AC 2008-1233: A DRAFT REFERENCE CURRICULUM FOR A MASTERS DEGREE IN SOFTWARE ENGINEERING: A JOINT INDUSTRY, ACADEMIC AND GOVERNMENT INITIATIVE

Arthur Pyster, Stevens Institute of Technology

Dr. Pyster is a Distinguished Research Professor at Stevens Institute of Technology, the Stevens Director of the Applied Systems Thinking Institute (ASysT), and a member of the Board of Directors of INCOSE. Previously, he was the Senior Vice President and Director of Systems Engineering and Integration for SAIC, Deputy Chief Information Officer and the Chief Scientist for Software Engineering at the Federal Aviation Administration, Chief Technical Officer at the Software Productivity Consortium, director at Digital Sound Corporation, Manager of Systems Engineering at TRW, and an Assistant Professor of Computer Science at the University of California at Santa Barbara. Dr. Pyster has a Ph.D. in Computer and Information Sciences from Ohio State University.

Richard Turner, Stevens Institute of Technology

Dr. Richard Turner is a Distinguished Service Professor at Stevens Institute and a Visiting Scientist at the Software Engineering Institute. He was most recently a Fellow with the Systems and Software Consortium in Herndon, VA. As a Research Professor at The George Washington University, he taught graduate courses and directly supported the Department of Defense working with a wide range of research organizations and system developers to transition new software-related technology to defense acquisition programs. Dr. Turner served in the Federal Aviation Administration for nearly a decade, and has worked for several engineering firms addressing the needs of defense, intelligence and civil government agencies. Dr. Turner holds a DSc in Engineering Management from the George Washington University.

Devanandham Henry, Stevens Institute of Technology

Devanandham Henry is a PhD student and Research Assistant at the School of Systems and Enterprises, Stevens Institute of Technology. He has a B.Tech degree (1997) in Aeronautical Engineering from the Madras Institute of Technology – Anna University, Chennai, India and an M.Tech degree (2002) in Aerospace Systems Engineering from the Centre for Aerospace Systems Design and Engineering (CASDE), Indian Institute of Technology - Bombay, India. He was with the Aeronautical Development Agency, Bangalore, India for nine years working on Experimental Aerodynamics, Aircraft Performance, Air Intake and Design Optimization for the Air Force and Naval versions of the Indian Light Combat Aircraft before joining Stevens in the Fall of 2006 for a PhD in Systems Engineering. His professional interests include Enterprise Engineering and Systems Engineering education.

Kahina Lasfer, Stevens Institute of Technology

Kahina Lasfer is a Phd student in the School of Systems Engineering at Stevens Institute of Technology. Her research area is based on systems thinking in K-12 education. She graduated from Stevens Institute of Technology with a Masters degree in Computer Engineering, and then she worked with Lucent Technologies as a software developer first in embedded systems and then she held a position as a software designer/architect for CDMA2000 project where she participated in numerous projects developing several features to enhance the existing software system. She is now participating in a project to create a model curriculum in software engineering.

Lawrence Bernstein, Stevens Institute of Technology

Larry Bernstein is the Distinguished Service Professor of Software Engineering at Stevens Institute of Technology, Hoboken, NJ. He wrote "Trustworthy Systems Through Quantitative Software Engineering," with C.M. Yuhas, Wiley, 2005, ISBN 0-471-69691-9. He had a 35-year executive career at Bell Laboratories managing huge software projects deployed worldwide. Mr.

Bernstein is a Fellow of the IEEE and the Association for Computing Machinery for innovative software leadership. He is on the Board of Center for National Software Studies and Director of the NJ Center for Software Engineering and is an active speaker on Trustworthy Software in the IEEE Computer Society DVP program.

