

The Current State of Software Engineering Masters Degree Programs

Arthur Pyster, Richard Turner, Devanandham Henry,
Kahina Lasfer, Lawrence Bernstein
Stevens Institute of Technology
{APyster, RTurner, DHenry, KLasfer, LBernste}@Stevens.edu

Kristen Baldwin
U.S. Office of the Under Secretary of Defense (Acquisition, Technology, Logistics)
Kristen.Baldwin@osd.mil

Abstract

A broad coalition of professionals from academia, industry, and government, under sponsorship of the US Department of Defense, is building a new model curriculum for a Masters Degree in Software Engineering (SwE). Before beginning to create such a model, a study of 28 programs was completed to determine the current state of SwE masters-level education in the US and abroad.

1. Introduction

Worldwide, software delivers most of the value in new products. Software is the underlying technology that advances the capabilities of many of contemporary life's tools and toys. Medical devices, automobiles, aircraft, environmental and power generation systems, mobile phones, and entertainment components are all dependent on software-driven functionality. Much of the complexity of those products and systems resides in and is addressed by software. Because of this complexity and the inherent difficulties of software development, most of the "surprises" that occur in system integration and after product shipment and system deployment can be traced back to incorrect software implementation.

The ability of any large company or government agency to manage its projects and organization depends heavily on sophisticated software systems that support its business and technical processes, ranging from logistics systems to manufacturing systems to customer relationship management systems. Yet, reports from the General Accountability Office [1], the Standish Group [2], and others have painted the same story for years – that creating and evolving large-scale software on schedule, on budget, with expected functionality, is uncommon.

Software engineering (SWE) is the acknowledged discipline by which large-scale, trustworthy, and complex software is developed. Many universities teach software engineering at the undergraduate level. More than 30 colleges and universities helped create the model curriculum for undergraduate SWE education that the ACM and IEEE published in 2004 [3]. Many universities offer a masters degree in SWE. Yet, it was back in 1991 when the Software Engineering Institute of Carnegie Mellon (SEI) created a model curriculum for graduate education in SWE [4]. A fresh look at a model graduate curriculum is in order considering the reliance of the world economy on the quality of senior SWE professionals.

The iSSEc (integrated Software and Systems Engineering curriculum) project is creating a model graduate SWE curriculum that reflects new understandings in how to build software, how software engineering depends on systems engineering, and how software engineering education is influenced by individual application domains, such as telecommunications and defense systems. The resulting curriculum will be suitable for a university education leading to a Masters Degree in SWE.

The initial iSSEc activities included the formation of an Early Start Team (EST) in July 2007. The EST is limited to 15 to 20 experts from industry, government, academia, and professional associations. The establishment of a larger Core Team from those same four community segments is planned for 2008.

Before simply gathering a team and plunging into the model curriculum, prudence suggested scouting out the territory. Therefore, the first step in the iSSEc project was to understand the structure and content of currently implemented masters-level programs. Over 50 universities in the United States and many others globally offer a masters-level degree in SWE. Data from 28 programs has been collected, validated with a knowledgeable faculty member, and analyzed to enable a reasonable description of the current state of practice.

2. Methodology

The first step was to identify sources for the investigation. A list of candidate schools and graduate programs was constructed through web searches, author contacts, and recommendations from members of the EST. The range of schools listed included traditional universities, web-based programs, and government-associated schools. Nearly all the programs and schools identified and contacted not only agreed to take part in the survey, but were delighted to participate and supportive of the study's value to their program and the community at large. Unfortunately, that enthusiasm did not always ward off delays in returning validated data. The 28 programs that are included in the study are shown in Table 1.

Table 1. Programs Included in Survey	
Air Force Institute of Technology	Naval Postgraduate School
Brandeis University	Penn State University – Great Valley
California State University – Fullerton	Quebec University (Canada) *
California State University – Sacramento	Rochester Institute of Technology
Carnegie Mellon University	Seattle University
Carnegie Mellon University West	Southern Methodist University
DePaul University	Stevens Institute of Technology
Drexel University	Texas Tech University
Dublin City University (Ireland) *	University of Alabama – Huntsville
Embry-Riddle Aeronautical University	University of Maryland University College
George Mason University	University of Michigan – Dearborn
James Madison University	University of Southern California
Mercer University	University of York (UK) *
Monmouth University	Villanova University
* <i>Non-US Schools</i>	

It was obvious that we needed some taxonomy to add structure to the analysis of competencies covered in the program curricula. Rather than create yet another software

engineering competency model, the team decided to use the SWEBOK [5] as a widely available, collaboratively developed and thoroughly vetted taxonomy. To simplify data collection, only the first 3 levels of the SWEBOK were used, although some data was collected at the 4th level from a few programs. Subjects taught that were not visible at the 3rd level of the SWEBOK were collected as well.

