iSSEc

Integrated Software and Systems Engineering curriculum

Organizing Workshop of the Early Start Team

August 15th & 16th, 2007 Arlington, Virginia, USA

WORKSHOP REPORT

STEVENS Institute of Technology

ASysT INSTITUTE

Contents

1. iSSEc Project

Worldwide, most of the value in new products and systems is delivered through software. Much of the complexity of those products and systems resides in and is addressed by software. Most of the "surprises" that occur after product shipment and system deployment can be traced back to software being implemented incorrectly. Software is the underlying technology to advance mobile phones, automobiles, and aircraft. The ability of any large company or government agency to manage its projects and organization depends heavily on sophisticated software that supports its business and technical processes, ranging from logistics systems to manufacturing systems to customer relationship management systems. Software is everywhere. Yet reports from the Software Engineering Institute, the Standish Group, 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 and complex software is developed. Many universities teach software engineering at the undergraduate level. More than 30 colleges and universities helped create the 2004 model curriculum for undergraduate SwE education that was published by the IEEE and ACM.

Many universities offer a Masters degree in SwE. Yet the only existent model curriculum for graduate SwE was created in 1991 by the Software Engineering Institute. Since then the World Wide Web has vastly changed how the world communicates, and software being developed today has changed considerably as a result. Considering the reliance of the world economy on the quality of senior SwE professionals, the lack of a current model graduate curriculum is dismaying.

The iSSEc (integrated Software and Systems Engineering curriculum) Project is creating a new 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 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 project plan for iSSEc included the formation of an Early Start Team (EST) by July 2007. The EST would be limited to 15 to 20 experts from industry, government, academia, and professional associations. The plan also called for establishing a larger Core Team several months later from those same four community segments.

2. Organizing Workshop

The EST assembled for the first time in a workshop that was held on Aug 15-16, 2007 at the ANSER facility in Arlington, Virginia. The top workshop objective was to bring the members of the EST physically together in order to kick-start iSSEc and to coalesce the team around developing a strawman curriculum over the next several months. Detailed workshop objectives are included in Appendix A. When complete, the strawman curriculum will serve as an intermediate project deliverable, and also will provide direction when the larger Core Team's effort begins.

EST members were selected to ensure representation from industry, government, academics, and professional societies. EST participants include people who were associated with earlier efforts, such as the Software Engineering Body of Knowledge (SWEBOK) and SE2004, which is the model software engineering undergraduate curriculum.

As on Aug 15th, 2007 the Early Start Team consisted of:

1. Bruce Amato, *Department of Defense*
2. Mark Ardis, *Rochester Institute of Technology*
3. Barry Boehm, *University of Southern California*
4. Murray Cantor, *IBM*
5. Gary Hafen, *Lockheed Martin*
6. Thomas Hilburn, *Embry-Riddle Aeronautical University*
7. James McDonald, *Monmouth University*
8. Ernest McDuffie, *National Coordination Office for NITRD*
9. Bret Michael, *Naval Postgraduate School*
10. Paul Robitaille, *INCOSE, Lockheed Martin*
11. Doug Schmidt, *Vanderbilt*
12. Mary Shaw, *Carnegie Mellon University*
13. Richard Thayer, *California State University at Sacramento*
14. Joseph Urban, *National Science Foundation*
15. Osmo Vikman, *Nokia, Finland*
16. David Weiss, *Avaya*
17. Robert Edson, *ANSER*

18. Art Pyster, *Stevens Institute of Technology*
19. Larry Bernstein, *Stevens Institute of Technology*
20. Richard Turner, *Stevens Institute of Technology*
21. Sarah Sheard, *Stevens Institute of Technology*
22. Devanandham Henry, *Stevens Institute of Technology*

Of the above, Murray Cantor, James McDonald, Doug Schmidt and Richard Thayer could not attend the organizing workshop, and Mary Shaw joined the workshop on its second day.

3. Workshop Proceedings

The two-day organizing workshop was held at the ASysT Institute, a collaborative endeavor of Stevens Institute of Technology and Analytic Services. ASysT is located in the Arlington, Virginia, USA headquarters of Analytic Services. The workshop agenda is included as Appendix B.

3.1 Welcome & Introduction

The workshop began with a welcome by Dr. Art Pyster, principal investigator of the iSSEc project, Distinguished Research Professor at Stevens Institute of Technology, and the Stevens Director of the ASysT institute.

