Principles of Curriculum Design and Revision: A Case Study in Implementing Computing Curricula CC2001

M.R.K. Krishna Rao, S. Junaidu, T. Maghrabi, M. Shafique, M. Ahmed, K. Faisal
Information and Computer Science Department
College of Computer Sciences and Engineering
King Fahd University of Petroleum and Minerals
Dhahran 31261, Kingdom of Saudi Arabia

{krishna,sahalu,maghrabi,shafique,mahmed,faisal}@ccse.kfupm.edu.sa

ABSTRACT
Our department has recently revisited its computer science program in the light of IEEE/ACM Computing Curricula 2001 (CC2001) recommendations, taking into consideration the ABET's Criteria for Accrediting Computing programs (CAC 04-05). The effort resulted in a revised curriculum. This paper presents the different decisions we made with regard to the curriculum orientation, knowledge units coverage, transition management, and monitoring and assessment. The paper also sheds some light on challenges faced. Tables provided in the paper show that the curriculum successfully implements CC2001 recommendations while satisfying the CAC 04-05.

Categories and Subject Descriptors
K.3 [**Computers & Education**]: Computer and Information Science Education – Computer Science Education

General Terms
Documentation, Design, Standardization.

Keywords
Curriculum revision, CC2001, Core Technologies.

1. INTRODUCTION

Though KFUPM is the oldest university in the country, it has always strived to keep up-to-date with advances in science and technology as well as emerging trends in education philosophy. It has been very active in starting new programs and revising existing programs. It currently has 15 departments granting BS degree in more than 20 majors. In addition the university grants MS and PhD degrees in most of these majors. Attempts have been made to get accreditation from international boards for these programs. For instance, our computer science program was recently reviewed and certified by ABET/CSAB to be substantially equivalent to any program in North America accredited by them. The university provides modern facilities like smart classrooms and good laboratories giving every opportunity for students to excel. Class section sizes are strictly limited to 30 so that a healthy teacher-student ratio is maintained for its nearly 10,000 students. Its reputation and overall quality of education attracts a good student population. Its admission procedure screens students in two stages and only the top 4% students passing the second examination are admitted.

A couple of years ago, we subjected our old program to an assessment by a CSAB visitation team. In response to their feedback, we embarked upon revising our curriculum to closely follow CC2001 recommendations, meet our program objectives and satisfy ABET's Criteria for Accrediting Computing programs for the 04-05 cycle (CAC 04-05).

Designing a new program or revising an existing program is a demanding exercise that requires skill, care and patience. Any team embarking on this exercise needs to address the following questions:

1. what general principles of curriculum design should be closely followed and which of them given higher priority,
2. what general skills should be imparted to the students to produce graduates that are ready for productive life after leaving university and to be lifelong learners,
3. what particular skills and concepts, specific to the discipline, should be taught and reinforced with good project work, and
4. how best to achieve the above goals, e.g., through activities-based or objectives-based curriculum design.

Answers to these questions determine the direction taken by the team and the objectives of the program.

We identified (1) progression, (2) balance, (3) flexibility, (4) coherence, (5) integrity and (6) currency as the important principles guiding our curriculum design exercise and decided to accord the highest priority to flexibility.

The general skills identified as important to be imparted to the students are (a) communication skills, (b) critical thinking skills, (c) mathematical rigor, (d) scientific temper and (e) ability to work in teams. These skills are highly valued by employers in their recruitments (see e.g., [6]). The specific computer science concepts and skills identified as central to our curriculum are program construction, analysis and testing, data structures, algorithms, programming paradigms, operating systems,

ITiCSE'05, June 27–29, 2005, Monte de Caparica, Portugal.

databases, data mining, discrete structures, theory of computation, logic, networks, security, distributed computing, graphics, multimedia, modeling, human computer interaction and software engineering. These concepts cover most of the items listed as core technologies in [5] and the core knowledge units identified in [1]. Our decisions on general and CS skills are in accordance with the objectives of our program that our graduates will

1. be able to understand, analyze, design, and implement high quality software solutions to real-life problems,
2. be able to function effectively as team members,
3. be able to neatly present their developed solutions in written as well as verbal forms,
4. be aware of their professional and ethical responsibilities, and
5. be able to pursue independent continuous learning.

