A Framework and Process for Curricular Integration and Innovation Using Project Based Interdisciplinary Teams

Frank P. Coyle and Mitchell A. Thornton
Computer Science and Engineering Dept
Southern Methodist University
Dallas TX 75275
{coyle, mitch}@engr.smu.edu

Abstract

This paper describes a framework and process for ongoing curricular integration and innovation based on feedback from the performance of undergraduate interdisciplinary teams working on problems that reflect the needs of industry. The five-step process is based on a top-down, objectives-based approach to bringing the computer science and engineering curriculum in line with computing practice. Critical to the effort is the selection of projects for undergraduate teams since project definitions serve both as an opportunity to challenge students and to lay the foundation for departmental direction. Each project is associated with a set of capability requirements that reflect what team members should both know and be able to do to meet project requirements. Upon project completion, expectations and actual performance are used in a curricular feedback loop to identify possible curricular modifications.

We report on an ongoing implementation of this framework at Southern Methodist University (SMU) where integration between computer science and computer engineering serves as a testbed for the framework. At SMU, undergraduate students from computer science and computer engineering are teamed to develop applications that serve to bring core curricula in line with current trends in computing and industry needs.

1. Introduction

Moving forward into the 21st century we find society growing ever more dependent on computing technology. For the disciplines of computer science and computer engineering (CS&E) the task of preparing students for both research and industry is made difficult by the changing nature of the field itself. As the report on *Strategic Directions in Computer Science Education* [1] points out, one of the complicating factors in addressing how best to prepare CS&E students is that the distance between the foundations of computing and its research and application frontiers is considerably shorter in CS&E than in other disciplines. In many ways computing is different in character from other engineering disciplines, due to both the intangible nature of software and the discrete nature of software operations. While disciplines such as Electrical and Mechanical Engineering are based on laws and principles that reflect the nature of the physical world, the relatively recent CS&E disciplines are based on layers of abstraction that continue to build on each other as the field advances.

1.1 Technology Evolution

To illustrate the changing nature of CS&E, Figure 1 illustrates the evolving nature of change in both computer science and computer engineering. In the 1970s, an understanding of lexical theory was important in building compilers that could take advantage of ideas associated with Abstract Data Types. However, the evolution of compiler generators and the emergence of object-oriented languages changed the focus from how to implement languages to how to build systems using object-oriented languages. The late 1980s and 1990s saw the widespread use of languages such as C++ and Java that incorporated and enforced the concept of Abstract Data Type. Then, in the late 1990s, as a result of actually building systems using object-oriented languages, design patterns emerged as a way to manage the complexity in large systems object systems. Today, we see many of these ideas moving higher up the abstraction stack with concepts such as Model Driven Architecture (MDA) providing new abstractions that define new building blocks for system building.

Similarly in the area of computer engineering, the design of computer hardware has moved from schematic capture and laboratory prototyping to the design of circuits using abstractions that reflect Electronic Design Automation (EDA) software tool capabilities. In terms of hardware components, the development and configuration of FPGAs and ASICs is accomplished via the use of Hardware Description Languages (HDLs) such as Verilog or VHDL. Because these tools operate at a higher level of abstraction than the circuit boards of an earlier era, computer engineers spend more time focusing on system-

level issues that include formal specification, hardware-software co-design, and efficient integration of existing specialized circuitry referred to as intellectual property (IP).

The net effect is that today, computer engineering is increasingly driven by higher level system issues. Rather than emphasis on how to design a chip that performs some particular function, computer engineering are now attempting to put an entire system on a chip (SOC). From an educational standpoint, understanding the inner workings of flips flops and gates is not as important as understanding the tradeoffs and issues surrounding SOC design where at least one computational element (a DSP or microcontroller), must coexist with and communicate with peripheral interfaces, pipeline accelerators, embedded firmware and IP blocks. The complexity of such design tasks requires engineers able to work with sophisticated tools and to understand abstraction interfaces and the implications of various kinds of interconnections as the key to successful design and implementation.