Kristen Baldwin, Office of the Under Secretary of Defense (Acquisition, Technology, Logistics)

Kristen Baldwin is the Acting Director, Systems and Software Engineering in the Office of the Deputy Under Secretary of Defense for Acquisition and Technology (DUSD(A&T)). She is the DoD focal point for all policy, practice, and procedural matters relating to systems and software engineering. Ms. Baldwin was named Deputy Director for Software Engineering and System Assurance in February 2007. Prior to OSD, Ms. Baldwin served as a Science and Technology Advisor in the Army's Office of the Deputy Chief of Staff for Operations and Plans, and at the Dismounted Battlespace Battle Lab, Fort Benning, GA. Ms. Baldwin began her career at the US Army's Armament Research, Development, and Engineering Center, Picatinny Arsenal. Ms. Baldwin received a Bachelors degree in Mechanical Engineering from Virginia Tech in 1990 and a Masters in Systems Management from Florida Tech in 1995.

A Draft Reference Curriculum for a Masters Degree in Software Engineering: A Joint Industry, Academic, and Government Initiative

Abstract

Over 50 universities in the United States and many others globally offer a masters degree in software engineering. However, the most current software engineering reference graduate curriculum was developed by the Software Engineering Institute at Carnegie Mellon over 15 years ago. Given how differently today's software is used and developed, a fresh look at graduate programs is needed. A broad coalition of professionals from academia, industry, and government is creating a new reference curriculum. This paper presents the current draft of that curriculum.

The curriculum team conducted an initial study of existing SwE graduate programs that showed broad diversity in goals, content and requirements for admission and graduation. The reference curriculum is strongly influenced by SE2004 and the SWEBOK, but also considers industry desires concerning the skills and competencies they expect to see in a graduate. It is designed to provide a graduate-level core curriculum based on a common body of knowledge and to be flexible enough for individual academic organizations to create the program that best responds to their goals, individual strengths and target student population.

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 U.S. Government 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 reference 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 reference curriculum for graduate education in SwE[4]. A fresh look at a graduate reference 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 iteratively developing a graduate SwE reference curriculum (GSwERC) 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. At this point, at least three curriculum iterations are planned – GSwERC v0.25, GSwERC v0.5, and GSwERC v1.0. The first iteration is complete and the second is being written now. More on the specific content of these releases is explained later in this paper. The resulting curriculum will be suitable for a university education leading to a Masters Degree in SwE.

Engagement

Four types of organizations must engage in creating the reference curriculum in order to ensure its correctness and to maximize its usefulness and impact:

1. The industrial and government workforce who are the customers of the curriculum, establish the demand-side requirements for the curriculum. Those requirements take the form of needed SwE competencies in graduating students; i.e., knowledge they expect to be learnt, skills they expect to be mastered, and behaviors they want to be demonstrated. That workforce will be represented by industrial organizations selected from a wide range of market segments and government organizations that acquire and operate large complex software-intensive systems.

2. Educators respond to the demand-side requirements with their own supply-side offerings, i.e., they must propose courses that effectively teach competencies at the graduate level. Note that the range of competencies may exceed that which can be achieved in a Masters program.

3. Professional societies that encourage best practice in SwE must also encourage the creation, dissemination, and use of a reference curriculum.

4. Government organizations that may fund elements of the curriculum (such as the National Science Foundation) actively observe both reference curriculum creation and implementation. The U.S. Department of Defense Deputy Director of Software Engineering and System Assurance is an active sponsor.

Development Approach

As with all good software engineering projects, the team leadership selected an iterative, evolutionary approach to curriculum development. Stevens Institute of Technology began the project in Spring 2007 with sponsorship from the U.S. Department of Defense. In July 2007, Stevens formed an Early Start Team (EST) of several invited experts from industry, government,

academia, and professional associations. As iSSEc has matured, EST participation has grown. Table 1 lists the current list of EST members and observers.