An Excel-based survey instrument was developed to collect and organize the data from the selected programs. The instrument collected data about the program, the courses, and the competencies taught within each course. Table 2 describes the data collected.

Table 2. Data Collected	
Program Information	
<ul style="list-style-type: none"> • University • Degree • Program Focus • Thesis / Credit requirements • Required Courses • Semi-Required Courses (50% or more likelihood student will take these courses) • Elective Courses • Other Related Degrees • Admission Requirements 	
Course Information	
<ul style="list-style-type: none"> • Course Number • Title • Prerequisites • Description • Course Outcomes / Goals 	
Competencies	
<ul style="list-style-type: none"> • (For each course against the SWEBOK to the 3rd level) 	
P::= Primary Focus Of Course	
c::= Covered In Class	
<Blank>::= Not Covered	

The program and course information was collected from public sources – primarily the Web. Using this information, an initial mapping of the course topics to the SWEBOK was performed by the team. Missing information was highlighted and any questions were captured during this initial pass. Once the data was entered, contact was made with a professor at the target institution. The initial data, emailed to and reviewed by that contact or a recommended substitute, was discussed in a telephone conference with members of the team. Missing data was filled in, errors were corrected, and in many cases, the contact made changes directly to the instrument and emailed it back to the team.

While attempts were made to standardize the way in which the data was provided, there were still some differences in the level of detail provided and the interpretation of the instructions by the academic program personnel. When the person filling out the collection instrument either developed or taught a specific course, the subject was covered to a much greater level of granularity. When the person did not have such experiential knowledge, the information tended to be more generic. This led to adjustments in the way the analysis was performed.

To accommodate the differing levels of granularity, as represented by the differing SWEBOK levels of the data, it was decided that data would be analyzed at the 3rd SWEBOK level and reported at the 1st SWEBOK level. This required the team to heuristically aggregate from lower levels to higher levels where the data had been provided at the finer granularity. Table 3 shows an example of how the levels were aggregated when necessary.

Table 3 – Level Aggregation				
	P: PRIMARY FOCUS OF COURSE;c: COVERED IN CLASS	Course 1	Course 2	Course 3
	SOFTWARE COMPETENCIES(SWEBOK - 2004)	Req	Req	Semi-Req
1	SOFTWARE REQUIREMENTS	C	P	
1.1	Software requirements fundamentals			
1.1.1	Definition of software requirement	c		c
1.1.2	Product and process requirements		c	
1.1.3	Functional and non-functional requirements	c	c	
1.1.4	Emergent properties		c	
1.1.5	Quantifiable requirements		c	
1.1.6	System requirements and software requirements	c		
1.2	Requirements process	c		
1.2.1	Process models		c	
1.2.2	Process actors			
1.2.3	Process support and management			
1.2.4	Process quality and improvement			
1.3	Requirements elicitation	c		
1.3.1	Requirements sources		c	
1.3.2	Elicitation techniques		c	
1.4	Requirements analysis	c		c
1.4.1	Requirements classification		c	
1.4.2	Conceptual modeling			
1.4.3	Architectural design and requirements allocation			
1.4.4	Requirements negotiation		c	
1.5	Requirements specification			
1.5.1	System definition document			
1.5.2	System requirements specification		c	
1.5.3	Software requirements specification		c	
1.6	Requirements validation			
1.6.1	Requirements reviews		c	
1.6.2	Prototyping			
1.6.3	Model validation		c	
1.6.4	Acceptance tests			
1.7	Practical considerations			
1.7.1	Iterative nature of requirements process		c	
1.7.2	Change management			
1.7.3	Requirements attributes		c	
1.7.4	Requirements tracing		c	
1.7.5	Measuring requirements			

3. Findings

The findings from the study fall into two general categories: program characteristics and curriculum characteristics.

3.1. Program Characteristics

The spectrum of programs investigated led to a number of findings about how the programs were managed, their faculty resources, size, longevity, and individual personalities. Some of the more interesting findings are:

- SWE is largely viewed as a specialization of Computer Science – much as systems engineering was often viewed as specialization of industrial engineering or operations research years ago. Data shows that only 26% of the programs are in Software Engineering departments, 44% are within Computer Science departments, and the rest are in a myriad of other academic organizations. For example, Stevens Institute of Technology houses the SWE program in the School of Systems and Enterprises.
- Faculty size is generally small with few dedicated SWE professors. Results show that 48% of the programs have 5 or fewer full-time and adjunct SWE faculty members. For programs with many faculty, there is often a heavy reliance on adjuncts for teaching. We believe having such small full-time faculty size makes programs more brittle. A small change in faculty composition could have a large impact on curriculum content. An example of this brittleness was observed in one of the few candidate programs unable to participate in the study – falling enrollment had led to staffing cutbacks and less time for external activities.
- Student enrollments are generally small compared to Computer Science and other engineering disciplines. The data shows 29% of the programs have 25 or fewer students and 71% have 100 or fewer.
- Many programs focus on specific markets, often driven by local businesses; e.g., Monmouth University caters to a local workforce heavily employed by the Department of Defense while Carnegie Mellon University-West caters to a local workforce of entrepreneurs from Silicon Valley. Other focuses observed were acquisition of defense systems, embedded real-time systems, entrepreneurial technology companies, quantitative software engineering, software economics, safety-critical systems, secure software engineering, and highly dependable software systems.
- The admission requirements vary widely. Some will accept anyone with any bachelors degree and a B average while others require a Computer Science degree and at least 2 years of experience. Preparatory or leveling courses are widely provided to support students unable to meet all requirements. Many programs routinely waive academic requirements for students with extensive industrial experience.
- Program outcome goals are quite diverse. Programs are set up to produce graduates according to the perceived needs and desires of the target student population. Some programs focus on developing skilled software development team members. Others focus on the skills and knowledge required to manage complex projects. In some cases, the graduates are prepared to be chief engineers and software executives.
- Programs continue to be started despite the widespread concern over the decline in computer science majors over the last few years. Of the 28 programs in our study, 8 were started since 2001.
- On-line offerings are popular, with many programs reaching students far from their physical campuses and some citing a global reach.

3.2 Curriculum Characteristics

The existing programs' coursework structure and content, and the relationship to standards such as SWEBOK will provide valuable input to the model curriculum

development. Although we collected data on all courses offered by the SWE programs, our initial analysis only looked at courses that were required or semi-required; a semi-required course is one which a student has at least a 50% chance of taking. Some of the more interesting findings are:

- Less than 40% of all programs required an introductory course on software engineering. Several faculty commented that their program was intended for students with work experience in software development. Students who would need an introductory course were either not welcome into their program or were expected to take undergraduate courses to prepare them for the Masters program.
- There is a wide variation in the depth and breadth of SWEBOOK coverage in required and semi-required courses. The most well-covered areas are courses in requirements, design, and management. The least well-covered areas are courses in maintenance, configuration management, quality, and tools and methods.
- It is clear that the SWEBOOK alone does not represent the breadth of many program's required courses. While we expected to see many elective courses in areas covered only in passing within the SWEBOOK, or even topics not included in the SWEBOOK at all, we were somewhat surprised that such courses were required in many programs. Some of the unexpected required courses focused on specific programming languages (such as C++, Java, and C#), software economics, human factors/user interface design, and legal/ethical issues of software development.
- One of the goals of the iSSEc Project is to examine and address the degree to which systems engineering should be integrated into a graduate SWE curriculum. Not surprisingly, few surveyed programs explicitly address systems engineering in their required and semi-required courses.
- "Object-Oriented" is the standard development paradigm. Almost no one teaches structured methods in software engineering except for historical interest. This has interesting consequences because most systems engineering programs still teach structured methods. This difference can lead to "model clashes;" e.g., when trying to integrate a systems architecture developed using structured methods with underlying software architectures developed using object-oriented methods.
- The flexibility of coursework varies widely. For example, one school offered no electives – every course was required. On average, students take 11.6 courses for their degree, 8.3 of which are required or semi-required.
- Capstone practicums and projects are frequently required. While most programs offer a thesis option, students generally preferred the practicum or project.

4. Conclusions and Next Steps

Analysis of the data gathered in this study continues. The initial work completed has produced a reasonable profile of SWE masters programs currently offered. There are, however, many questions that remain. As always, when analyzing the data, the value of information not gathered emerged. Some of the areas identified for further study are:

- Analysis of electives
- Differentiation between different types of schools (private vs. public; large vs. small; specialized vs. general)

- Data collection from additional non-US schools
- Program development questions:
 - Why do you have MSE vs. MS in CS (with SWE spec.)?
 - How did you decide the number of required courses?
 - How did you design your curriculum?
- Program evolution questions:
 - How has your program content changed since inception?
 - How have student demographics changed over time?

The team intends to do a follow-up survey within the next year that will address these questions and will establish a baseline against which the impact of the model curriculum can be measured. We hope that measurement will be accomplished through re-surveys of the community in a 2-year cycle over the next 6 to 10 years.

5. References.

- [1] General Accountability Office. Defense Acquisitions: Assessment of Selected Major Weapons Programs, GAO-06-391, April, 2006.
- [2] Standish Group International. The CHAOS Chronicles, 2003.
- [3] LeBlanc, Rich, et al. Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. ACM and IEEE Computer Society, August 2004.
- [4] Ford, Gary. 1991 SEI Report on Graduate Software Engineering Education, Software Engineering Institute, CMU/SEI-91-TR-002, April 1991.
- [5] Abran, Alain, et al. Guide to the “Software Engineering Body of Knowledge (SWEBOK), IEEE Computer Society, 2004.