As for the fourth question, how best can we achieve the above goals, we deliberated on activities-based and objectives-based curriculum design and as explained in a later section, we have opted for the objectives-based curriculum design mainly because most of the textbooks organize their content in topic-based fashion. We also deliberated on the six implementation strategies for introductory courses mentioned in CC2001 and decided to continue with the objects-first strategy adopted by our department even before CC2001 made this classification.

The rest of the paper validates the result of our revision exercise by checking that the decisions made in answer to the questions discussed above are closely followed and the stated goals are met. Section 2 discusses the strategic decisions we made and Section 3 outlines how best our curriculum covers the core knowledge units. Section 4 discusses how we ensure that general skills are imparted adequately, while Section 5 validates the curriculum against the general principles of curriculum design.[1] Section 6 discusses implementation issues and Section 7 compares our program with other CS programs.

2. STRATEGIC DECISIONS

This section discusses the decisions we made and the justifications for them. The main decisions are:

1. continuing with our three-course sequence strategy of covering the introductory material,
2. provision for a rich coverage of theory,
3. adoption of objects-first curriculum design,
4. promotion of AI course from elective to core course,
5. making Database, Networking and Software Engineering courses prerequisite to the Summer Training or Coop programs, and
6. establishing four concentration areas.

We have been using a three-course sequence of introductory computer science courses for the past six years. This sequence is aimed at introducing students to the fundamentals of computer science and developing the skills necessary for applying them in higher-level courses. Implementation of this introductory sequence before the current review was based on an amalgam of the breadth-first and the object-first approach. The new design is based the objects-first model emphasizing object-oriented design and programming from the very beginning, in line with the world wide shift towards object-oriented software engineering.

With an aim of providing our students with a good mathematical foundation, we decided to have two discrete structures courses, one algorithms and complexity analysis course besides an elective course on theory of computation. Our discrete structures courses include material on abstract algebra, temporal logics and an introduction to Church-Turing thesis besides the core topics suggested in [1].

With regards to curriculum orientation, we deliberated on activities-based and objectives-based curriculum design. We opted for an objectives-based curriculum mainly because most textbooks organize their content in topics-based fashion.

We added a core course on Artificial Intelligence, in response to the observation by a CSAB visitation team. We also require our students to take the Databases, Networking and Software Engineering courses before going for summer training or coop programs. This is in response to students' and employers' input towards maximizing the gain in team-working skills and practical experience from these programs.

Aiming to give our students ample opportunity to pursue their topics of interest in a focused manner and greater detail, we structured our advanced and elective courses into 4 concentration areas taking the faculty expertise and the needs of our immediate community into account. They are: Information Management, Intelligent Systems, Net-centric Computing and Systems.

3. CORE KNOWLEDGE AREAS

In deciding the extent of coverage of various CS concepts, we used the CC2001 report as a guide and adequately covered the 13 suggested knowledge areas. We describe this coverage in the next few paragraphs with emphasis on the coverage in the introductory courses sequence and the theory-based courses sequence.

Table 1. Introductory Sequence Coverage

	Core & elective units covered			
Knowledge area	ICS102	ICS201	ICS202	Total
PF. Programming Fundamentals (38)	16+7	11+4	12+3	53
AL. Algorithm & Complexity (31)		3+1	20+6	30
PL. Programming Languages (21)	11+5	15+5	2+0	38
SE. Software Engineering (31)		7+2	2+2	13

Table 1 shows the coverage in the introductory sequence: ICS102 (Introduction to Computing I), ICS201 (Introduction to Computing II) and ICS202 (Data Structures). These courses roughly correspond, respectively, to CS101o, CS102o and CS103o of the CC2001 report. The numbers in brackets (under the "Knowledge Area" column) indicate the minimum coverage

[1] It may be noted that the organization of our paper follows the V-model of software development and validation described in Sommerville [11, figure 19.3].

hours recommended. The numbers in the "Total" column show that the introductory sequence cover all the core knowledge units of Programming Fundamentals, Programming Languages and part of the Algorithms & Complexity and Software Engineering knowledge areas. Some elective knowledge units are also covered in the introductory courses. For example, multithreading is covered in ICS201, elements of physical database design (B-Trees, IM9) and data compression & decompression (NC7) are covered in ICS202.

Each course in the introductory sequence has a laboratory component to complement and provide practical coverage of the lecture material. The laboratory tasks are typically extended with design and programming assignments to carryout tasks beyond classroom coverage. There is also a final lab project that students are required to defend at the end of the semester.

