A flexible curriculum for computer science undergraduate major

Aleardo Manacero Jr., Rildo R. dos Santos, Norian Marranghello, Aledir S. Pereira, Adriano M. Cansian and José C. L. Ralha

Abstract—*This paper describes an innovative approach to establish a CS curriculum, aiming flexibility and minimization of the time spent in the classrooms. This approach has been developed at the Paulista State University - Unesp - at São José do Rio Preto, and is producing very interesting results. The load reduction is achieved through a series of fundamental core and breadth courses that precede depth courses in specific areas. The flexibility comes as a side effect of the depth courses, which can be adapted without any changes in the core courses. In the following pages we fully describe our motivations, actions and results.*

Index Terms— *Computer Science Curriculum, Flexible Curricula*

Introduction

Every person involved with computers and/or computer science (CS) knows that the body of knowledge in this area changes at impressive speeds, both in hardware and in software fields. As an example, ten years ago computer networks were used basically for file transfers and remote login sessions, being restricted mostly to university campuses. Nowadays we not only have the World Wide Web but also the Internet is accessible by several millions of people. This kind of evolution demands that CS programs change their curriculum at the same pace, in order to keep an adequate level in the professionals graduated from their classes. However, every curriculum becomes a giant piece of knots just after few changes, difficulting any later modification. In order to overcome this problem it is mandatory that a curriculum has to be flexible enough to absorb changes at any rate.

Another large demand that should be attended by CS programs is the gap between what is taught at academia and what is needed in the businesses. This gap can be reduced through design assignments more oriented to the technologies under use in the enterprises. However this demands an improvement in the time spent by the students in practical tasks. This could be achieved through the reduction in the time taken by regular classes, which are usually high in the Brazilian undergraduate system (over six classroom hours per day).

At this point it is desirable to quickly describe the Brazilian standards for naming majors related to computing. The government defines two different groups of majors in the computing field, one that comprises those that take the computer itself as their goal and another comprising those that have the computer application as their goal. In other words, we classify the BSc majors named **Computer Science** and **Computer Engineering** in the first group, where the development of computers or computing technologies are their goals. On the other side we put in the second group the BSc majors on **Information Technology** and a pre-service **Teacher Education** major on computing, where the use or development of solutions for non-computing problems are their goals. The curriculum described here falls in the first group.

This curriculum was established with the goals of flexibility and practical works in mind. It was developed during the reform process of the CS major offered by Paulista State University - Unesp - at São José do Rio Preto. This process took five years to be concluded, going through two unsuccessful attempts before completion. In the first two versions the curriculum adopted a standard approach, with many courses and few opportunities for changes. The third and final version received university approval late in 1996, being implemented for the 1997 class.

In the following pages we describe first the former curriculum and the history behind the reform process. After that a thorough description of the new curriculum is given, altogether with results and conclusions drawn from its implementation.

Motivation and historic

The history of this curriculum starts in 1991, when a group of faculty members decided that what they were teaching was too outdated. They assembled a task force in order to study several different curricula, including ACM's [1] and its Brazilian equivalent from the Brazilian Computer Society (SBC) [2]. The task force suggested two different curricula in 1992 and in 1994, but both were dismissed after preliminary discussions. In the following paragraphs we describe the motivations for change and what led to the current curriculum in a historical perspective.

Why change

The main reason for the creation of a task force intended to the modification of the original curriculum was a disagreement about the orientation between those who were teaching at that time and those who developed the curriculum. The original group had mostly a mathematical background while the new faculty members came from engineering programs. This difference originated a discussion about how much time should be spent in each component of the curriculum (mathematics, computer core and computer technologies). This discussion took almost five years to come to an end.

All are with Department of Computer Science and Statistics - Unesp, São José do Rio Preto, Brazil. E-mails: aleardo,rildo,norian,aledir,adriano,ralha@dcce.ibilce.unesp.br

0-7803-6669-7/01/$10.00 ©2001 IEEE

Indeed, although our former students were well recognized by prospective employers and graduate programs, that curriculum needed urgent modifications in order to have a better balance between core and technological courses. Its major flaw was an excessive load on math courses, which took a large amount of time, to the detriment of technological courses. Besides everyone certainly assures that math is important, the problem in those courses was that their contents could be easily packed in less time than what was reserved to them. Table shows the load distribution for that curriculum, spread along math, computer core, computer professional and design courses.