Table 1 – EST Members and Observers as of February 2008

1. Rick Adcock, *Cranfield University and INCOSE, UK*
2. Edward Alef, *General Motors, USA*
3. Bruce Amato, *Department of Defense, USA*
4. Mark Ardis, *Rochester Institute of Technology, USA*
5. Larry Bernstein, *Stevens Institute of Technology, USA*
6. Barry Boehm, *University of Southern California, USA*
7. Pierre Bourque, *Quebec University and SWEBOK volunteer, Canada*
8. John Bracket, *Boston University, USA*
9. Murray Cantor, *IBM, USA*
10. Lillian Cassel, *Villanova and ACM volunteer, USA*
11. Robert Edson, *ANSER, USA*
12. Dennis Frailey, *Raytheon & Southern Methodist University, USA*
13. Gary Hafen, *Lockheed Martin and NDIA, USA*
14. Thomas Hilburn, *Embry-Riddle Aeronautical University, USA*
15. Greg Hislop, *Drexel University and IEEE volunteer, USA*
16. Philippe Kruchten, *University of British Columbia, Canada*
17. James McDonald, *Monmouth University, USA*
18. Ernest McDuffie, *National Coordination Office for NITRD, USA*
19. Bret Michael, *Naval Postgraduate School, USA*
20. William Milam, *Ford , USA*
21. Fernando Naveda, *RIT and IEEE volunteer, USA*
22. Ken Nidiffer, *SEI, USA*
23. Art Pyster, *Stevens Institute of Technology, USA*
24. Paul Robitaille, *Lockheed Martin and INCOSE, USA**
25. Doug Schmidt, *Vanderbilt, USA*
26. Mary Shaw, *Carnegie Mellon University, USA*
27. Ann E Sobel, *Miami university and IEEE volunteer, USA*
28. Robert Suritis, *IBM, USA*
29. Richard Thayer, *California State University at Sacramento, USA*
30. Richard Turner, *Stevens Institute of Technology, USA*
31. Joseph Urban, *National Science Foundation observer, USA*
32. Ricardo Valerdi, *MIT & INCOSE, USA*
33. Osmo Vikman, *Nokia, Finland*
34. David Weiss, *Avaya, USA*

**EST member until December 2007*

Before the EST first met, Stevens started to conduct a survey of existing programs offering a masters degree or equivalent in software engineering. The thought was that any effort to create a reference curriculum should understand what is currently practiced; e.g., the diversity of the program objectives, how many courses are typically required, and what competencies are taught.

That survey was completed in November 2007[5]. The EST held a workshop in August 2007, established goals, defined the curriculum's scope, broke into four primary teams, and began working on sections of the draft curriculum. Working remotely, the teams created initial versions of their sections primarily authored by the leaders of each team and came together for a second workshop in December 2007 to review their drafts and to refine the schedule for the remaining effort. A third meeting was held in mid-February 2008 among the author team to finalize version 0.25, the first curriculum draft suitable for release to a limited set of international reviewers from a wide range of industry domains and academia . Version 0.25 was released March 1, 2008. Their feedback will be included in Version 0.5, which will be released in the second half of 2008 for an open review by all interested parties. In 2009, we expect to publish Version 1.0 of the curriculum, which will incorporate broad community feedback.

iSSEc Project Goals

The following qualitative project goals have been identified to determine project success. These goals transcend the near-term creation of GSwERC 0.25 and 0.5, and reflect the longer-term objective of creating and deploying GSwERC 1.0:

1. Improve existing graduate programs in software engineering from the viewpoint of universities, students, graduates, software builders, and buyers.
2. Enable the formation of new graduate programs by providing guidelines on curriculum content and advice on how to implement those guidelines
3. Support increased enrollments in graduate software engineering programs by increasing the value of those programs to potential students and employers

Current State of SwE Graduate Curricula Survey Results

Before simply gathering a team and plunging into the reference 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 was collected, validated with a knowledgeable faculty member, and analyzed to enable a reasonable description of the current state of practice.

A list of candidate schools and graduate programs was constructed through web searches, author contacts, and recommendations from members of the EST. A wide-range of schools and graduate programs was sought, including public and private schools, traditional campus-based and novel web-based programs, and schools of various sizes. Some schools have been offering graduate degrees in software engineering for more than 20 years. Two schools had just begun offering their degree in 2007. 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. Table 2 lists the programs surveyed.

Table 2. 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

** Non-US Schools*

It was obvious that some taxonomy was needed to structure the analysis of competencies covered in the program curricula. Rather than create yet another software engineering competency model, the team chose to use the SWEBOK[6] as a widely available, collaboratively developed and thoroughly vetted taxonomy.

The survey found great diversity in nearly all aspects of the programs and provided the EST with a broad range of approaches to consider. Among the many findings most relevant in creating the reference curriculum:

- SwE is largely viewed as a specialization of Computer Science. Data shows that 44% are within Computer Science department and 26% are in Software Engineering departments.
- 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.
- 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 two years of relevant experience.
- There is a wide variation in the depth and breadth of SWEBOK coverage in required and semi-required (those which a student has at least a 50% chance of taking) courses.

