



Rethinking About Computer Science Curriculum – Long Overdue*

H.N. Mahabala

Indian Institute of Information Technology
1408, Second Cross, BSK I Stage,
Ashok Nagar, Bangalore 560 050, INDIA
hnmahabala@yahoo.com

Graduates as they come out of a bachelors' program are expected to be ready for employment, which implies that they have the necessary skills and exposure to key technologies of interest to employers. Whereas, when we initiated computer programs in the 60's we put together a bunch of topics as part of our academic programs. Actually, we did not have enough Computer Science (CS) topics, and we included Numerical Analysis and topics from Operations Research!

Progress in the field of computer technology as practiced in the industry today makes one wonder whether we are preparing the students for today's state of technology. Our students are given the impression that programming in a high level language is adequate as preparation for the current highly competitive computer field. It is a fact that major employers in IT in India are forced to work mostly with earlier technology and so may find our graduates (students of Botany and even house wives) suitable for employment. It was interesting that during the peak of Y2K in 1999, any graduate was given a short training in conversion, and was used to handle the flood of work. This make-shift arrangement made one wonder the relevance of academic programs in computer science!

Such a gap between academic programs and job environment must have happened in all emerging fields and sooner than later, academic programs fell into step only to slip back later! Usually, new academic programs were started to close the gap. To recall a bit of history, it seems only one general program in engineering, mostly civil engineering existed, and mechanical and electrical disciplines evolved in course of time!

Faculty tend to develop deep attachment to certain topics and start believing that they cannot imagine how any graduate can graduate without learning those topics, referred to as fundamentals. I remember introducing the Karnaugh map approach to simplifying Boolean expressions for practically, a major portion of a course in digital design. We believed Karnaugh simplification was fundamental, but as progress took place, time devoted to teaching Karnaugh's method kept reducing and computer programs based on the Q&M method made manual simplification unnecessary. Same is perhaps true of skill in manual arithmetic manipulation.

Unfortunately, the time available for teaching in an undergraduate program in CS cannot be extended. So it has become necessary, to take a look at our CS curriculum framed quite some time ago to purge obsolete topics to make room for more relevant concepts and skills, to make a fresh graduate aware of the current state of technology.

To illustrate the point: A few years back Indian curriculum had a course on operating system based on a book by Donovan, very teachable (and examinable!) material. When I met Donovan a few years later at MIT, I mentioned to him that his name was very familiar to CS students in India because his book was a prescribed text book. He blurted out "not that outdated stuff"!

We blissfully keep on teaching early concepts in memory management and processing time optimization based on elementary concepts discussed in that book, knowing fully well that no student will ever be expected to optimize using those elementary concepts in today's real life problems. They do not realize that current operating systems are way ahead. Computer hardware has broken the barrier of limitations on memory capacity, processing time, graphics, database management and bandwidth limitation, etc. If we can at least meaningfully

* Reproduced from CSI Communications, November 2009



make a student aware of the current state of computer technology, we can claim to have prepared a student for today's work environment. How do we do it?

The conventional practice of updating each course will not be very successful. We need to drop some courses and perhaps combine a few courses to make room for new topics. A more effective approach will be to collect from various categories of users the topics they feel are key to their field and prepare a content map, and then allocate them to the required number of courses. Names of such composite courses may have to be invented! Faculty will have to be trained to teach them. I remember to have taken five courses on radio technology, which was the peak of electronics in 1958. We heard just before graduation that something called the transistor was just invented! Today we have at the most one lecture on radio, if at all!

To be fair to academic people, one should recognize that attempts at revision of curriculum were being done periodically. What we need is not just a revision, but a complete overhaul! There are many attendant problems associated with recasting the curriculum, such as updating teachers, availability of teaching resources, modernizing the examination system to handle a very large number of students, and lastly, keeping in step with rapid advances in technology. We owe to incoming youngsters an education in CS synchronized with technology.

SUGGESTIONS FOR REVISED CURRICULUM

The following suggestions are made to initiate discussion. Readers are free to delete / modify any of the suggestions.

