Teaching what they need instead of teaching what we like – the new Software Engineering Curriculum at the University of Stuttgart

Prof. Jochen Ludewig and Ralf Reißing

Institut für Informatik der Universität Stuttgart

Breitwiesenstr. 20-22, D-70565 Stuttgart

phone/fax +711-7816-355/-380

(ludewig | reissing)@informatik.uni-stuttgart.de

http://www.informatik.uni-stuttgart.de/ifi/se/se.html

Abstract

At the University of Stuttgart, a new curriculum called Software Engineering was launched in October 1996. It is offered by the Department of Informatics, and supplements the standard informatics curriculum started in 1970. The curriculum is new not only for the University of Stuttgart, but for all German speaking universities.

This article first provides some background information about German universities in general. After this, the reasons for starting a new curriculum, its goals, and its limitations are discussed. Finally, the current state and future steps are outlined.

Keywords

software engineering education, software engineering curriculum, practical approach

Background

The education system in Germany differs significantly from the education systems in most English speaking countries; one might say that there are more differences than similarities. Therefore, some basic facts about the general conditions for any university curriculum may be useful for most readers.

In Germany, only a minority (about 22 % out of one million in 1994) of the school-leavers finish at the highest level, which is required for entering a university. These young adults have usually attended school for 13 years and had to pass the final examination ("Abitur"). Most of the males also did a one-year military or social service.

The Abitur is the only requirement for entering any university; students can enrol for any major they like except for a few with limited capacity, like medicine and architecture. In computer science, access to the universities has never been restricted. Private universities

are almost unknown in Germany; all others are funded and controlled by the states (16 states form the Federal Republic of Germany) in which they are located. Students don't have to pay anything but some minor fees for insurance etc. Though there are certainly differences in quality between the universities, these differences are small compared to the differences in the US. There is no ranking, official or inofficial. Private preferences, in particular the distance from home, are the most important factors when parents and students choose a university.

Competition between universities is a fairly new idea in Germany; in former years, most universities had more students than they could handle, and some students were not accepted at the universities of their choice, but were moved to another place. Recently, the number of young people who want to become engineers and scientists decreased considerably. Though computer science was not affected as seriously as traditional engineering, in most universities the capacity in our field exceeds the demand. Since the governments cut down funding severely, the faculties try to prove their importance by stable numbers of newly enrolled students.

While we have incorporated many English words into our language (e.g. "software"), "computer science" is neither used as a foreign word, nor directly translated. Instead, we use the word "informatics" ("Informatik"); in the sequel, "informatics" and "computer science" are treated as synonyms.

All German curricula in informatics are based on the same structure (fig. 1). After two years of studying, there is a set of examinations called "Vordiplom" (pre-diploma). After another two years, students should have finished all their exams. The final semester is to be spent on the diploma thesis. Then, after 9 semesters, the student will receive a diploma, which is equivalent to a master's degree. Note that the average career of our students is far more chaotic than fig. 1 suggests. Deviations occur due to individual reasons, sometimes also due to organizational deficiencies in the university.
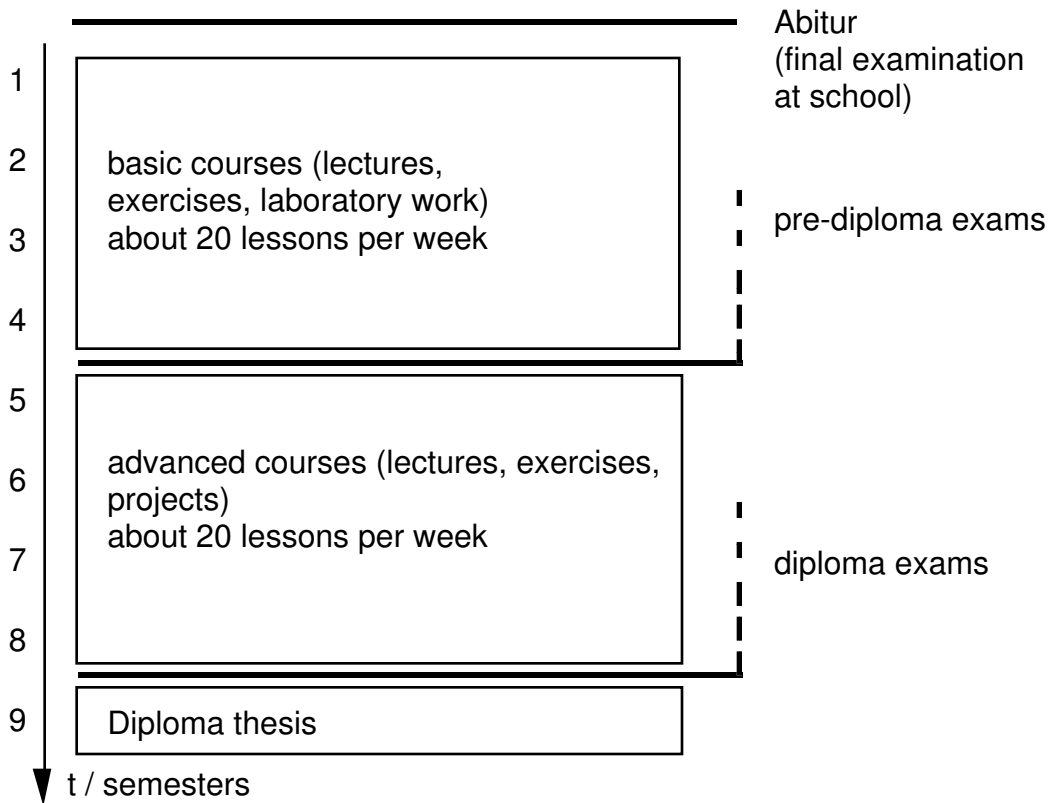
Fig. 1: General schema for an engineering or science curriculum in Germany

