Suggestions for Computer Science Curriculum at BITS Pilani

Drafted by the 2006 batch of Computer Science Association, BITS Pilani
http://www.bits-csa.org/

Thanks are due to the Task Force for Curriculum Redesign
&
CSA members who participated in the online discussions.

April 18, 2011

Courses and Electives

Drafted By: Soumyadeep Ghosh (2006A7PS018P)
Ph.D. candidate, Princeton University
Based on input from different members (past and present) of the Computer Science Association (CSA)

The changes proposed to the existing curriculum try to increase the core competency of each A7 student in the core areas of Computer Science. This document presents suggestions about how the new curriculum for Computer Science at BITS should be structured to achieve this aim.

Summary of Suggested Course Structure

Here are our suggestions about the courses that should be *compulsory* for all Computer Science students, distributed across different key areas of the field:

- **Pre-CDC requirements**: Discrete Structures for Computer Science, Microprocessor Programming and Interfacing, Data Structures (Lab and Theory)
- **Theory**: Theory of Computation, Theory of Algorithms, Programming Languages
- **Software Systems**: Operating Systems, Database Systems, Computer Networks, Compiling Techniques, Object Oriented Analysis and Design
- **Computer Systems**: Digital Electronics, Computer Architecture and Organization

Here are a list of the courses that should be offered as electives to students from A7 (some of these courses are offered already):

- **Theory**: Graph Theory, Complexity Theory, Advanced Algorithm Design
- **Security**: Information Security, Network Security, Cryptography
- **Intelligent Computing**: Artificial Intelligence, Machine Learning
- **Graphics**: Computer Graphics, Image Processing, Pattern Recognition, Multimedia Computing
- **Systems**: Distributed Systems, Advanced Computer Architecture, Parallel Computing and Data Storage Technologies & Networks
- **Others**: Software Engineering

Compulsory Discipline Courses (CDCs)

This section contains suggestions on the content of each individual course that should be a part of the A7 CDCs. The descriptions are given along the lines of the classification introduced in the summary. As per this scheme, each A7 student will be required to complete 14 mandatory Computer Science courses, a number that was presented in the original curriculum modifying proposal.

- *Pre-CDC Requirements*:
  1. **Discrete Structures for Computer Science**: The course on Discrete Structures for Computer Science should have a more elaborate discussion of the algorithmic aspects of different mathematical procedures, as well as a discussion about the aspects related to the implementation of the said procedures and structures. The course in its current form, just concentrates on the discrete mathematics, while leaving a gap between the math and its practical use.

2. **Microprocessor Programming and Interfacing**: The existing course on Microprocessor Programming and Interfacing is sufficient; however, it could be pushed to 2-1, instead of its current slot at 2-2; in order to allow for two different courses for Digital Electronics and Computer Architecture.
3. **Data Structures (may be packaged as part of A7-specific Computer Programming)**: There should be a separate course on data structures, that talks about the theoretical and implementation-specific aspects of data structures such as lists, trees, heaps, hash tables, etc. This course can be offered as part of the *proposed discipline-flavoured Computer Programming courses.* This change will allow students to build a stronger base in the theory of algorithms, through a split of the existing Data Structures and Algorithms course into two different courses.

- *Theory*:

1. **Theory of Computation**: The existing course on Theory of Computation is sufficient.
2. **Theory of Algorithms**: This course should deal with different algorithmic techniques such as divide-and-conquer, dynamic programming, greedy algorithms, etc. In addition, graph algorithms and the algorithmic aspects of complexity theory can also be discussed. Some advanced data structures such as fibonacci heaps, rank pairing heaps, etc. can be discussed in the context of advantages for an algorithm's performance. An introduction can be provided to topics such as approximation algorithms, offline/online algorithms, and randomized algorithms.
3. **Programming Languages**: A separate course on Programming Languages should be offered in lieu of a compiled course on PL and Compiler Construction. This course should teach the basics of programming language design and analysis. Different topics such as functional programming languages, object-oriented languages, type systems, abstraction mechanisms, operational semantics, safety and security guarantees provided by a language can be discussed. In addition, implementation techniques such as object representations and garbage collection can also be covered.