We have to make room for new topics in the revised curriculum and as such, deleting or compressing a few conventional topics will be necessary. Instead of taking a negative approach, which provokes angry protests, teachers should be more constructive. Every topic has a pedagogical value and it is hard to argue for its deletion. I would like to state what I would like to see included. Here are some suggestions:

1. Programming in C has gained importance and industry has accepted it as a mother language on which new extensions will be based. I think C has to be learnt by doing and not through lectures. We have to develop a "learn yourself" CAI for mastering C through plenty of examples and exercises on the lines of "Java in 21 days". Students can even be

certified through an online exam. It is sad that currently, the longest program a student develops is limited to 30 statements. This should change and they should work with programs with at least 100 lines, using subroutines/procedures. It does not matter if they copy the code, for in crucial software development in industry, we encourage borrowing code [Visual C++ facilitates copying (actually adapting proven code)]. No more lectures on going through syntax of C.

2. Data structures are also better learnt through CAI with plenty of examples and assignments. The course covering C can be utilized to cover data structures also.
3. Object oriented (OO) languages such as Java, C++ and C# are basically C with extensions for Object definition and invocation. Many Java courses repeat, rather ineffectively, the syntax of C! Many students honestly believe that Object Oriented Programming is really coding in Java. Strangely, many IT industries claim OO adaptation by using object syntax in a mindless way, with obviously no advantage of OO thinking. What a misinterpretation of the importance of object oriented thinking. It is better to teach Object modeling and thinking and cover necessary object syntax. Cover all OO languages by comparison. Some believe Java can be taught as the first programming language. It cannot be done because OO requires one to appreciate requirements of large software. Teach C well and get them to look at a large C system (say 1000 lines) and then illustrate how to manage that large code using OO approach.
4. We have to move away from developing software by coding. We now have tools such as .NET and J2EE referred to as application generators which help one to create industry quality software with minimum effort. One can include access to internet, password protection, GUI, etc. by just choosing appropriate "inclusion". We found that 85% of the final code was generated automatically under .NET. We have to introduce "Three- Tier Architecture" replacing the good old "Client-Server Architecture" days of developing simple applications such as "payroll" that are server – packages available to download. Internet Orientation is the name of the game.
5. Internet programming has become an essential skill viz. creating an attractive web site, use of client-end validation using javascript, use of XML, web



services, Public Key Encryption, etc. Students should create an attractive web site about themselves which can also be used to apply for jobs. Students can find out guidelines for designing good internet based applications by studying good websites such as amazon.com, ebay, etc. I think popular books for dummies are excellent self learning books.

6. I consider software testing an essential skill. I teach programming and testing concurrently, because every program has to be tested before it can be used. Students learn to create necessary test cases and should use them to test. Course should also cover performance testing and security testing; students should be introduced to software quality assurance and SW quality standards.
7. Course on databases has to cover elementary Data mining. Students should be introduced to methods for disaster recovery of data, Two Phase Commit protocol.
8. There can be no worthwhile computing without networking. Updated course in Networks should make room for some recent topics. Principles of cell phone system, ATM, Payment system, LAN, WiFi, etc.
9. Embedded systems and Robotics. The use of processors in PCs has saturated, but their use in intelligent appliances has limitless potential. Every graduate must be exposed to the areas of processor

based instrumentation, artifacts and gadgets. All electronic applications are specialized embedded systems. We should have a course on embedded systems with extensions to robots. Knowledge of embedded systems has to be part of every engineer's under graduate education.

10. Software Engineering students should study typical web sites such as Amazon.com, ebay.com, Railway reservation system etc. to pick up design clues for successful online applications, methods for receiving payments on the Internet, protecting the security of such applications. Student projects should be based on .NET or J2EE. Students seem to pick up technology aspects, but are weak in application design aspects. GUI and related navigation can be learnt only by doing. And critically studying good web sites.

In order to facilitate adoption of the new curriculum, we have to develop detailed syllabi for each new course with suggestions for assignments and quizzes. An example of how detailed the syllabus has to be can be found in the syllabi that I had the privilege to have developed, for the A and B examinations conducted by DOEACC. Some study manuals will have to be developed, because none of the existing textbooks cover the assorted topics included in the new syllabus.