Table 2 shows the coverage of our theory (core) courses. These three courses together cover the Discrete Structures area and the remaining part of the Algorithm & Complexity area that was not covered by our introductory sequence. In addition to covering the core requirements of CC2001, we have over twenty lecture-hours coverage on elective topics like number theory, automata, lattices, abstract algebra and temporal logics. This strong mathematical foundation equips our graduates well to pursue higher studies in many areas of computer science and to solve real-life problems.

Table 2. Mathematical Foundation Coverage

	Core + elective units covered			
Knowledge area	ICS253	ICS254	ICS353	Total
DS. Discrete Structures (43)	45	15		60
PF. Programming Fundamentals (38)			6	6
AL. Algorithm & Complexity (31)		6	38	44

Nine other courses in the curriculum cover the other core knowledge units in a topics-based fashion as highlighted in Section 2. These courses cover knowledge units in the areas of Computer Architecture, Operating Systems, Computer Networks, Programming Languages, Artificial Intelligence, Databases and Software Engineering. Most of these courses have a laboratory component, in line with our emphasis on learning by doing.

An exception to the topics-based coverage is the coverage of core units in Social and Professional Issues knowledge area. Coverage of these issues is spread across core courses in the curriculum (primarily in a humanities course offered by a sister department and secondarily in our software engineering course).

In summary, this section highlights how our curriculum covers the core knowledge areas in CC2001 report. While covering these knowledge units adequately, we give a lot of emphasis on practice and theory. Our curriculum also covers a significant subset of the core technologies identified by Denning in [5].

4. GENERAL SKILLS

This section discusses how we ensure that our graduates have adequate general skills identified in Section 1.

Communication skills: The importance of communication skills in the present era of fierce competition cannot be over-emphasized. It is impossible to find a career that does not require them. In particular due to rapid changes in computer and software technology, it is imperative for a computer science graduate to possess good communication skills. Our curriculum aims to develop

- **writing skills** through courses like software engineering which require production of written documents as deliverables at regular intervals, and a specialized course in Technical Report Writing (where students are required to write about recent developments in CS (based on articles that appeared in journals and magazines in the last two years) for general audience, to be evaluated by English teachers),
- **oral presentation skills** through senior courses having projects as a part of assessment (e.g., database systems, software engineering, senior/coop project) and require formal presentations, and
- **ability to critique oral presentations** by requiring students to evaluate presentations from their classmates (however, it is left to the teacher to decide whether to take student evaluation into account or not) in many of courses requiring oral presentations.

Critical thinking skills: Critical thinking may be defined as the ability to analyze facts, generate and organize ideas, defend opinions, make comparisons, draw inferences, evaluate arguments and solve problems [4]. It inculcates a way of reasoning that demands adequate support for one's beliefs and an unwillingness to be persuaded unless support is forthcoming [12]. Our courses on discrete structures and AI contain a significant component of deductive reasoning and logic teaching how to draw inferences and evaluate arguments. The courses on algorithms and data structures teach analysis and problem solving, and courses like databases and software engineering introduce brainstorming (generate ideas), compare-and-contrast and decision making.

Recently, we came to a conclusion that teaching critical thinking skills needs a concerted effort and initiated a research project to infuse various critical thinking skills into the content of 3 introductory courses (ICS 102, 201 and 202) and reinforce in 3 intermediate and advanced courses (ICS 353, databases and software engineering) using graphic organizers that include questions for clarification, questions about viewpoints and perspectives, questions that probe assumptions, evidence, reasoning, implications and consequences.

Mathematical rigor: As both CC1991 and CC2001 rightly point out, mathematics techniques and formal mathematical reasoning are integral to most areas of computer science, and should be introduced early within a student's course work. We have opted for two courses in discrete mathematics, in addition to the two calculus courses every undergraduate student at our university should take. In fact, we cover topics like abstract algebra and temporal logics, which are not part of the core knowledge units identified in CC2001. We included abstract algebra and lattices to

facilitate study of programming languages semantics, and temporal logics to facilitate study of agent systems (where BDI logics use various modal operators [9]) and concurrency.

Scientific Temper: CC2001 states that the scientific method (data collection, hypothesis formation and testing, experimentation, analysis) represents a basis methodology for much of computer science, and it is vital that students must "do science"—not just "read about science." Our curriculum has twelve courses with a lab component. These courses provide many opportunities to give students a solid exposure to the scientific methodology and develop scientific temper. A course on statistics provides basic competencies in experimentation and data analysis, which are identified as important empirical skills in [3].