Area	Percentage of time
Math	48.3
Computer core	14.3
Computer professional	28.6
Design	8.8

TABLE I
LOAD DISTRIBUTION IN THE FORMER CURRICULUM.

Another major flaw in that curriculum was that it did not give the exact importance to some emerging technologies, such as computer networks, or put some others too late in the courses schedule, such as operating systems. Those details were a direct consequence of the mathematical background of the curriculum developers. At that time, the task force started the work developing a new curriculum following a more technological orientation. This orientation was kept through the three versions presented to the departments in charge of the CS major. Here we have to explain that our university follows a hierarchy where majors are offered by colleges and some of the departments inside a college have the duty of manage the major and its courses.

What had to be changed

As stated earlier, the previous curriculum had several critical points that had to be changed. Most of them were related with the unbalanced load distribution between professional and math courses. Others were related with the modernization of professional courses, bringing up recent topics that were not covered by the older courses. Table lists critical issues with some of the courses in the former curriculum.

From Table it is easy to notice that the math courses presented an excessive load to the students. We have already mentioned that this load was inappropriated mostly due to the time spent in the classroom and not by the content taught in each course. As an example, the calculus course was spread along two years, compromising 300 hours of time, but did not cover differential equations. In the new structure it covers every topic of the old structure plus differential equations, but in just three semesters and 240 hours.

In the technological courses the problem was their expected

Course	Problem
Calculus	Excessive load and lack of certain topics
Linear algebra	Excessive load
Algebra	Excessive load and content misdirection
Physics	Coming late on the program
Introduction to Programming	Inadequate contents
Compiler Design	Coming too late
Operating Systems	Coming too late and lack of design
Computer Networks	Was optional

TABLE II
CRITICAL ISSUES OF SPECIFIC COURSES.

schedule, with courses coming too late in the program in some cases. That made almost impossible to provide in-depth studies (through follow-up courses) on the topics covered there or even specific design courses. As an example, *Compiler Design* was scheduled for the last semester, which prevented offering an in-depth course about non-conventional compilers.

The problems with core courses were similar to those with math courses, that is, their complexity demanded less time than what was devoted to them. With a careful reformulation the core courses could be taught in less time or could have its contents augmented if we kept the same load. We choose the later solution for the new curriculum.

The early proposals

The first attempt for a new curriculum aimed a reduction on the amount of time spent studying mathematics, without any prejudice on its contents, and an increment on the time spent on professional and design courses. It never got an unanimous approval, even inside the computer science department, because it introduced too many new courses, demanding an extra year for a student to fulfill the curriculum requirements. The work over this version came to an end when several of the faculty members in the task force left the university.

A new task force started trying to remove the drawbacks in the first version and proposed a shorter one in 1994. Then, the task force started a discussion process with the involved departments, which proved to be rather slow. Finally, early in 1995, this version was abandoned when a third, and more attractive, proposal came into discussion.

The third proposal is the one described in this paper and added a flexible approach that was not present in the previous versions. We will not describe the two former proposals here since they never got implemented, what turns impossible any objective comparison between them and the current curriculum.

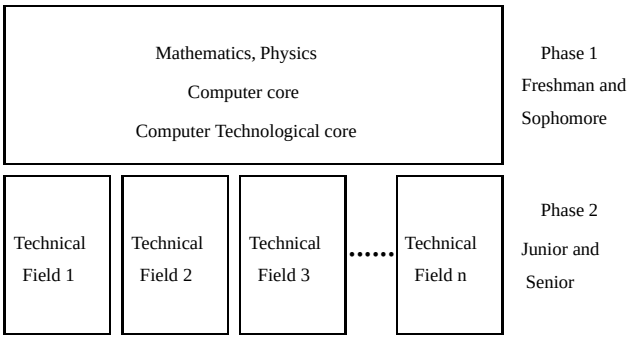

Fig. 1. Curriculum structure split in basic and technological phases

The new curriculum

The curriculum implemented has two main directives: a technological orientation and a flexible structure. The first one came directly from the original motivations for change and the second was the heart of the third proposal, coming up as a solution to avoid constant modifications in the whole structure. Some of the ideas behind the curriculum flexibilization came from the Electrical and Computer Engineering major from Carnegie-Mellon University [3], with several adaptations to the Brazilian higher education laws [2], [4] and university requirements.