- 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.

Developing the Curriculum

In the first meeting of the EST in August 2007, the team heard presentations on the SWEBOK, the Software Engineering Undergraduate Curriculum SE2004, SEIs 1991 Report on Graduate SwE Education, and the INCOSE Systems Engineering Graduate Curriculum Framework[7] . From these presentations and the initial results of the survey of existing graduate programs, the team agreed to an outline for the curriculum document (Shown in Figure 2) and established four teams to develop the main parts of the document – Guidance and Outcomes, Architecture, Body of Knowledge, and Packaging. These teams reported back at the 2nd meeting in December, comments and discussion ensued, and the teams reassembled to modify and extend their work.

	Size estimate 25-50 pp
Section	Target Page count
Executive Summary	1
Introduction	3
(Addressable markets, spectrum of degrees, project rationale)	
Curriculum Guidance	3
Student capabilities	
- Masters program entrance requirements (expected knowledge and skills)	2
- Masters graduate Capabilities (expected knowledge and skills)	5
Curriculum	
- Requirements (for the curriculum, "-ilities")	1
- Body of Knowledge (Deltas + and - to SWEBOK)	10
- Curriculum Architecture (to meet requirements)	3
- Course Packaging guidance (order, content, texts, readings, radical packaging) (includes examples of alternative approaches, such as integrative vs. discrete)	10
Discussion of Teaching Methods (philosophical?)	2
Curriculum implementation guidance	
Appendices	(as necessary)
- Rationale (on critical decisions)	
- Program support (faculty, infrastructure, scope – evolution & growth)	
- Mappings	
References/Bibliography	

Figure 2 – Curriculum Document Outline, August 2007

During the meetings, and with input from the development area teams, the EST converged on the following scope of the curriculum for Version 0.25:

- Duration of the curriculum would be the equivalent of ten 3-credit semester courses plus a required capstone experience (e.g. project, thesis)
- The core material that every student would be expected to learn would be limited to no more than could be taught in the equivalent of four or five 3-credit semester courses.

Guiding Principles, Assumptions, and Context

This section articulates the foundational guidance for developing the GSwERC materials: the guiding principles, assumptions, and context for the entire GSwERC effort. Version 0.25 has 17 guidance statements plus elaboration. The statements without elaboration are:

1. The principle purpose of GSwERC will be to provide a framework for development and improvement of curricula that provide software engineering education at the masters degree level.
2. The masters degree described by GSwERC will be a professional degree targeting practicing software engineers. Nevertheless, with slight modification, GSwERC will serve as the foundation for those with a research interest who ultimately seek a doctoral degree.
3. A masters program that satisfies GSwERC should require about as many credit hours as typical programs do now.
4. Software engineering is a field with sufficient knowledge, practice, and theory that it stands separately from computer science.
5. Software Engineering draws its foundations from a wide variety of disciplines.
6. All software engineering students must learn to integrate theory and practice.
7. The rapid evolution and the professional nature of software engineering require an ongoing review of the corresponding curriculum.
8. Development of GSwERC will be sensitive to changes in technologies, practices, and applications, new developments in pedagogy, and the importance of lifelong learning.
9. GSwERC will go beyond knowledge elements to offer significant guidance in terms of individual curriculum components.
10. GSwERC will support the identification of the fundamental skills and knowledge that all graduates of a masters degree program in software engineering must possess.
11. GSwERC will be based on an appropriate definition of software engineering knowledge and a flexible architecture.
12. GSwERC will be international in scope.
13. The development of GSwERC will be broadly based.
14. GSwERC will include exposure to aspects of professional practice as an integral component of the graduate curriculum.
15. GSwERC will include discussions of strategies and tactics for implementation, along with high-level recommendations.

16. The distinction between SE2004 and GSwERC will be clear and apparent.
17. GSwERC will identify prerequisite requirements for students to enter a masters program in software engineering.

