A COLLABORATIVE BACHELOR'S DEGREE IN SOFTWARE ENGINEERING

Gregory W. Hislop[1], Spiros Mancoridis[2], P. M. Shankar[3]

Abstract - *This paper discusses a new Bachelor of Science in Software Engineering (BSSE) that is offered via a collaboration of three departments of one university. The sponsors span the disciplinary areas of computer science, computer engineering, and information systems.*

The combination of disciplinary areas helps provide a broad foundation for the program. At the same time, while the idea of joint sponsorship may make sense, putting that idea into practice has its difficulties. Academic units have differences in organizational culture, and disciplines may vary in research tradition and curricular expectations. Administrative issues are also more complicated for a collaborative program. All of these factors may make it more difficult to achieve a satisfactory result.

This paper begins with a discussion of the development of the degree program including the process used and issues that had to be addressed along the way. Next, the curricular content of the degree is outlined, with particular attention to contribution and perspective provided by each of the degree sponsors. Finally, the paper discusses initial experiences in offering the degree.

Index Terms - Software engineering education, Undergraduate degree programs.

BACKGROUND

The idea of software engineering is commonly traced to a NATO Conference held in the late 1960s [8]. Degree programs started to emerge in the United States during the 1970s and have continued to spread during the years since. Early programs were at the graduate level, and many of these programs were influenced by a model curriculum developed by the Software Engineering Institute (SEI). [4]

Undergraduate programs developed more rapidly in other countries than in the United States, and that early start is reflected in the distribution of undergraduate degrees today. Canada, Great Britain, and Australia in particular have been early leaders in this area.

The first undergraduate program in the United States did not appear until the 1990s [9], but the number of programs has grown substantially in recent years. An ongoing effort by the Working Group for Software Engineering Education and Training [10] to track degree programs indicates that there are currently at least 68 BSSE programs worldwide including 21 in the United States.

The structure and content of the undergraduate degrees has been influenced by both accreditation considerations and model curricula. In the United States, ABET, through the Engineering Accreditation Commission, has established accreditation of software engineering degrees.

In addition, there has been an ongoing series of efforts to develop model curricula. Projects having impact in this area include guidelines for software engineering education [2], the ACM/IEEE-CS Software Engineering Body of Knowledge (SWEBOK) project [1], and the Computing Curriculum Software Engineering volume currently under development [7].

Development of software engineering at Drexel University has followed a path similar to many institutions in the U.S. A graduate degree was developed during the 1990s [6]. Discussion of an undergraduate degree did not begin in earnest until the late 1990s and the BSSE did not begin operation until 2002. The remaining sections of this paper discuss the development process, structure and content of the degree, and provide comments on initial experience with the program.

The Drexel program is a collaboration among several departments. As interest in software engineering degrees grows, more universities are likely to consider this type of arrangement. Developments in accreditation of software engineering degrees and licensing of software engineering professionals indicate that colleges of engineering will be interested in this area. At the same time, many of the faculty members interested in software engineering are in computer science or other departments. This state of affairs will make collaboration among departments an obvious option at many universities.

DEVELOPMENT PROCESS

Development and implementation of new degree programs is often a difficult task. University faculty members are generally skeptical about new disciplines, and faculty and administrators are cautious about potential implications for resource allocation. Software engineering, particularly in the U.S., has certainly been no exception to these general difficulties. As a new discipline it's validity and content are not universally accepted. In addition, it presents particular problems for universities since it is not clear where it fits within existing university academic units. Development of software engineering at Drexel has reflected these general concerns.

[1] Gregory W. Hislop, Information Science and Technology, Drexel University, Philadelphia, PA, hislop@drexel.edu
[2] Spiros Mancoridis, Computer Science, Drexel University, Philadelphia, PA, smancori@mcs.drexel.edu
[3] P. M. Shankar , Electrical and Computer Engineering, Drexel University, Philadelphia, PA, shankar@cbis.ece.drexel.edu

Internal Considerations

In 1997, Drexel began a Master's degree in Software Engineering as a joint offering of College of Arts and Science, the College of Engineering, and the College of Information Science and Technology [6]. The original proposal for the MSSE dated to 1992, and it had taken about four years to get the program approved. In retrospect, key factors in getting approval had been establishing a small working committee with one faculty representative from each college, development of a common understanding about the degree, and support by the deans of the three colleges.