Teamwork: It is a widely accepted fact that software development is a team activity and a single individual (even if he is a super programmer) cannot develop any reasonably complex software required by the society in this information era. To give students an appreciation for teamwork and make them aware of coordination/ cooperation problems that may crop up once in a while with teams, our introductory courses give project assignments and senior courses in Database and Software Engineering give projects of reasonable sizes. Senior projects, summer projects and coop work further strengthen teamwork ethics. Taking advantage of the booming software industry around us, most of our students go for coop and summer training in industry.

5. PRINCIPLES OF GOOD CURRICULUM

This section validates our curriculum against the principles identified in Section 1.

Progression: By closely following CC2001 prescriptions about introductory, intermediate and advanced courses, we ensure that the demands on the learner in intellectual challenge, skills, knowledge, conceptualization and learning autonomy increase. Introductory courses employ lower level cognitive skills (knowledge, comprehension, application and analysis) while intermediate and advanced courses require the higher-level cognitive skills (synthesis and evaluation) of the Bloom's taxonomy [2].

Balance: As discussed earlier, our curriculum includes a substantial subset of core technologies identified in [5] covering enough breadth. This together with our concentration areas and courses in physical education and ethics as well as exposure to the world outside the university through coop and summer projects ensure a healthy balance between breadth and depth of the subject material and overall personal development and academic achievement.

Coherence and Integrity: The coherence principle requires that the program has a logical structure and is linked to the program objectives, and the integrity principle requires that the stated objectives are feasible. The coherence and integrity of our curriculum follows from the fact that every deviation from CC2001 recommendations and the old program is made only when a careful analysis justified it.

Currency: Our labs use latest tools like Rational Rose, Oracle, MS Project, and the two main platforms –Linux and Windows– ensuring the currency of our program. Furthermore, we have two 'special topics' courses so that latest trends in the industry can be introduced to students by the interested faculty members. These two courses are regularly offered and are generally well-populated.

Flexibility: We have accorded the highest priority to the flexibility principle and decided to have two options (with coop or summer training) and four concentration areas in Information Management, Intelligent Systems, Net-centric Computing and Systems administration. The program allows a student to choose seven elective courses of his choice giving a lot of flexibility to the student to pursue his/her interests to a greater depth than other topics.

6. IMPLEMENTATION & MONITORING

This section discusses the implementation and monitoring activities of our curriculum revision exercise.

Designing a new undergraduate program or revising an existing one is a challenging task. Getting it approved by the concerned authorities and implementing it is even more challenging. We took more than two years[2] in getting the approval of our colleagues (who are not directly involved in curriculum design) and authorities – confirming that curriculum revision needs patience in addition to skill and care. For the implementation of the revised program, three activities were identified: (1) transition from the current to the revised program, (2) ensuring the achievement of stated objectives and skills, and (3) continuous monitoring of the program for needed changes.

Objective for the transition phase was defined to move maximum students to the revised program in minimal time with no loss or minimal loss of courses taken by a student from the old program. A mapping table between the existing courses and the revised courses was developed. From the mapping table, a set of acceptable moves were developed. For each move a transition rule was defined. These rules were given to all the academic advisors for effective advising and to the office of the registrar for implementation.

For ensuring the achievement of stated objectives and skills, a detailed course description template was developed. The template included the set objectives, learning outcomes, topics and subtopics along with the estimated time for each topic, textbook information, evaluation techniques, grading policy, and lab details for the courses with labs. The departmental curriculum committee analyzed the detailed course descriptions, tabulated the departmental/program objectives with the course objectives, and ensured that the departmental objectives are appropriately supported by the course objectives.

Continuous monitoring of the program is done using various assessment tools. These tools include graduating student survey, faculty survey, alumni survey, and employer survey. Every graduating student is required to complete a questionnaire developed by the curriculum committee. The questionnaire addresses all aspects of the program related to the core technologies covered, general skills, and curriculum design.

[2] We take solace in knowing that we are not alone in this aspect and most curriculum revisions take similar amount of time [7].

Completed questionnaires provide the department with immediate feedback from its graduates. For soliciting input from alumni, employers, and faculty, similar surveys are conducted. These surveys address various aspects of the program and provide comprehensive feedback on the program. Feedback from all these sources is used to identify the shortcomings of the program. Some shortcomings are addressed immediately while others are used as input for the next revision of the program.