Expectations at Entry

Among the most challenging decisions is deciding what students should be capable of when they enter the masters program. After considerable discussion, the EST agreed that students entering a masters program should have:

1. The equivalent of an undergraduate degree in computing or an undergraduate degree in an engineering or scientific field and a minor in computing. The GSwERC Body of Knowledge more completely defines the expected prerequisite knowledge, and
2. The equivalent of an introductory course in software engineering, and
3. At least one year of practical experience in some aspect of software engineering or software development.

The rationale for these expectations is:

1. ***Degree.*** Many existing masters programs in software engineering expect students to have a bachelors degree in an engineering or scientific field, but not a degree in computing. Such students generally bring much of the math skills and the ability to think analytically, both of which are essential to software engineering. Students often have programming experience, although it is usually programming in the small without the benefit of understanding how to address issues associated with large or complex software. However, software engineering depends heavily on computer science, even though software engineering has grown to become a separate discipline.

   In order to engineer software, a student must have mastered the fundamentals of computing, including computer hardware, operating systems, data structures, algorithms, and discrete math. Students who do not have at least a minor in computing will generally lack that mastery.

   Universities frequently offer leveling courses to students who enter a masters program lacking the expected background in computing. In order to make the expectations more concrete, GSwERC identifies specific computing knowledge areas such as *operating systems* with specific Bloom taxonomy levels (in this case *comprehension*) that students should know when enrolling in a masters program.

2. ***Software Engineering.*** The majority of masters programs in the recent survey do not start students with an introductory course in software engineering. These programs assume that the student has picked up the equivalent knowledge either from earlier coursework or from professional experience. GSwERC follows the practice of the majority of programs in that regard. The GSwERC identifies specific software engineering knowledge areas such as *software requirements* with specific Bloom taxonomy levels (in this case *comprehension*) that students should be expected to know when enrolling in a masters program.

Universities frequently offer an undergraduate course that introduces software engineering to students who do not have the equivalent knowledge from a prior course or professional experience.

3. ***Experience.*** Software engineering is a practical field and it is a truism that there is no substitute for experience. The richness of the discussions in a graduate class and the sophistication of the analysis that a student can perform are driven, in part, by the maturity of the students. Students with at least one year of practical experience in some aspects of software engineering or software development have a significantly deeper appreciation for the issues that are examined in the masters program.

   Universities could offer internships to students lacking the expected experience, or otherwise involve them in a significant practical experience early in their masters program.

Expectations at Graduation

Analogous to the statements about expectations at program entry, there are *outcomes* or statements about what a student should be capable of at graduation. Currently, there are nine outcomes plus elaboration. The outcomes without elaboration follow.

Graduates of a masters program in software engineering will:

1. Show mastery of the software engineering knowledge and skills, and professional issues necessary to practice as a software engineer in a variety of application domains with demonstrated performance in at least one application domain.
2. Understand the relationship between software engineering and systems engineering and be able to apply systems engineering principles and practices in the engineering of software.
3. Show mastery of software engineering in at least one specialty such as embedded devices, safety critical systems, highly distributed systems, software engineering economics, or one of the knowledge areas of the GSwERC Body of Knowledge.
4. Work effectively as part of a team, including teams that may be international and geographically distributed, to develop quality software artifacts, and to lead in one area of project development, such as project management, requirements analysis, architecture, construction, or quality assurance.
5. Reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.
6. Design appropriate software engineering solutions that address ethical, social, legal, and economic concerns.
7. Understand and appreciate the importance of feasibility analysis, negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.

8. Learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.
9. Analyze a current significant software technology, be able to articulate its strengths and weaknesses, and be able to specify and promote improvements or extensions to that technology.

Curriculum Architecture

The curriculum architecture is similar to the one included in the 1991 SEI report and is compatible with the existing masters programs that were surveyed. It provides a structural basis for programs to package courses that will teach the body of knowledge and will enable students who meet the entrance expectations to ultimately satisfy the graduation expectations. Figure 3 provides an overview of the architecture.

