponding lecture. The latter are part of the subject. We have added some such new laboratories (for example to Algorithms and Data Structures) to enable better understanding of the subjects taught at the corresponding lectures. Therefore although in the new curriculum there are more laboratory hours, there are fewer subjects called "laboratory".

THE NEW CURRICULUM

Here we provide the new curriculum. The meanings of columns and the general assumptions (as to the number of semesters etc.) are the same as for the previous table.

Subject	Sem.	Lect.	Exc.	Lab.	Exam
Fundamentals of Mathematics	I	2	2		y
Mathematical Analysis I	I	2	2		y
Linear Algebra I	I	2	2		y
Introduction to Programming	I	4	4	2	y
Discrete Mathematics	II	3	3		y
Mathematical Analysis II	II	3	3		y
Object-oriented Programming	II	2	2	2	y
Individual Programming Project	II			2	
Computers and Networks Architecture	II	2			y
Logics	II	2	2		y
Algorithms and Data Structures	III	2	2	2	y
Databases	III	2		2	
Operating Systems	III	2	2	2	y
Probability Calculus and Statistics	III	2	1	1	y
Programming Tools and Languages I	III	1		1	y
Numerical Methods	IV	2	2	2	y

Automata Theory, Languages and Computations	IV	2	2		y
Software Engineering	IV	2		2	y
Networks and Communications Protocols	IV	2		2	y
Fundamentals of Law and Business	IV	2			y
Programming Tools and Languages II	IV	1		1	y
Team Programming Project I	V			2	
Program Semantics and Verification	V	2	2		y
Security of Computer Systems	V	2		2	y
Elective I	V	2	2		
Elective II	V	2	2		y
Proseminar	V			2	
Programming Tools and Languages III	V	1		1	y
Team Programming Project II	V			2	
Computer Graphics	VI	2		2	y
Languages and Programming Paradigms	VI	2		2	y
Applications of Computer Science - Seminar	VI	2			
Elective III	VI	2	2		y
Elective IV	VI	2	2		y

CONCLUSIONS AND FUTURE WORK

As we explained at the beginning of this paper, one can never be fully satisfied with the current organization of the teaching process; at least because it is not possible to proof that it is optimal. Also the areas of Computer Science and Software Engineering are growing and expanding so quickly that there still will be need for improvements and amendments to the teaching process, curricula and syllabi. But nevertheless we can say that we are satisfied we the proposed new program and we are confident that it will help us to preserve the high level of teaching our students and will provide them with adequate knowledge at the start of their professional stage of life. And we know that in few years we will have to think about another change of our curriculum.

REFERENCES

[1] http://ecet.ecs.ru.acad.bg/ecet/index.php.
[2] http://ecet.ecs.ru.acad.bg/etndec/.
[3] http://www.mimuw.edu.pl/studia/dzienne/informator/mgr0506.html.
[4] http://europa.eu.int/comm/education/policies/educ/bologna/bologna.pdf
[5] http://icpc.baylor.edu/icpc/finals/2003medals-web.htm
[6] http://www.topcoder.com/tc
[7] http://www.acm.org/education/curricula.html

ABOUT THE AUTHOR

Assistant professor Janusz Jabłonowski, PhD, Institute of Informatics, Department of Mathematics, Informatics and Mechanics, Warsaw University, Phone: +48 22 55 44 484, E-mail: janusz@mimuw.edu.pl.