This experience provided the basis for collaboration in developing the BSSE. The effort began at the suggestion of the Dean of the College of Engineering. After discussing the potential for the project, the Deans of the three colleges agreed to start the development of the BSSE using a collaborative approach similar to the approach for the MSSE. A three person faculty committee again accomplished development work although many additional people including faculty with software engineering interests, and curriculum committee members in the relevant departments and colleges have contributed as well. The development work proceeded much quicker for the BSSE, and the degree was approved about fifteen months after the discussions first started.

External Considerations

The BSSE development process was also guided by a variety of external considerations. These included:

Model curricula – The SEI software engineering education efforts provide several products that are useful for curriculum development. Most directly related is the recently published "Guidelines for Software Engineering Education, Version 1." [2] since it focuses on undergraduate education. In addition to the undergraduate guidelines, SEI also produced an older curriculum model aimed at graduate degree programs. Although the intent is different, the general structure and content of this model provides another useful reference point.

Accreditation criteria – One of the goals for the BSSE is that the degree be accreditable. Therefore the development was guided by the ABET criteria for accrediting SE programs. Due to the non-prescriptive approach to accreditation criteria adopted by ABET, these criteria do not provide any detailed blueprint. However, the criteria did provide useful guidelines for amount of content by broad subject area.

Other BSSE degrees – Several of the existing BSSE programs provided additional points of comparison for the Drexel development effort. The resulting degree benefited from ideas developed by other degree programs. In addition, key faculty members at several of the institutions currently offering a BSSE provided consultation on an informal basis.

Relevant Conferences – A variety of conferences have provided coverage of software engineering education in recent years. In particular, the IEEE/ACM Conference on Software Engineering Education and Training and IEEE/ASEE Frontiers in Education have provided good opportunities for faculty involved in SE education to compare notes.

For a degree program being developed today, several additional considerations would be relevant to the effort. These include:

SWEBOK - The ACM/IEEE-CS Guide to the Software Engineering Body of Knowledge has evolved through several versions and provides an important reference point for degree content. While the SWEBOK covers material generally thought to be beyond the reach of an undergraduate degree, the general framework is still very relevant [1].

CCSE - The ACM/IEEE-CS Computing Curriculum project contains a software engineering volume. At the time of this writing, the volume is still under development, but it's focus on undergraduate curriculum for SE makes its relevance clear [7].

SWENET - This NSF project is still under development, but has as a goal to build an online community of software engineering educators. The focus of this work is creation, collection, and distribution of software engineering course materials with particular emphasis on undergraduate programs [5].

WGSEET - Continuing efforts by the Working Group for Software Engineering Education and Training provide results and opportunities for participation in development of concepts and materials relevant to BSSE degrees [10], [3].

DEGREE CONTENT AND STRUCTURE

The BSSE draws on the strengths of each of the sponsoring departments to support the curriculum. This scope helps creates a degree that is appropriate for students interested in a wide range of application domains.

The degree structure is based on a fairly large core of required courses. This is combined with track options that support application domain interests of the students.

Track Structure

This section presents the overall structure of the BSSE curriculum with comments and explanations of the approach taken.

Software systems are developed to support activity in a wide variety of problem domains. Some examples are:

Embedded systems – Ranging from software to control household appliances to software to control spacecraft, software of this type has the common feature of interacting extensively (e.g., via sensors and controls) with the device hardware.

Information systems – These systems feature data processing facilities such as databases and information

retrieval systems and tend to have extensive facilities for interacting with humans.

Systems software – Computing infrastructure includes software such as operating systems, compilers, database management systems, and transaction processing systems.

These general examples are indicative of the broad range of problem domains addressed by software. Software engineers can be more effective if they have an understanding of the problem domain in which they work. Therefore, it is typical for SE degree programs to include some courses that build knowledge of an application domain. This approach is in line with accreditation guidelines and tends to improve marketability of graduates. For the Drexel degree, each of the participating academic units brings expertise and course offerings in separate application domains.

To support software engineering for a variety of application domains, the Drexel BSSE contains two tracks, as follows:

Engineering track – This track supports students who are interested in developing software for engineering applications and for embedded systems. It can also support students with interests in some types of systems software. Students in this track take the first and second year Drexel engineering curriculum common to all students in the traditional engineering majors. They also take some additional engineering electives.

General track – This track supports students who are interested in developing software for information systems, systems software, and a variety of other problem domains. Students in this track take additional elective courses in information systems and / or computer science, additional courses in liberal studies, and also have some required courses in business.

Degree Requirements

The two tracks closely parallel each other in structure. The tables below summarize the major components of the degree. All credit hours are quarter credits. These tables are followed by comments on the structure and how the two tracks vary.