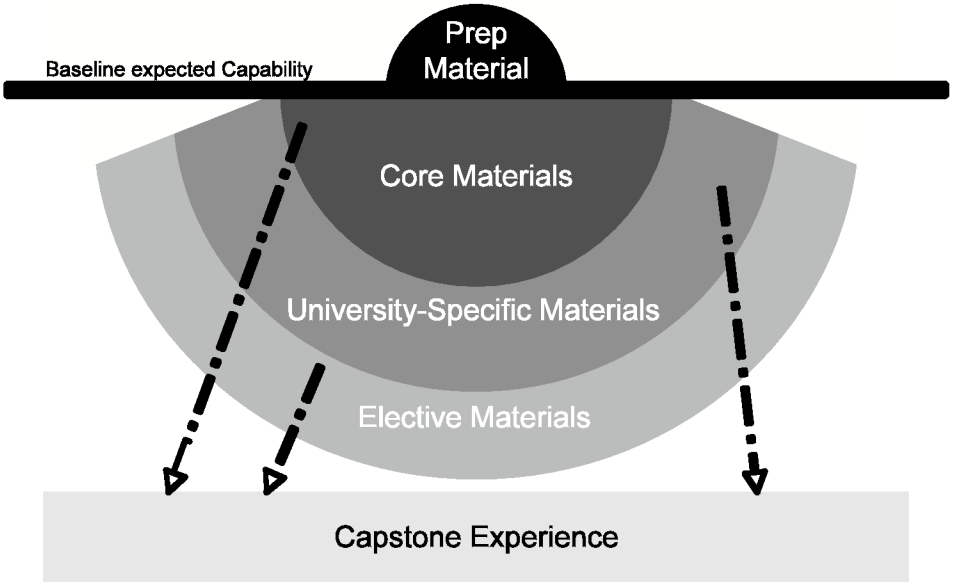

Figure 3 – Architecture Overview

The preparatory material is described in the expectations at entry to the masters program. Students who need this material should expect to take longer to complete their masters degree. Core material is described in the section on the body of knowledge. University-specific courses allow institutions to tailor the degree program to their particular focus (such as defense systems or safety-critical systems) or to faculty strengths. These capabilities could include domain-specific study, deeper experiences in key software engineering technology or research areas, or any other experiences desired by the university for all of its graduates. These will vary by institution because of student demographics, teaching/research/professional focus, delivery mechanisms, external constituents, infrastructure, and accreditation issues. Elective material will allow students to pursue their own interests as well as provide for faculty to pursue their own emphasis areas.

Body of Knowledge

The most difficult task in the entire curriculum effort is creating the Body of Knowledge (BOK) – deciding what is the core knowledge needed for a software engineer at the masters level. If the core knowledge is too large, universities will not have the flexibility needed to tailor their programs. If the core knowledge is too small, the current fragmentation of graduate education will continue, and the reference curriculum will provide little value.

The primary source for the BOK was the SWEBOK[6]. Additional knowledge elements were added from the IEEE/ACM undergraduate reference curriculum[3] and from the INCOSE BOK[8]. In the study and analysis of these sources, various changes were made to satisfy the GSwERC expected student outcomes and to accommodate the needs and views of industry, academia, and the computing professional societies. The current draft of the GSwERC BOK specifies 50 knowledge units together with a Bloom taxonomy level that all students are expected to meet; e.g., all students should have *application* level mastery of *software quality fundamentals* and *software design notations* by the time they graduate.

Much of the discussion about the contents of the core has centered on whether a masters in software engineering is a first or second professional degree. In many engineering fields, a student seeking a masters degree also has a bachelors degree in the same field. Given the desired outcomes for the masters curriculum in software engineering, GSwERC seems to fall somewhere between the two. Unlike many other engineering fields, most software engineering masters students do not have a baccalaureate degree in software engineering, although, with the expanding number of accredited BSSE programs, this may change in the future. Nevertheless, a strong background in computing is expected for students entering the masters program.