- *Software Systems*:

1. **Operating Systems**: The current course on Operating Systems should be augmented with a practical component, that gives an insight into the practical implementation of real-world operating systems. An ideal exercise would be a semester-long project to incrementally implement a small operating system with bare-bones functionalities related to scheduling and memory management. In case this is impractical, there should be small evaluated exercises which require students to understand and modify existing operating system code.
2. **Database Systems**: The existing course on Database Systems is sufficient.
3. **Computer Networks**: This course should not have Operating Systems as a pre-requisite. However, the course content (as described in the handout) is sufficient to provide an introduction to the area.
4. **Compiling Techniques**: This course should deal exclusively with the Compiler construction part of PLCC. The structure of the course (as it exists) can be derived exactly from PLCC, except from removal of the Programming Languages related portions. The course should contain additional details about the compiler back-end and the optimizations performed therein.
5. **Object Oriented Analysis and Design**: The new elective on *Object Oriented Programming and Design* should be made compulsory for all A7 students. In addition, elements of object-oriented analysis should be introduced in this course; and the course should focus more on the object-orient design aspects.

- *Computer Systems*:

1. **Digital Electronics and Computer Organization**: The existing course should be taught with a more A7-specific flavour. The content on digital circuit design can be condensed to comprise only half of the course; and the course can then contain more topics about computer organization.

2. **Computer Architecture**: The existing course *Advanced Computer Organization* should morph into this course.

Electives

In keeping with the proposal to offer 12-20 discipline-specific electives for each discipline, we suggest that a minimum set of the following electives (some of which are offered already) should be offered in Computer Science:

- *Theory*: Courses such as **Graph Theory**, **Complexity Theory** and **Advanced Algorithm Design** can be offered as electives. A course on complexity theory would be particularly appropriate, keeping in line with its growing importance in theoretical computer science.
- *Security*: A course on **Information Security** would introduce students to security issues in computing, communications, and E-commerce. In particular, topics like goals of security systems, the concepts of confidentiality, integrity, and availability, basic cryptology, private and authenticated communication, electronic commerce, software security, physical security, disaster recovery and reliability can be discussed as part of this course. In addition, a course on **Network Security** could deal with security over computer networks.
- *Intelligent Computing*: The courses that can be offered in Intelligent Computing are **Artificial Intelligence** and **Machine Learning**, both of which are already offered by the department along with **Data Mining**.
- *Graphics*: This area would comprise courses already offered such as **Computer Graphics**, **Image Processing**, **Multimedia Computing**, and **Pattern Recognition**.
- *Systems*: A course on **Advanced Computer Architecture** that deals with topics such as superscalar processors, vector processing, multi-processor and multi-core architectures, cache coherency, etc. would be a great option for students interested in Computer Architecture. A graduate-level course is offered in *Advanced Architecture and Performance Evaluation* right now. In addition, for those who are interested in Operating Systems and Computer Networks, a course on **Distributed Systems** can also be offered. It must be noted that BITS does offer a graduate-level Distributed Systems course called *Advanced Operating Systems*. Advanced OS is conducted in a very theoretical fashion; it progresses at a slow pace and although it has a good discussion about distributed systems, it completely misses out on other important aspects of OS due to lack of programming assignments.

  OpenMP, SSE and MPI programming could be introduced in Parallel Computing elective. The availability of SSE vector architecture in recent Intel processors presents a simple opportunity to demonstrate the use of SIMD processing. The efforts required to write code for SSE for simple assignments is minimal and such assignments could be one-night exercises.
- *Others*: Electives should also include a course on **Software Engineering**. This would be especially help students who are interested in going into the industry. It must be noted that Software Engineering is already offered as a CDC for Information Systems students. The same courses may be offered to A7 students as an elective.