At first glance, the pre-diploma might be confused with a bachelor's degree. But a pre-diploma is in fact something very different (see fig. 2). It is just the entrance to the second half of the curriculum, and has no meaning outside the universities. Nobody enters a university with the intent to leave after the pre-diploma exams. Therefore, the courses before the pre-diploma do not not have to constitute a complete education. They rather contain all the basic and theoretical courses which provide the foundations for more specialized courses to be taken in the third and fourth year. There are no undergraduates and graduates in our universities. There are students who have already passed their pre-diploma (which is regarded as the harder part) and others who have not.

master's
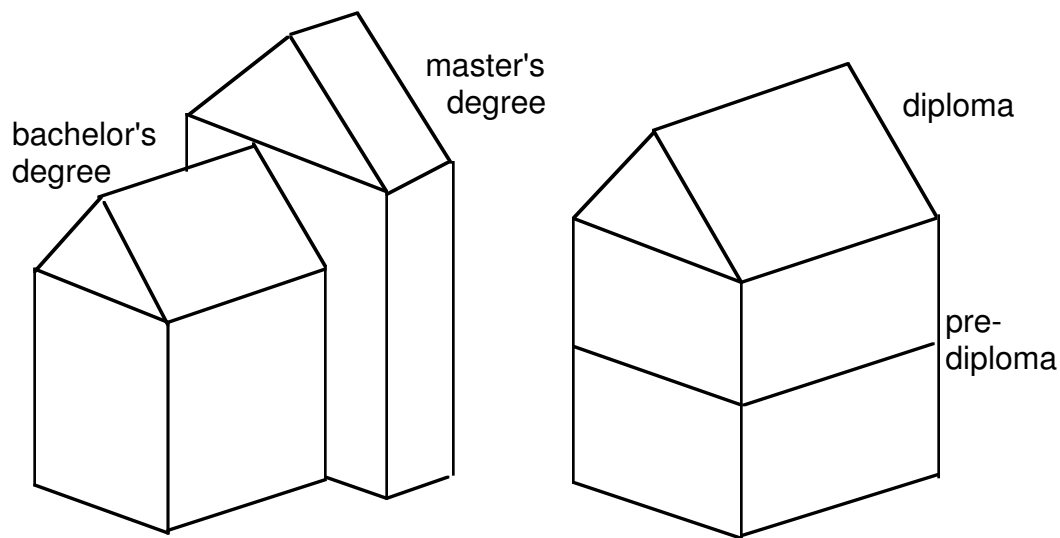degree

bachelor's
degree

diploma

pre-
diploma

Fig. 2: The difference between a bachelor's degree and a pre-diploma

Note that there is a new tendency in some German universities to offer a bachelor degree as well, in order to attract more foreign students. This is currently (i.e. since 1997) a highly controversial issue.

Students get comparatively little guidance and no continuous attention from the faculty. They have to find out which courses they should take, which lectures are worth attending, which libraries provide the books they need. Examinations are usually independent from the courses; students may register for an examination right after the course or later, maybe years later. So they may postpone them if they want to. They must meet a few deadlines, intended to prevent the so-called "eternal students", but there are still a couple of students around who have been enrolled for more than 15 years. Students who run into trouble, but do not actively seek help, don't get any support. The dropout quota is high; in informatics, about half of those that once started will actually get the degree. The average time from enrolment to the diploma is about 12 semesters.