The model adopted here was based on the concept that computer science is a wide and rapidly moving area and that it is impossible to know all of its subtleties. The range of topics demand many courses to cover them, turning the curriculum too large. The speed of changes age the curriculum continuously, demanding modifications too often. Finally, due to its wideness, its virtually impossible for a student to deeply understand every topic and for a graduate to work in any area since the professional specialization is almost mandatory in this field.

These concepts lead to the anchors of this curriculum: flexibility, speciality, responsibility. The flexibility attacks the aging of the course contents, avoiding constant modifications. The speciality enables that only few topics have to be examined in its gory details, while the others are examined just through their main concepts. At last, but not least, the responsibility is demanded from the students since they have to choose which topics they will study in details (although they may do this with the help of a tutor assigned by the major's council).

We introduced these anchors in the curriculum by a time division. In the freshman and sophomore years the curriculum grid is spread along mandatory math, physics, computer core and technological core courses. In the junior and senior years the grid is spread along technological and design courses grouped into areas of concentration (five in our curriculum). In Figure , we show how to perform the time division and the split in areas of concentration for technological courses during the final years.

As told in the previous paragraph, in the first two years we concentrate the mandatory courses. Although this is a commonplace in several curricula, here we introduce a set of courses classified as technological core courses, which are, in essence, the heart of this curriculum. They are introductory courses to all topics inside the technological fields covered in the final years, comprising the main concepts of each subject. This is made possible through a merge between similar topics inside a single course, like the one that links computer networks and operating systems [5].

These introductory courses are taught in the sophomore year and have three goals: give the basic concepts for most of the technological subjects, reduce the number of mandatory courses, and give the guidance to the students for choosing the field of computer science in which they will specialize. The third goal is a consequence of the responsibility anchor, since we cannot ask that students have it without giving them the support to do so. The first will be justified in the next section. Finally, the second goal is intended to the reduction on the time that students have to spend in classrooms, leaving spare time for laboratory work, library studies and leisure.

With this structure we are also capable of modifying parts of the curriculum without changes in the whole piece. This means flexibility since any subject can be inserted or removed from the curriculum through just two steps: a minor modification in the introductory course that approaches the subject's field and its own insertion or removal. Therefore, the grid of courses becomes flexible and easily manageable by the faculty and by the students.

Curriculum grid

The grid established for this curriculum is eight semesters long, with all but three mandatory courses spread along the first four semesters, as shown in Figure . Two of these three courses are in the fifth (*Linear Algebra*) and sixth (*Graph Theory*) semester just to get a better load balance along the curriculum. The last one is a capstone design course intended to be executed during the senior year.

The topics taught in most of the mandatory courses are the conventional ones, such as *Calculus I*, *II* and *III*, *Physics I* and *II*, and *Digital Systems*. The differences appear in the courses named "*Foundations in ...*", which comprises the technological core courses (one for each concentration area) and a special freshman course (*Foundations in Computer Science*) intended to make students recognize that math is there because it is necessary to get a better understanding of computer science technologies. A short description of the Foundations courses is given now.