In addition to these courses, we suggest that additional seminar-based courses could be offered by interested faculty members. These courses could relate to topics of special interest to the faculty (closely aligned to their research). They could be offered as 2 unit courses, and would mostly relate to discussing the latest research topics in the chosen area. Such courses would especially aid students who wish to go for graduate study after they complete their Bachelor's degree.

A major point that is often cited by foreign universities and companies is that students from Indian Universities often have the same profiles (with the same courses on their transcripts), which makes it difficult for examiners to distinguish between two students. Offering a number of electives from different areas can help A7 students make a conscious effort to augment their interest in a particular field with more knowledge, while at the same time providing evidence to graduate schools and recruiters of their interests.

Additions

Drafted By: Swapnil Ghike (2006A7PS487P)
Masters candidate, University of Illinois at Urbana-Champaign

We would like to suggest a few things which we believe will help in increasing the impact of the proposed curriculum changes. This document has been drafted after having discussions with some CS/IS alumni pursuing higher education or jobs in computer science related fields. We believe that the suggestions contained in this document are a result of the combined experiences of computer science enthusiasts from the recently graduated batches.

- **Introduction to general tools and techniques in the first/second year.**
  1. Often BITS students fail to work out smart solutions to easy enough problems (which could be as simple as taking automatic backup of files). Most students don't look beyond C/Java programming to obtain quickfire solutions. Students should be introduced early to techniques like bash scripting that could be used to express solutions quickly. It is not necessary to teach such techniques thoroughly, even an introductory exercise will provide impetus to interested students to explore further. Early introduction to the fact that programming/scripting can yield smart solutions will encourage students to look at CS as fun.
  2. Introduce vim/emacs at an early stage, highlighting the really cool features of these editors. I believe that a good editor is a must to develop a liking towards programming in general.
  3. This suggestion is due to the observation that a larger proportion of undergrads outside India tend to develop a liking towards some stuff in CS, whereas our students largely look at CS from the perspective of scoring grades or as a way to obtaining secure jobs. There are hardly a couple of students in the whole batch who attempt to compile their own kernels or play with system libraries even when they are in 4th year. Perhaps they will enjoy the process more if they could be introduced to the fun part of CS early.
- **Dealing with large amount of code.** When it comes to 'hacking' some code together in one place, our students miserably fail. In general, I feel that the lack of geek quotient in our students renders them less amenable to dealing with large amounts of code. For example, I still have problems reading and writing Makefiles which becomes a bottleneck for me in understanding relevant parts of an open source software quickly. Our state-of-the-art is that many students don't know how to edit/save their files using vi editor in DSA lab tests. There should be a rigorous exercise on program compilation and debugging, which will expose the weaknesses of students in all aspects of coding like use of editors, understanding of programming languages, navigating code, figuring out library dependencies, issues with environment settings, scripting, systematic debugging, etc.
- **Introduction to SVN.** This point is separately mentioned because our students dont know how to systematically organize their code files. A very few of them also commit the grave mistake of deleting their whole programming assignments by mistakes due to carelessness and lack of proper backup systems. I have suffered from this tragedy and it was definitely a very painful experience. Introduction to using subversion systems is a must and it is not at all difficult.
- **Availability of printers.** We need more accessible printers that work with fast systems. Perhaps every student can be given an account and charged for the prints he/she issues via the account.
- **A message-passing cluster in the campus.** This cluster could be used for various assignments in networks, parallel computing, advanced OS. It could also be used for many research projects by faculty. I think this suggestion is not outlandish because most IITs already have such clusters. In many cases, it is built with the help of students. For instance, at my current university (UIUC), handful of undergrads of 3rd and 4th year built a 400 Teraflop cluster out of only GPUs with the help of professors and researchers in one semester.