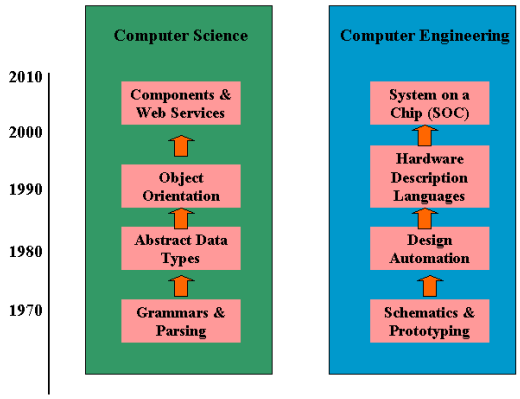

Figure 1. CS&E foundations change as technology evolves

Technology Evolution

This paper describes a framework and process for ongoing CS&E curricular integration and innovation centered on undergraduate interdisciplinary teams. A five-step process describes a top-down, objectives-based approach based on the selection of team projects. Each project is associated with a set of capability requirements that reflect what team members should both know and be able to accomplish in order to meet project requirements. Upon project completion, expectations and actual performance is used in a curricular feedback loop to identify possible curricular modifications.

At Southern Methodist University (SMU) the framework has been implemented by creating teams from both computer science and computer engineering. This serves as a laboratory for exposing undergraduate students to significant application challenges while at the same time gathering the data needed to bring the core curricula in line with current trends in computing and industry needs.

2. The Process

The project-based team approach to maintaining curricular currency is described in five steps. The steps include (i) project selection, (ii) project capabilities definition, (iii) team configuration, (iv) project execution, and (v) curricular feedback.

2.1 Process Elements

Figure 2 illustrates the flow from initial project selection to the utilization of project data for curricular modification.

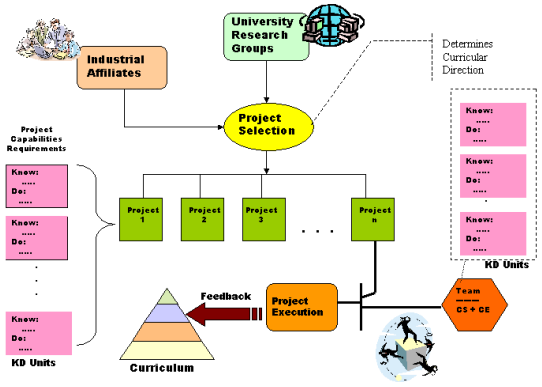

Figure 2. Project-based process for maintaining curricular currency

Project Selection. Project selection is the first and most important step in using team projects to drive curricular innovation. Projects should reflect both departmental strengths and intended technical directions. Project ideas may come from surveying the needs of academic research groups either within or external to the university. Open source projects are an excellent source of project ideas since they are well suited to extension and modification. In addition, local industry can serve as a source of projects since it is often the local industries that hire graduates and often serve as a source of research funding.

Project Capabilities Definition. Defining the skills and capabilities needed by team members to successfully complete their project provides a baseline against which to measure team performance. Capabilities are defined in

terms of Know-Do (KD) units, which specify what factual knowledge students should know and what tasks they should be capable of executing.

Team Configuration. Teams are assembled to carry out the specified projects. When KD descriptions of project requirements are available, students are asked to self-evaluate both their knowledge and skill levels on each of the KD units prior to the start of the project. For departments with well-defined objectives-based courses, these KD units can be added to a project database for subsequent evaluation and input for the feedback.

Project Execution. During project execution, projects are measured with respect to how well initial objectives are met and student performance is tracked via self-evaluation and project manager reports.

Curricular Feedback. At project completion, objectives, capabilities and performance are analyzed with the goal of determining the preparedness of students for project tasks. It is at this stage in the process that student capabilities (or lack thereof) are compared against expected performance. Courses advertised to provide required knowledge and performance skills are evaluated with respect to student performance. Project performance also provides an opportunity to identify required capabilities not provide by the core curriculum. This provides a department an opportunity to reevaluate its curriculum and decide on ways (if any) to bring missing capabilities into the curriculum.

2.2 Curricular Maturity Model

We realize that the full implementation of the curricular feedback process described in section 2.1 takes both time and resources. To provide a measure of the degree to which an institution has adopted the team-based, project-driven approach to curricular innovation, we define an informal Computing Curriculum Maturity Model (CCMM) modeled on the Capability Maturity Model (CMM) widely used in the evaluation of software engineering process maturity. [2]

As illustrated in Figure 3, the CCMM consists of five maturity levels. An institution at Level 0 does not include any team-based project activity in its curriculum. Level 1 means the institution has team based projects but these projects are ad hoc and do not formally related to assessment or evaluation.

Level 2 is where projects selection moves from ad-hoc to some basis in departmental direction. At Level 3, effort is made to define explicit project objectives and capabilities in terms of Know-Do units. Level 4 adds student capability tracking based on student KD units and at Level 5, the curriculum is evaluated based on student performance.