TABLE I
DISCIPLINARY CORE COURSES

Disciplinary Core	Credits
Software Engineering	32 - 35
Computer Science	9
Information Systems	9
Computer Engineering	9

Disciplinary Core Courses - These courses contain the technical content of the degree. The software engineering courses were created for this degree, while the computer science, information systems, and computer engineering courses represent an effort to make use of existing courses where possible. The course sets are identical for the two tracks except that three credits of introductory software engineering are integrated into the freshman engineering courses for the engineering track students.

TABLE II
GENERAL EDUCATION, DOMAIN, AND ELECTIVE COURSES

Engineering Track	Cr.	General Track	Cr.
Engineering Electives	12	CS/IS Electives	12
Other Engineering	30	Other Engineering	0
Mathematics / Statistics	26	Mathematics / Statistics	26
Basic Science	21	Basic Science	21
Liberal Studies	27	Liberal Studies	36
Business	0	Business	9
Free Electives	12	Free Electives	21

General Education, Domain, and Elective Courses – The two tracks have similar structures with variations to suit the intended application domain for each track. In addition, for many of the mathematics and science courses, there are multiple versions that are acceptable for the BSSE. These typically include a general version of the course and a variant used by the traditional engineering majors. Some additional details of these variations are:

- **Domain electives** – the students in the engineering track take engineering electives (software engineering or traditional engineering disciplines). The students in the general track take computing electives (software engineering, computer science, or information systems) as appropriate. These electives are courses at the 200 level or above.
- **Other Engineering** – the engineering track students take additional engineering foundation courses to provide a background in engineering of physical systems.
- **Mathematics/Statistics** – students take calculus and statistics in either engineering versions or general versions.
- **Basic Science** – students take lab science course sequences. Engineering track students focus on physics and chemistry. General track students have additional options in biology.
- **Liberal Studies** – the general track has an expanded requirement focused primarily on building problem solving skills and communication skills.
- **Business** – the general track requires some course work in basic business concepts.
- **Free electives** – the general track provides for some more free electives to accommodate additional problem domain courses and to make it more reasonable for students to pursue a minor in areas such as computer science, information systems, or bioinformatics.

Software Engineering Courses

This section provides an overview of the SE courses created for the BSSE.

- **Foundations of Software Engineering I** - Teaches students basic programming concepts within a software engineering process that involves specification, documentation, and testing. The process concepts emphasize good internal documentation practices, specifying functional requirements, defect tracking and analysis, and "black-box" testing strategies.
- **Foundations of Software Engineering II** - Introduces students to additional programming concepts. Teaches students how to design, implement, and test object-oriented software applications using simple reusable components. Introduces basic techniques for creating reusable software components.
- **Foundations of Software Engineering III** - Presents intermediate and advanced programming concepts. Introduces students to issues and practices for working with medium-size software systems. Teaches students basic techniques for using application frameworks. Introduces students to software development in teams.
- **Software Specification and Design I** - Study of the principals, practices, and techniques used to gather system requirements and document them in a requirements specification. Includes techniques for requirements discovery such as user interviews and prototyping. Introduces approaches for organizing and expressing software requirements in a requirements specification.
- **Software Specification and Design II** - Continues study of requirements with increasing emphasis on converting requirements into a software system design. Presents alternate approaches to design representation including diagrammatic and formal approaches, techniques for evaluating specifications, specification and design tools, and use of specifications to develop system-level tests.
- **Software Architecture I** - Study of macro-level software system architectures with an emphasis on approaches to interconnection and distribution of system components. Coverage includes common system architectures such as "client-server", "event broadcasting", and "pipe and filter".
- **Software Architecture II** - Continues discussion of software architecture with a focus on micro-level architecture including patterns, frameworks, and component-based software engineering.
- **Software Verification and Validation** - Presents theory and practice of software testing. Emphasizes structural testing including such topics as path testing, dataflow testing, logic based testing, syntax testing, program slicing, mutation testing, fault injection, program perturbation, and testing tools. Discusses techniques for test construction and test suite evaluation, and validation against requirements and design models.
- **Software Evolution** - Covers issues related to change in software systems. Addresses principles and techniques of corrective software maintenance, software enhancements, and software product families. Introduces students to issues of change in large software systems including configuration control, change and product management.
- **Design Project I, II, and III** - An independent project in which student teams design and implement a software system under faculty guidance. Students apply a defined software engineering process for the project including process customization as appropriate.

CHANGES AND EVOLUTION

Although the Drexel BSSE has been operational for less than a year, there are already a variety of issues that will need to be addressed to continue development of the program. These fall into the general categories of administrative issues and curricular issues.