In Germany, evaluations are as new as competition is. In our informatics department, we have volunteered to be evaluated in 1996. Now we have got more figures and explanations for our situation. We know for instance that most students have a job which may consume more of their time than studying does. Many of those who never make it to the degree do not fail in their exams, but are eaten up by the businesses they run. After postponing the exams a couple of times, they lose contact, and eventually drop out.

The effects can be summarized as follows: Students who receive a degree from a German university are comparatively old (about 26). They have managed to survive under very tough conditions. They have learned to organize their work and to ignore problems of lesser importance. For the really good ones, the German system is an attractive challenge, in which they can, and do, prosper.

The Informatics Curriculum: Goals and Principles

In many universities, informatics is taught very much like a mathematical science, i.e. as a collection of theories, analytical methods, and formal notations. These topics are obviously important for our profession. But most of our graduates find jobs in which they build new software systems or modify existing ones. They are doing the same kind of work engineers do, though with different goals and materials.

Therefore, the constructive aspect of informatics is more important for them than the analytical aspect. In other areas this distinction has eventually ended in a separation of fields: Maxwell's equations are fundamental both for physicists and for electrical engineers. But the physicist uses them in order to understand, while the electrical engineer applies them (usually in some simplified shape) in order to build something (see fig. 3a)
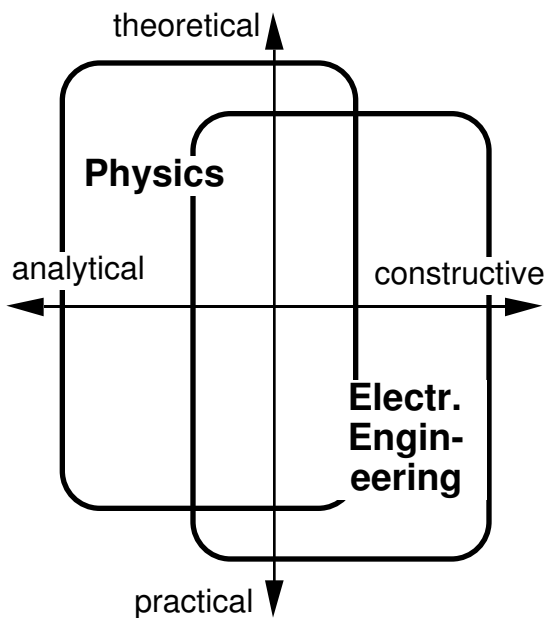
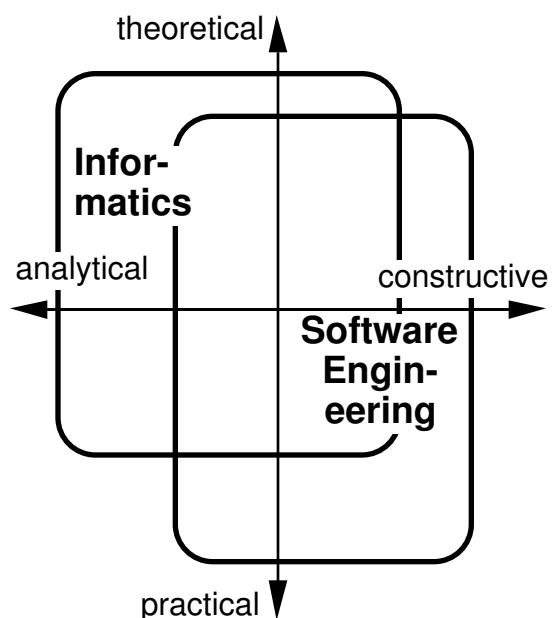Fig. 3a   Electrical engineering versus physics

Fig. 3b   Informatics versus software engineering

In informatics, we have not yet reached the point for such a clear separation. But our new curriculum is an attempt to give an example of what such a "constructive informatics" might look like (fig. 3b). It is still full of compromises, mainly because there are simply not enough faculty members to start from scratch. But the general structure seems to be sound.

When the idea of a software engineering curriculum was first discussed in 1995, there was also the alternative idea of a joint curriculum in cooperation with some engineering department. We do not believe in such a concept. For those we teach, deep understanding and extensive knowledge in one area, combined with the ability to open the doors to the neighbors, is far better than a quilt-like education, which is in any particular area inferior

to what specialists know, or can do. There are thousands of scientists and engineers who became full time or part time programmers. If those with a degree from an informatics department are not discernibly better in producing software than these "immigrants" are, there is no need for informatics.