- **Foundations in Computer Science:** an introductory course that shows what computer science is and the mathematical concepts need in technological fields [6];
- **Foundations in Languages and Computing Theory:** a course that presents the concepts of algorithm complexity and computability, formal languages and automata, and language

Semester 1	Semester 2	Semester 3	Semester 4
Calculus I	Calculus II	Calculus III	Probabilities
Analytic Geometry	Linear Algebra	Numerical Analysis	Foundations in Scientific Computing
Technical Writing	Formal Aspects of Computing	Digital Systems	Foundations in Computer Systems
Physics I	Physics II	Data Structures	Foundations in Information Systems
Programming I	Programming II	Foundations in Languages and Computing Theory	Foundations in Automation Systems
Foundations in Computer Science			

Semester 5	Semester 6	Semester 7	Semester 8
Linear optimization	Graph theory		Capstone Design
+ 4 courses	+ 4 courses	+ 4 courses	+ 2 courses
The 14 courses listed for semesters 5 to 8 refer to a set of 6 mandatory courses indicated for a specific area, 4 elementary courses and 4 courses of free choice			

Fig. 2. Suggested grid for this curriculum

paradigms;

- **Foundations in Scientific Computing:** a course that presents the hazards and solutions related to the application of computers into scientific problems;
- **Foundations in Computer Systems:** a course that discusses systems that control computers and serve as interfaces between hardware and software, such as operating systems [5];
- **Foundations in Information Systems:** a course that deals with the treatment of information inside the computer and techniques to develop solutions for user applications;
- **Foundations in Automation Systems:** a course that presents topics related to hardware, mainly those on digital systems design and industrial automation.

These courses covers all the main technologies in computer science. The topics inside each one were chosen due their correlation. For example, we put language paradigms together with computability since they have a set of formalisms oriented to proofs in common. None of these topics can be deeply covered here in order to accommodate all of them. The density in each topic is defined by its relevance and by the understanding that if a student will need a deeper knowledge on a topic he/she will have it in the junior/senior level courses.

After the completion of the mandatory courses the student has to choose an area of concentration, following one of the kernels provided by the Foundations courses. This is required to fill in the grid for the junior and senior years and is intended to allow the definition of the minimal set of courses that have to be taken by the student.

This minimal set comprises six mandatory courses about subjects inside the chosen area, four courses taken from a list of elementary courses, one from each one of the other areas and four more courses chosen from any technological subject.

The six mandatory courses are designed to provide an early specialization for the student. They are separated into three elementary and three advanced courses. The elementary courses covers the main subjects inside the area, like *Operating Systems Design*, *Computer Networks* and *Concurrent Programming* in the computer systems' area. Table lists the mandatory courses in each of the five concentration areas. Notice that some courses are mandatory in two areas, being classified sometimes as elementary in one and as advanced in the other.

The four courses taken from the list of elementary courses are required to provide in-depth knowledge at least in some of the subjects inside the areas that did not attracted the student. This requirement avoids an ultra-specialization that could come from the freedom given to the students about what courses they will take. The courses marked with † in Table are the elementary ones on each area. Notice here that if a course is mandatory for two areas, it cannot be picked as the elementary course for a student completing the area where it is classified as advanced.

Finally, the four courses taken from any technological area are an opportunity either for diversity or for more specialization. They also provide room for experimentation with new subjects, before any decision about its insertion as a mandatory course in one area is done.

With all of these choices made available, we demand from the students some degree of responsibility, first in the transition between sophomore and junior level, when an area of study has to be defined, and later in the choice of the eight courses that are not mandatory for his/her area. This responsibility is partially shared with a tutor, who is assigned to the

Area	Courses
Languages and Computing Theory	- Automata and Formal Languages † - Compiler Design † - Algothims Complexity † - Computability - Design of Programming Languages - Topics on Programming Languages
Scientific Computing	- Numerical Software † - Nonlinear Optimization † - Numeric Resolution of Differential Equations † - Finite Elements - Numerical Resolution of Partial Differential Equations - Topics on Scientific Computing
Computer Systems	- Operating Systems Design † - Computer Networks † - Concurrent Programming † - Real-Time Systems - Computer Architecture and Organization - Topics on Computer Systems
Information Systems	- Database I † - Software Engineering † - Information Organization and Retrieval † - Database II - Design and Implementation of Information Systems - Topics on Information Systems
Automation Systems	- Assembly Languages † - Computer Architecture and Organization † - Systems Modelling † - Systems Simulation - Microprocessors and Microcontrollers - Topics on Automation Systems

TABLE III
LIST OF MANDATORY COURSES ON EACH CONCENTRATION AREA

student as soon as the area definition is made. The tutor is usually a researcher in the chosen area, acting as a curriculum advisor.