- **Emphasis by faculty on completing the projects and publishing the results.** We have often seen that many formal projects or even the projects started by faculty dont complete or dont yield any results. While we agree that every project cant yield a tangible result or can run into unforeseen problems, we believe that there is a scope for a better approach to dealing with projects in general. There should be an emphasis by faculty on completing the projects in a finite timeframe, and efforts must be made to publish the results in a reputed conference/journal. This will help the participating students gain knowledge of the cutting edge research around the world in areas related to their respective projects, will drive them to rigorously figure out solutions and present and debate them with a concrete understanding. The more rigorous nature of projects and theses will help students who will opt for jobs as well as students opting for higher studies.

- **Defence of theses.** The general feeling about theses in Pilani campus is that they are less stringent that they ought to be. We have expressed our opinions about this in a separate section in this document. One suggestion is that if the thesis is defended by the student in front of a committee consisting of 3 or more faculty members, perhaps it will make sure that the students do not fall prey to slack and they will work on thesis projects more sincerely. Also, it will ensure that the thesis is criticized by the faculty working in difficult areas of computer science. An early feedback from the committee can also provide good suggestions to the student and his/her thesis advisor and it can provide better opportunities for interdisciplinary research within CS.

- **More projects on commercial technologies.** There is a need for there to be more projects in courses, pref. using Open Source software. E.g: Hadoop was used in Data Mining course last semester. The course can ask students to work with Open Source software while Computer Science Association can help build a community around it amongst students.

- **Alumni involvement in courses.** There are alumni interested to take up credited courses for A7 students who have discussed the same with Group Leader before. Such initiatives can be started after confirming the availability and motivation of instructors and students.

Restricting PS/TS to discipline: Pros and Cons (and other points about Thesis)

Drafted By: Vineet Pandey (2006A7TS054P)
Final semster Thesis student, Seoul National University

From the changes that have been proposed, the new curriculum looks to remove some of the current courses and introduce courses which shall help a student delve more into Computer Science while providing opportunities to take up other electives which will help maintain a broad overview of different topics which BITS students are identified with.

The following discussion would make greater sense if we understand that there are broadly *three categories of students* in a particular discipline in their final year:

- those who like their discipline and will pursue higher studies or a job in related fields
- those who know they are interested in another field (no necessarily academic/technical)
- those who have not identified their interests clearly **(Majority)**

(TS = Final Year First Degree Thesis, PS = Practise School 2)

The suggestion to restrict 5 month long Practise School 2 or a semester long Thesis to the student's discipline merits a discussion.

- **Advantages for A7 students**
  1. **Experience.** The degree awarded is B.E(Hons.) Computer Science and along with studying courses, it is imperative for a student to **work on a long-term project** to have a well-rounded understanding and experience by the time he graduates.
  2. **Set PS process right.** There is a consistent trend amongst students to take up PS2 stations based on factors such as peer group, remuneration and location without caring much for the actual work which is offered. It is an immature way to decide and students do not apply their CDC knowledge by working on actual large-scale projects. Restricting PS will help students stay focused and learn more about use of CS in industry.
  3. **Misuse of TS by students.** The number of people taking up TS is very small as compared to PS, esp. in 4 year programs. TS option is generally used by students to stay on campus and have a relaxed time with little to no pressure since most faculty members react as per the motivation of the candidate owing to time constraints and other factors. Most students submit a thesis of questionable depth and a large part is copy-pasted from the handful of papers or previous reports seen. This trend is consistent irrespective of the actual discipline students do their TS in. By having students do TS in their own discipline, it would be better possible to monitor candidates who just intend to while away time by taking up thesis in a generic topic just to stay on campus and/or avoid their own discipline work.
  4. **Misuse by faculty.** Also, it has been observed that some faculty member in other disciplines get these Thesis students to perform tasks such as helping write a book etc. as a form of an informal deal for supervising thesis and provide recommendations. This can be looked at better by the Group Leader if the faculty supervising thesis are from his discipline.
  5. **Genuineness.** Stopping a student from doing TS in a diff. area does not stop him from taking it up after he graduates if he feels genuinely motivated.
  6. **Difficult to evaluate.** Since thesis grading would be introduced and the plan is to perform grading locally amongst students of the same discipline, evaluating an outlier thesis would be difficult and hence, it makes sense to have thesis in ones own discipline.