Outline of the Software Engineering Curriculum

Fig. 4 shows a (radically simplified) overview of the software engineering curriculum. Provided there are no special obstacles, students will usually try to follow this schedule for the first four semesters, i.e. until the pre-diploma. From that time on, they choose their individual paths.

	Mathem. and econom.	Theor. CS	Introduction to informatics	Technical CS and exercises		
1. sem.	5L 2E Adv. Math. I	3L 2E Logic	4L 2E Introd. I	4 P Program-ming	2E Eng-lish	24
2. sem.	5L 2E Adv. Math. II	3L 1E Theor. CS I	4L 2E Introd. II	3L 1E Techn. Inform. I	6L Funda-men-tals of tech-nolog.	23
3. sem.	4L 2E Eco-nomics	2L 1E Theor. CS I	3L 4P Introd. III	3L 1E Techn. Inform. II		22
4. sem.	2L 1E Statistics	3L 1E Theor. CS I	4L 2E Introd. IV	4 P Basic SW project		19
						88
5. sem.	16 Main courses (4 out of 5, always including SE)	14 Project 1 in informatics	12 Application subject			19
6. sem.						19
7. sem.		14 Project 2 with legacy software	14 Project 3 in the application area			21
8. sem.	10 Arbitrary courses in CS					21
						80
9. sem.	20 Diploma thesis					

examination or proof: certificate ▪▪▪▪▪▪▪▪   written or oral examination ▬▬▬   mark on performance

Fig. 4   General outline of the software engineering curriculum.
Shaded areas are project work (including the one-person project diploma-thesis.

The numbers in the boxes are quantities. "L" stands for "lecture", "E" for "exercise", "P" for "practical work"; the unit is in any case a lesson of 45 min. The figures on the right hand side indicate the workload in each semester.

For instance, "3L 2E Logic" (in the top line) means that during 15 weeks (a winter semester), each week 3*45 minutes are spent on lectures, another 2*45 minutes are spent on exercises.

A new Course in Theoretical Computer Science

New courses in theoretical computer science are important components of our software engineering curriculum.

The education in mathematics, theoretical computer science and other theoretical areas, as offered by our universities, is credited highly. While no one in industry wants us to extend these courses, very few want them to be reduced. Sound theoretical foundations are appreciated very much.

Most students do not like formal theories, but study them in order to pass their exams. Afterwards, they soon forget most of it. Therefore, most of them will never learn how practical and useful theoretical computer science is. When questions are not asked in a formal context, but are taken from an application area, very few students can identify the underlying structures they learned in theoretical computer science. On the other hand, they are hardly able to name a single practical use for any of the theories. theoretical computer science and practical work seem to be different areas having practically nothing in common.

We change that. We want neither more, nor less theoretical computer science than we have in the informatics curriculum, but a *different* theoretical computer science, a "practical theoretical computer science". What could such a theory look like?

Traditional lectures and books contain many theorems, each followed by a proof, not by its typical, or most important, applications. This contrasts sharply with the actual needs: Except for some professors and researchers, the *only* purpose of a theorem is its application; it is sufficient if a student has seen a few proofs and understands the principle of proving. There is no need to prove dozens of theorems. But there is an urgent need to demonstrate for every theorem its particular power in practical applications. Theorems whose power cannot be shown should not be treated. theoretical computer science is *not* an end in itself, at least it is not for a software engineer. It is a useful aid in solving real problems. Therefore, all courses in theoretical computer science have to be taken before the pre-diploma.

Projects and the Application Area

There is no doubt about the importance of practical experiences. This holds true, of course, for all engineers. But in software engineering, practical experience is not always as

beneficial as it should be, because many companies do not demonstrate *good* software engineering. In such an environment, all a student can learn is that software engineering is hard, often too hard for those who never learned it properly. When students are hired for their first job, they are often not expected to continue learning, but to teach their colleagues.

If we want to make sure that students get a chance to apply what we teach, thus strengthening their new knowledge, we must offer much practical work *within* our curriculum. The most important type of such work is the student project.

Since our students study – at least – for two years between pre-diploma and diploma thesis, we require them to participate in three different projects. These overlap, because each takes 12 (or 18) months. Projects are performed in groups of 6 to 10 students.

What is the most important aspect of a such project? Is it the goals? The organisation? The people? The time schedule? Well, all these components are important. But, at least for a project in software engineering, most important is the existence of a customer, who defines success and failure. An average customer does not understand the software people, and vice versa. Customers often change their minds, are not willing to read formal descriptions, and argue about whether or not the product meets the requirements. Any project without such a customer is much too far from reality to be useful. The worst – but fairly common – situation occurs when students are their own costumers. How can they learn to communicate about requirements with a customer who does not speak the informatics lingo? Who teaches them to deal with unexpected changes? Where do they experience that the primary purpose of a piece of software is to please the customer, not the developer?