Administration

The BSSE began as the effort of three colleges within the university. In the time since then, the department of Computer Science has been moved to the College of Engineering. Nonetheless, there are still two departments in Engineering plus the College of Information Science and Technology collaborating on the BSSE. This continues to offer a broad base and significant depth in faculty resources to support the program. At the same time, there has always been recognition that this is a difficult model for administering a degree program. This difficulty has been particularly apparent in three areas during the first year of operation:

- **Student Care** - Software engineering students feel somewhat adrift since there is no single focal point for their degree program. Advising is not as carefully developed and delivered as it might be since the advisors for the program have much larger responsibilities advising students in other degree programs.
- **Marketing** - The program is not being marketed as well as it might be. At best, marketing attention is an afterthought by the units that participate. In addition, the collaborative approach to the program makes it difficult to describe in a simple way that prospective students can understand.
- **Curriculum Development** - While there is a mechanism for curriculum development, the requirement to coordinate across three administrative units makes it difficult to act in a timely manner.

Curriculum

Software engineering is a rapidly evolving field. Advances in the discipline require continuous changes to the curriculum. In addition, continued developments in accreditation and model curricula are both likely to require adjustment of existing programs like the Drexel BSSE.

The ABET accreditation criteria were known at the time this degree was proposed. However, the first accreditation visits have only taken place in the current academic year. The results of those early visits are likely to provide new insights into how the accreditation criteria will be interpreted. These insights, in turn, are likely to suggest curricular adjustments.

The evolution of curricular models is likely to have as much or more impact as developments in accreditation. The Computing Curriculum Software Engineering volume is still being drafted at present, but it seems clear that it will be the most visible undergraduate SE curriculum model to date. As such, it will provide an important reference point for all existing and new BSSE programs. Drexel faculty members have been active in the development of this model, and we expect to use it as a key reference point for evolution of the BSSE.

Finally, it is important to note that software engineering seems to benefit from an interdisciplinary approach. Traditional disciplines such as computer science, computer engineering, and information systems all include elements that are useful in building a strong software engineering program. The experience at Drexel highlights some of the difficulties in trying to work across disciplines in creating a degree program. At the same time, this experience also makes it clear that this approach offers some substantial benefits if it can be made to work.

REFERENCES

[1] Abran, Alain, James W. Moore, Pierre Bourque, and Robert Dupuis (eds). *Guide to the Software Engineering Body of Knowledge.* Los Alamitos: IEEE-CS Press. December, 2001. Also online at http://www.swebok.org

[2] Bagert, Donald J., Thomas B. Hilburn, Gregory W. Hislop, Michael J. Lutz, Michael McCracken, Susan A. Mengel. "Guidelines for Software Engineering Education, Version 1." *Technical Report. CMU/SEI-TR-99-032.* Software Engineering Institute. October, 1999.

[3] Duley, Rick, Gregory W. Hislop, Thomas B. Hilburn, Ann E. K. Sobol. "Engineering an Introductory Software Engineering Curriculum." *Proceedings, 16th Conference on Software Engineering Education and Training.* IEEE CS Press. 2003.

[4] Ford, Gary, Norman E. Gibbs. "A Master of Software Engineering Curriculum." IEEE Computer. September, 1989.

[5] Hislop, Gregory W., Thomas B. Hilburn, Michael J. Lutz, Susan Mengel, Mark J. Sebern. "Software Engineering Course Materials." *Proceedings, 16th Conference on Software Engineering Education and Training.* IEEE CS Press. 2003. Also online at http://www.swenet.org

[6] Hislop, Gregory W., Spiros Mancoridis, and P. Mohana Shankar. "Creating a Jointly Sponsored Master of Science in Software Engineering." *Proceedings, Frontiers in Education '99.* Los Alamitos, CA: IEEE CS Press. November, 1999. pp. 7-11

[7] Mengel, Susan A., Ann E. K. Sobel, Rich LaBlanc, Mordechar Ben-Manachem, Timothy C. Lethbridge, et al. "IEEE-CS/ACM Computing Curriculum Software Engineering Volume Project." *Proceedings, 16th Conference on Software Engineering Education and Training.* IEEE CS Press. 2003. Also online at http://sites.computer.org/ccse/

[8] Naur, P. and B Randall, (eds.) *Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee.* NATO. 1969.

[9] Naveda, J. Fernando, and Michael Lutz. "Crafting a Baccalaureate Program in Software Engineering." *Proceedings, 10th Conference on Software Engineering Education and Training.* IEEE CS Press. 1997.

[10] Working Group on Software Engineering Education and Training. Online at http://www.sei.cmu.edu/collaborating/ed/workgroup-ed.html.