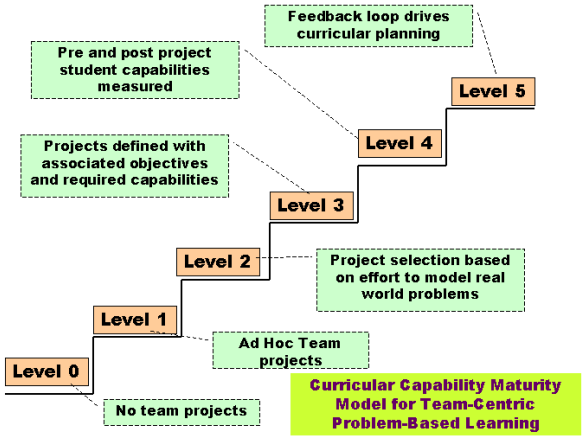

Figure 3. The curricular capability model (CCMM)

3. SMU's Hardware-Software Co-Design Project

The SMU Co-Design Project is a collaborative effort between software engineering and computer engineering faculty with technical, cultural and pedagogical dimensions. Technically, the focus is on the development of model-based architectures for systems with both hardware and software components and tools that support the transformation of high level models into either hardware or software implementations. The capability to generate either software or hardware enables hardware-software allocation based on performance constraints requirements rather than a priori decision before timing bottlenecks have identified.

The Computer Science and Engineering Department at SMU presents a unique opportunity as a testbed for the CCMM framework in that it includes the subspecialties of software engineering and computer engineering within the same department. The department offers both a BS, MS, and PhD in Computer Engineering as well as an MS Software Engineering. Both the computer science and computer engineering undergraduate programs have senior level capstone courses where students work in teams on specific projects. The computer science capstone course is a two-semester course where students work on team based projects. The computer engineering capstone course is also two semesters and focuses on circuit design using high level description languages such as VHDL and Verilog. [3]

3.1 Curricular Integration

At SMU, interdisciplinary projects that bridge the gap between software and computer engineering are defined based on ongoing embedded systems research and industry consultation. Projects are defined that

require some aspect of both hardware and software and teams are composed from both the software and hardware senior level design classes. Currently we define sets of capabilities for each of our projects, which places our efforts at Level 3 on our CCMM scale. In subsequent semesters we plan to move to Level 4 by assessing the pre and post capability level of students participating in the projects.

This integrated approach has the advantage of providing students with an opportunity to address real-world problems while at the same time exposing students to different cultures – i.e. that of computer science vs. computer engineering. Our approach also has the advantage of introducing students to research ideas (currently the use of model-driven architecture for hardware/software co-design). It is our belief that this effort will provide feedback to the department about student preparedness in tackling project level work and create a process to help define departmental direction and specialization. Among its many benefits, the approach offers the opportunity to integrate the education and research environment of the university.

3.2 Accreditation Impact

Adoption of the CCMM model makes provides a built-in approach to satisfying the requirements of ABET accreditation.[4] Among the ABET requirements are (a) an ability to function on multi-disciplinary teams (b) an ability to identify, formulate, and solve engineering problems (c) broadening education to understand the impact of engineering solutions in a global and societal context (d) recognition of the need for, and an ability to engage in life-long learning (e) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

4 Summary

In this paper we describes a framework and process for curricular integration and innovation based on a team-based undergraduate capstone course. The approach is objectives-based and is based on the definition of problems and associated capabilities needed to execute the project. Project definitions serve both as an opportunity to challenge students and to lay the foundation for departmental direction. Upon project completion, expectations and actual performance are used in a curricular feedback loop to identify possible curricular modifications. A Curricular Capability Maturity Model is outlined that will help departments assess their maturity in using team-based projects to fine-tune their curriculum.

A description of an ongoing implementation of this framework at SMU is provided based on senior level capstone courses in software engineering and computer engineering. While the project is in the early stages, we hope to track data and use results to bring our curricula in line with current trends in computing and industry needs.

References

[1] A. B. Tucker, "Strategic Directions in Computer Science Education," *ACM Computing Surveys*, vol. 28, pp. 836 - 845, 1996.

[2] M. C. Paulk, C. V. Weber, B. Curtis, and M. B. Chrissis, *Capability Maturity Model, The: Guidelines for Improving the Software Process*: Addison-Wesley, 1995.

[3] D. J. Smith, *HDL Chip Design. A practical guide for designing, synthesizing and simulating ASICs and FPGAs using VHDL or Verilog*. Madison, AL.: Doone Publications, 1996.

[4] A. E. A. Commission, "Criteria For Accrediting Engineering Programs," 2003.