Comparison with Actual Programs

Ultimately, GSwERC will include detailed recommendations on how to package a curriculum that is consistent with the BOK and architecture in a way that satisfies the expected outcomes. At this point, however, we instead chose to compare four actual masters programs with the reference curriculum guidelines. We thought this would provide a good picture of the distance between "reality" and the GSwERC guidelines. If four actual masters programs are all highly inconsistent with the GSwERC recommendations, we may have set the bar too high or in a way that is too far removed from current practice to be implementable. On the other hand, if one or more of the four programs completely satisfies all the GSwERC recommendations, the proposed curriculum could be redundant. Professors on the EST from Embry Riddle University, Monmouth University, Naval Postgraduate School, and Southern Methodist University all prepared these analyses. All four programs align reasonably well, but not completely, with the GSwERC recommendations. Figure 4 shows how the programs assessed their overall compliance with the 9 outcomes on a scale of 1 to 5, where 1 means *not addressed at all* and 5 means *fully addressed*.

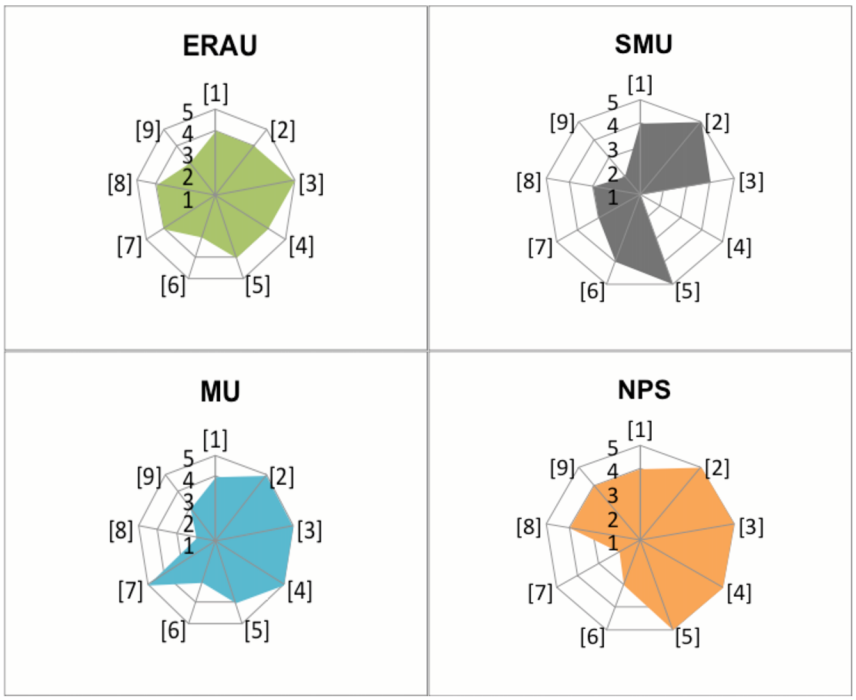

Figure 4 – Comparing Four Programs with GSwERC Outcomes

Conclusions

Developing a new SwE graduate curriculum is a daunting task, made more difficult by the rapid changes in the discipline of software engineering and the evolving relationship between software engineering and systems engineering. The iSSEc project has had the benefit of a broad, dedicated, and extremely capable Early Start Team and solid foundation documents on which to base their effort.

The version number *0.25* reflects the relative (im)maturity of the curriculum at this point. We believe the guiding principles, expectations when students enter the program, and expected outcomes when students graduate are largely correct, although, of course, review by others will test that belief. The curriculum architecture is quite flexible and seems to support the variety of graduate programs that were studied at the beginning of this effort. The GSwERC BOK, on which the curriculum is based, is grounded in the SWEBOK with a small number of additions and changes. Comparison with existing program seems to indicate an appropriate scope and depth in the BOK and outcomes.

While certainly in need of broad validation and refinement, this initial draft should serve as a foundation for the wider community to mature. If you are interested in participating in the further development of the curriculum, please contact one of the authors.

Bibliography

1. U.S. Government 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. Pyster, Arthur, et al. The Current State of Software Engineering Masters Degree Programs, Proceedings of the 21st IEEE-CS Conference of Software Engineering Education and Training, April, 2008.
6. Abran, Alain, et al. Guide to the "Software Engineering Body of Knowledge (SWEBOK), IEEE Computer Society, 2004.
7. R. Jain and D. Verma. A Report on Curriculum Content for a Graduate Program in Systems Engineering: A Proposed Framework, 2007.
8. INCOSE, *Guide to Systems Engineering Body of Knowledge*, International Council on Systems Engineering.