Project 1 is an academic project, with strict project management, organized within the department of informatics. The customer will usually be a professor. Tutors make sure that students follow the process and reach their goals. After project 1, the participants have learnt that there is a path – which is not always smooth – from the problem to its solution. They have experienced the effects of poor documentation and insufficient communication, and they have seen software quality assurance from both sides. They have given presentations and written reports. And, in contrast to projects in industry, they have had the opportunity to reflect on, and discuss, all difficulties.

In the second project, legacy software is the additional hurdle. We do not know of any informatics curriculum in which legacy software plays a significant role. But most tasks in the world of software deal with legacy systems. Therefore, project 2 is on a real problem which includes legacy software, which may be re-engineered, replaced, modified, or supplemented. In addition to the duties mentioned for project 1, students are involved in

the organizational tasks like planning and project management. Since project 2 deals usually with a real problem, there is also a real customer (e.g. a bank, or an industrial company), who should pay for the software in order to stimulate his or her attention.

After the pre-diploma, every student has to choose an application area. As the name indicates, an application area is a field where software is applied, like robotics, traffic planning, and economics. If another department agrees to offer a couple of introductory courses and some projects to our students, we have got another application area.

If a student chooses, say, economy as his application area, we do not want him or her to become a second class economist. We want the students to leave the comfortable playgrounds they have been living in for years. They are supposed to enter a different world, where they have to understand the problems of those who do not (necessarily) understand the software solutions. Some students will stick to that area, while others will find their jobs in different fields; this is not our concern.

In project 3, the application project, students must combine their abilities in software engineering with their knowledge about the application area. This project will usually be the one which is closest to industrial reality. In these projects, the informatics department is involved only when the project is planned and outlined, because all projects must undergo some inspection before they are officially announced.

Other Features

We believe in programming as an important craft which is, and remains, very important in all areas of software engineering. Therefore, we have added two new programming courses (first and third semester). These courses pave the way for the projects following after the pre-diploma.

Business economics is not only an important application area of informatics, but also vital for producing, comparing, and buying software. We have integrated a course in business economics in the third semester.

The English language is extremely important for those working on software: Many manuals, almost all international conferences, and all programming languages are in English. We have our students participate in a TOEFL test (Test of English as a Foreign Language); they must achieve a minimum of 520 points. (This requirement is suspended until we can provide courses in English.)

In order to gain some basic insight into the terminology and methods of engineering, students should attend some introductory courses which are given in the engineering departments (no examinations!).

Many problems in software engineering are hard to understand for those who have never experienced the life outside the university. Therefore, we require our students to spend some time on jobs in industry, at least four months altogether. We do not have detailed requirements about the type of work our students do. It is the social experience what counts most.

Where are we today?

The concepts of the software engineering curriculum were settled in March 1996. They passed through all levels of administration of our university within three months, which is an incredibly short time for all the necessary steps. The Minister of Education finally agreed in July 1996 and allowed us to enrol up to 60 students. Such a limitation is unusual in informatics, but it is typical for a so-called "Modellstudiengang", an experimental curriculum. When the first students will have completed their studies, i.e. after five or six years, the achievements will be examined, and the curriculum is either terminated or changed into an ordinary curriculum, possibly with some modifications. In this case, we will probably shift the traditional informatics curriculum towards a more analytical profile, thus offering two truly complementary curricula.

Since we could not beat the drums before July 1996, we did not expect to find many students, and no minute was wasted to prevent overbooking. But when the curriculum started in October 1996, 76 students were enrolled – We had exceeded our limit. In 1997, 55 new students followed.

The new curriculum emerges like a road being built for a the vehicle that is slowly moving towards the construction site. A small committee makes sure nothing is missed and the necessary courses are actually planned.

Response from industry is enthusiastic. Projects, team work, better training in presentation and communication, courses in business economics and English, experience with legacy software: That is music to the ears of those who have to recruit employees for jobs in software engineering. The "Foundation for Science in Germany", the national sponsoring organisation of the German industry, has awarded a 75'000 DM prize (about 44000 US $) to the software engineering curriculum, a very welcome contribution.

Many universities in Germany have asked for information about the software engineering curriculum; some have stated their intention to start something similar. One of the most convincing arguments is the fact that we have attracted many students. No other informatics department has had a similar boom.

Acknowledgements