ghetto" that students take and forget, if ties are not explicit. On the other hand, I expect my experiences mirror those of many other faculty that student eyes often light up when they see mathematics collaborating with computer science in answering questions.

Course Pre-requisites

Pre-requisites for a course typically have two purposes:

A. Identify subject matter from one course that is utilized in the second
B. Assure a general level of sophistication, maturity, or experience

Content background (purpose A) allows a later course to proceed quickly and efficiently, and these pre-requisites may serve well. On the other hand, my earlier observations on making connections suggest that students may not understand pre-requisites unless the later courses explicitly utilize content from previous ones. In some cases -- particularly after a course structure has been in place for several years, it is easy to forget the reasons for pre-requisites, and faculty may forget to connect new ideas with content from previous courses.

Of course, from time to time, maturity pre-requisites (purpose B) may make sense, but I wonder if this purpose is overused -- at least on some campuses.

Pre-requisites can provide structure for a curriculum, but they also decrease flexibility in student scheduling. When material from one course is actually used later, and when ties among topics are explicit, then pre-requisites may strengthen a program substantially. However, when one course does not really build on another, then a pre-requisite may serve little real purpose.

Conclusion

Building on the comments of Tony and Peter, mathematics can provide valuable assistance when it is tied to topics in computing. Further, issues of topic selection, connections among topics, and pre-requisites certainly apply to the selection of mathematics within the computer science curriculum. However, the same questions likely apply much more broadly as well.

Bibliography

[1] Pedagogy Focus Group 2 of the Computing Curricula 2001 Task Force, "Report of Group 2: Supporting Courses", Draft 5.2, July 2, 2000.

Community College Corner

Associate-Degree Transfer Curriculum in Software Engineering

Robert D. Campbell

In 2004, curricular guidelines for undergraduate programs in Software Engineering were published under the title *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.* That report, together with accompanying materials, can be found at http://www.computer.org/education/. That work was the result of a joint task force of the IEEE-CS and the ACM; this task force included Beth Hawthorne, a member of the ACM Two-Year College Education Committee.

While that report spoke to a broad collection of undergraduate programs and curriculum options across the spectrum of software engineering, the ACM Two-Year College Education Committee undertook an initiative to formulate guidelines for software engineering preparation targeted specifically to associate-degree granting institutions. The result of that work, a report titled *Computing Curricula 2005: Guidelines for Associate-Degree Transfer Curriculum in Software Engineering*, shares common goals and outcomes with the above-mentioned undergraduate software engineering curriculum report. Furthermore, the associate-degree software engineering report shares common goals and outcomes with the Computer Science curricular guidelines for associate-degree granting institutions that were finalized and approved in 2003 and published under the title *Computing Curricula 2003: Guidelines for Associate-Degree Curricula in Computer Science.* That report, together with accompanying materials, can be found at http://www.acmtyc.org/.

By basing the *Guidelines for Associate-Degree Transfer Curriculum in Software Engineering* report on recently published sets of international curricula guidelines, the following goals are fulfilled:

- The use of computer science and mathematics courses from the *Computing Curricula 2003: Guidelines for Associate-Degree Curricula in Computer Science* report enables two-year colleges in the United States to incorporate a software engineering track easily into an existing computer science transfer degree program,

irrespective of the specific department offering the degree.
- The incorporation of the software engineering philosophy, concepts, coursework and outcomes from the *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* report helps to properly prepare students and facilitates seamless articulation.
- The report can be used to implement an introductory software engineering curriculum in countries outside the United States whose institutions have missions consistent with the US two-year college model.

Software engineering requires both the analytical and descriptive tools developed in computer science and the rigor that the engineering disciplines bring to the reliability and trustworthiness of the artifacts they design and develop. In particular, the field of software engineering must be viewed as a discipline with stronger ties to computer science than it has to other engineering fields. It must share common characteristics with other engineering disciplines, including quantitative measurement, structured decision making, effective use of tools, and artifact reuse, and must integrate the principles of discrete mathematics and computer science with engineering methodologies. In the United States, as many as one-half of all baccalaureate graduates initiate their studies in associate-degree granting institutions. For this reason, it is important to outline a software engineering curriculum track that can be initiated in the two-year college setting, specifically designed for seamless transfer into an upper-division program. This report recommends a program of study that specifically fulfills this requirement.

The *Computing Curricula 2003: Guidelines for Associate-Degree Curricula in Computer Science* report details a variety of paradigms for introductory computer science curricula, together with computer science and mathematics course descriptions. Two of these paradigms – Imperative-first and Objects-first – are suitable in a software engineering curriculum. The use of computing and mathematics courses, as well as overall curriculum structure, from these *Guidelines* enables two-year colleges to incorporate a software engineering curriculum track easily into an existing computer science transfer degree program. Students who complete this track could reasonably expect to transfer into baccalaureate software engineering programs consistent with the *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*.

Although full program considerations are detailed in the SE report, the major impact is the addition of a course identified as "SE201 Introduction to Software Engineering", with a prerequisite of CS102I or CS102O, and described as follows:

> *This core course introduces the basic principles and concepts of software engineering and provides the necessary foundation for subsequent SE courses at the upper division level. Topics include: basic terminology and concepts of software engineering; system requirements, modeling, and testing; object oriented analysis and design using UML; frameworks and APIs; client-server architecture; user interface technology; and the analysis, design and programming of simple servers and clients.*

Three things stand out in the examination of the *Guidelines*:
- The software engineering track fits very well into the computer science transfer degree program, and the SE201 course can simply take the place of a suggested second-year elective course.
- There is no impact on or addition to a student's initial computer science sequence of study.
- Students interested in the field of software engineering can simply be added to the existing computer science and mathematics courses.

Thus, the implementation of the software engineering track is a very realistic and achievable objective.

Interested parties are encouraged to examine the complete report *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* to become familiar with the full breadth of program considerations as well as the specific details of program implementation. See http://www.acmtyc.org/.

Special note: The ACM Two-Year College Education Committee is seeking participants to engage in an initiative to update the previous Computing and Engineering Technology report. Interested parties are encouraged to contact me at b.campbell@rvc.cc.il.us.

Upsilon Pi Epsilon

Honor Societies, College and Service

Jeffrey Popyack

Greetings! A session I attended at the 2005 annual meeting of the Association of College Honor Societies (ACHS) [1] came at a critical juncture. It has stimulated some serious thought on my part about service learning, the role of service in the college curriculum, and what it might mean for computer science educators in particular.