- **Disadvantages of restricting PS-TS**
  1. **Interest mismatch.** Many students realize that they are not interested in their discipline which they chose when they were admitted to BITS. After 6 semesters of different courses and exposure to various areas, they are in a much better position to understand what would be a better choice for them keeping in view the long career that lies ahead.
  2. **Similar PS work.** Most PS stations have very similar work for an A7 student: understand the framework the company uses and perform small tasks therein. So, it isnt realistic to believe that an A7 student has a lot of options in terms of PS stations, esp. if he wants to work on an unknown topic even in his discipline.
  3. **Risk-taking ability as a student**. Even though a student can take up things out of his interest after graduating (there are alumni working in movies, photography), the opportunities and risk-taking ability decreases since youre expected to be a professional now and not a student. Allowing a student to do a thesis or PS in a field of his interest gives him more time to fail and learn (or succeed).
  4. **Loss-loss for all.** Forcing a student to do a PS in Computer Science/ IT when he knows he does not wish to continue in this field does not help any party. Anecdotes do not prove anything but there is currently a Physics-Computer Science student on campus doing his thesis in literature out of genuine interest and passion and it took a PS at an IT company for him to be able to make that decision.

- **Other points**
  1. **Social problem.** For the problem of students deciding on PS based on money etc., it is important for students to make efforts to be aware of the nature of work offered in different areas beforehand. Previous year PS reports and project titles are available but few students go through it, and it is unrealistic to assume that the money-oriented mindset of society can be suddenly erased from students.
  2. **Judging quality of thesis.** Thesis evaluation procedure is a welcome change but it needs to be rigorous yet flexible enough so that students opting for TS do not suffer adversely from the trend of indiscriminate distribution of A grades in PS. The focus could be on evaluating the efforts put in by students rather than actual publishable results, since the bar to publish results in different areas varies a lot.
  3. **Relative or Absolute?** Relative grading for thesis makes less sense because different topics have different difficulty levels. It can follow the project model where most faculty members grade a students progress independent of other students. Similarly, the thesis supervisor can grade a student based on his and his work alone. There need to be general guidelines in place to help chart a map. The logistics of thesis defence can be difficult but it could be a strong motivation for a student to work seriously.
  4. **On or off-campus?** A student should be allowed to do thesis on-campus or off-campus, like right now. For an off-campus thesis, it should be the students responsibility to find a faculty member on campus (eligible to supervise first degree thesis) who would co-supervise the students thesis and perform the necessary evaluation.
  5. **Diffirent discipline?** A student should also be allowed to do thesis in another discipline. If he does it off-campus, then he shall have a guide at that location and must have a co-supervisor from Pilani who will need to be updated of all developments.

- **Alternate structure.** The best way to ensure that a dual degree student spends good time with both the MSc as well as the BE degree is by restructuring the curriculum in such a way that the student completes a Thesis on the MSc course right after having done its courses. An example restructuring could be something like this:

1st sem - Common courses

2nd sem - Common courses

3rd sem - MSc CDC1

4th sem - MSc CDC2

5th sem - MSc Electives

6th sem - MSc Thesis

7th sem - BE CDC1

8th sem - BE CDC2

9th sem - BE Electives

10th sem - BE TS/PS

Advantages:

1. This structure aligns well to the the CDC years of 4-year students
2. The electives are done only after basic understanding of the subject.
3. DCOCs and other courses can be structured through 3rd, 4th, 5th, 7th, 8th and 9th sem. This structuring will depend on the courses.