The university administration realized the importance of developing its own exit exams in addition to the national and international major field assessment tests. While major field assessment test has the advantage of providing us with a ready-made exam and results that can be compared against national groups, we have no control over the questions and a mismatch between the exam and the curriculum that may creep in because of the rapid changes in computer science. On the other hand, a locally developed exit exam eliminates this problem, but it takes a considerable amount of time to develop and must be constantly revised. Our department has initiated the work on developing an exit exam by preparing lists of MFE (mastery, familiarity, and exposure) topics for each course as suggested in [8].

7. OVERALL PICTURE

This section discusses how the revised curriculum compares with the old curriculum; ABET requirements and curriculums at other reputable universities. We have chosen the following three universities in view of their ranking in top 50 (one close to top 10, another close to 30 and the other close to 50) and the availability of information about their curriculum.

Table 3. Comparison of CS programs

Requirement	**New**	**Old**	**ABET** Criteria	**USC**	**G. Tech**	**Texas A&M**
Mathematics & Science	30	29	30	28	30	33
Humanities & Soc. Sciences	25	23	30	34	28	37
CS core and electives	62	68	≥ 40	50	51	54
Free electives	12	12	-	16	18	12
TOTAL	**129**	**132**	**-**	**128**	**127**	**136**

The above table shows that our revised program is comparable to the programs at the three reputable universities in North America[3] and meet ABET criteria better than the old program. However, we are still falling short of ABET requirements regarding humanities and social sciences. Ours being a technological university, there is a shortage of resources in offering adequate courses in humanities and social sciences. The university is aware of this and is currently working towards a solution to this problem.

[3] It also follows from the fact that the ABET/CSAB team certified our old program to be substantially equivalent to any program in North America accredited by them.

8. CONCLUSION

In this paper, we described the revision of our computer science program in the light of IEEE/ACM Computing Curricula 2001 (CC2001) recommendations, taking into consideration ABET's Criteria for Accrediting Computing programs (CAC 04-05). Rather than narrating curriculum revision with storytelling as suggested in [10], we described our revision using a validation process to show that the result of this exercise satisfied the goals set in answering the four questions stated in section 1 – good curriculum principles, general skills, discipline specific skills and means of revision.

9. ACKNOWLEDGMENTS

The authors would like to thank (a) the King Fahd University of Petroleum and Minerals for supporting this research work, and (b) other colleagues in the curriculum committee for their contributions and whole hearted support in the very demanding task of curriculum revision.

10. REFERENCES

[1] ACM//IEEE. *Computing Curricula 2001*. Electronic version available at http://www.acm.org/sigcse/cc2001/.

[2] Bloom, B.S. (Ed.), *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain.* Longmans, New York, 1956.

[3] Braught, G., Miller, G.S., and Reed, D. Core empirical concepts and skills for computer science, In *Proceedings of the SIGCSE symposium on Computer Science Education (SIGCSE'04).* ACM Press, New York, NY, 2004, 245-249.

[4] Chance, P. *Thinking in the classroom: A survey of programs*. Teachers College, Columbia University, 1986.

[5] Denning, P.D. Great Principles of Computing, *Communications of the ACM*, **46** (Nov 2003), 15-20.

[6] Hagan, D. Employer satisfaction with ICT graduates, In *Proceedings of the Sixth Australasian Computing Education Conference (ACE'04)*, 119-123, 2004.

[7] Lee, J.A.N., Humanization of the computer science curriculum, In *Proceedings of InSITE'02.* Informing Science Press, 2002, 908-914.

[8] McDonald, M. and Mcdonald, G. Computer science curriculum assessment, *In Proceedings of SIGCSE technical symposium on Computer science education* (*SIGCSE'99)*, 194 – 197, 1999.

[9] Rao, A.S. and Georgeff, M.P. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning* (KR'91), 473-484, 1991.

[10] O'Riley, P. A different storytelling of technology education curriculum revisions: a storytelling of difference, *Journal of Technology Education*, **7,** 2 (1996), 28-40.

[11] Sommerville, I. *Software Engineering*, Addison-Wesley, 2001.

[12] Tama, C. Critical thinking has a place in every classroom. *Journal of Reading*, **33** (1989), 64-65.