Results achieved with this curriculum

Despite that the first class officially enrolled in this curriculum started in 1997, we allowed that students of the 95 and 96 classes could integralize the new curriculum with some adaptations. This decision led to the adoption of the new curriculum by 100% of the 96 class and around 60% of the 95 class. The consequences were an early dismissal of courses that existed only in the old curriculum and an early offering of some courses introduced by the new one. This situation became useful by enabling an early development of all modifications planned for this curriculum.

At the time that this article was written, most of the adapted classes and some of the students in the 97 class have graduated, completing the curriculum cycle as it was planned. From assessment of these classes, we have obtained interesting results, proving some of the predicted remarks and providing some insights about flaws in certain situations. We start listing the identified problems:

(a) Need for more design works, especially in the "Foundations" courses;

(b) Need to strengthen the requirements for approval in the core courses;

(c) Need to change the schedule of some courses in the grid (the *Formal Aspects of Computing* course, moved from the first to the second semester);

(d) Need to add a pre-capstone design course.

The first two problems are already under work, resulting in an increase of the retention rates in the core courses and in an increase in the level of knowledge achieved after approval on them. The last problem will be dealt during the curriculum reform that will be necessary to be performed in 2002, due to modifications in the Brazilian higher education laws. As an

aside comment, the curriculum presented here accomplishes almost all requirements of those laws and will need only the addition of few more hours of load.

Beyond these problems several favorable conclusions can be drawn. They are:

(a) Decrease in the average turnaround time for graduation, which was 4.8 years until December of 1999 and is down to 4.5 years now. This average should get lower since most of the students are now graduating in four years;

(b) Decrease in the abandon rates, mainly due to the *Foundations in Computer Science* course that motivates the need for the mathematics introduced early in the curriculum;

(c) Increase in the students satisfaction through the large range of choices they have;

(d) Increase in the quality of the capstone design works, since the students have a deeper knowledge on the field they choose to develop their work.

From all these results it is possible to conclude that this curriculum offers a better model for CS majors, since it provides flexibility for changes and the possibility for early specialization (closing the gap with the so-called real world).

The advantages provided largely overcome the problems observed, and since most of the problems have easy solutions, it is expected that they will disappear in a near future.

As a final remark we believe that this approach for curriculum implementation is strongly indicated to any major that have characterists similar to those present in the computer science field, i.e., fast modification and wide range of subjects.

Acknowledgements

We want to acknowledge the Research Support Foundation of State of São Paulo, which supported the divulgation of this curriculum, the administration of Paulist State University - Unesp, which granted us with freedom to implement a experimental curriculum, and all the students that went through the experimentation process.

References

[1] ACM/IEEE-CS, "ACM Curricula Recommendations Volume I: Computing Curricula 1991", ACM Press, 1991.

[2] Brazilian Computer Society, "Reference Curriculum - 1996", available (on 03/02/2001) at *www.sbc.org.br/cr/crf96.html* (in portuguese), 1996.

[3] Dept. of Electrical and Computer Engineering, "Electrical and Computer Engineering Undergraduate Primer", available (on 03/02/2001) at *www.ece.cmu.edu/undergrad/primer*, 1995.

[4] Ministry of Education, "Curricula Directives for Computer Majors", available for download (on 03/02/2001) at *www.mec.gov.br/Sesu/ftp/curdiretriz/Computacao/co_diretriz.rtf* (in portuguese), 1999.

[5] Manacero, A. Jr., "Merging Operating Systems and Computer Networks: Why and How", in *Proc. of Internation Conference on Engineering Education - ICEE'98*, CD-ROM, Rio de Janeiro - Brazil, 1998.

[6] Manacero, A. Jr. and Marranghello, N., "Turning Math Attractive to computer science students: an application to model approach", in *Proc. of 29th Frontiers In Education Conference, FIE'99*, v.1, pp 13B24 - 13B29, San Juan - Puerto Rico, 1999.