In her welcome note, Kristen Baldwin, OSD and principal sponsor of iSSEc, brought out the motivation for OSD to support this effort. Software engineering is critical to ensure success in today's world of complex systems. However, a March 2007 MITRE study conducted for DoD clearly brought out inconsistencies in software engineering education and in the knowledge and skills that a typical software engineer would possess. The DoD is increasing its emphasis and attention on software engineering issues. Sizing and complexity of software are identified as critical issues.

Gary Hafen commented that software engineering is "systems engineering of a software product/service." This summarizes aptly the long term vision of iSSEc – to produce a curriculum for an integrated curriculum in software and systems engineering.

Art Pyster presented the workshop objectives, roadmap and roles/timelines of the iSSEc project for the first phase activities, which will produce a reference curriculum suitable for a Masters Degree in Software Engineering (attached as Appendix C-1). Although important, certificate programs and doctoral programs in software engineering were deemed out of scope for iSSEc at this time.

3.2 Summary Presentations

Brief presentations on earlier related efforts were made primarily by participants who were involved in those efforts.

"SWEBOK", Richard Thayer

Art Pyster presented the slides prepared by Richard Thayer, who could not attend the workshop. (presentation attached as Appendix C-2.) The current version of SWEBOK was released in 2004, and the next version is expected in 2009. Art Pyster mentioned that contact with Jim Moore is being established.

SWEBOK 2004 is presently being used a baseline to compare the competencies being offered by different programs in software engineering. The 2009 update is expected to include a number of topics not covered in the 2004 version. The update may be an addendum to the existing version and not a fully revised release.

"SE-2004", Thomas Hilburn

Thomas Hilburn presented the undergraduate reference curriculum "SE-2004" (presentation attached as Appendix C-3). He stated that a number of reviews, including two public reviews, were held before release of the curriculum. Unlike the graduate curriculum now being prepared, "SE-2004" addresses undergraduate students who would be entering the program without any prior experience or computing knowledge. This make the entry conditions for undergraduate programs more uniform than is possible for graduate programs. "SE-2004" also includes curriculum patterns applicable to various countries and regions.

"SEI – 1991 Curriculum", Mark Ardis

Mark Ardis presented the reference curriculum that was prepared by the SEI in their 1991 report on graduate software engineering education (presentation attached as Appendix C-4). He added that dissemination of the curriculum was also given much thought. Tutorial videos were recorded and distributed to universities across the world, encouraging others to use it. Mark stated that this effort met with only limited success, for unclear reasons.

"Systems Engineering Curriculum", Art Pyster

Art Pyster presented a summary of the Reference Curriculum for a Graduate Program in Systems Engineering prepared by Professor Rashmi Jain and others at Stevens Institute of Technology (presentation attached as Appendix C-5). He noted that many competencies were common between systems engineering and software engineering, and that there was tremendous opportunity for collaboration between the two disciplines. There were many touch points between software and systems

engineering that the software engineering curricula that were presented and discussed later at the workshop did not articulate well.

3.3 As-Is Analysis

In an attempt to understand current practice, iSSEc is conducting an analysis of software engineering graduate programs from around the world. As of the beginning of the workshop, 11 programs had been analyzed. Art Pyster presented details of the data, collection approach used and some preliminary analysis carried out with the data (attached as Appendix C-6). For each university, data was collected from the university website and further validated with a professor at the university.

Initial analysis clearly showed the large diversity in software engineering programs not just in course content, but also in a number of other factors including admission requirements, project/thesis requirements and percentage of total courses that are required.

The data collection and validation exercise will be continued to include a total of at least 30 universities. Analysis results will be published. Participants were requested to suggest names of programs and universities to be considered for data analysis, with priority to those where some direct contacts exist. Mark Ardis stated that during an earlier data collection exercise, 52 US universities were found to offer a graduate program in software engineering. Out of those 52 universities, 13 were accredited.

3.4 Expected Competencies

Workshop participants spent substantial time during both days of the workshop in deliberating which competencies (both knowledge and skills) that a student in a master's program in software engineering should be expected to master. A homework assignment had been given to all participants before the workshop, requesting them to list out the graduate curriculum requirements from their perspective. Gary Hafen and Bret Michael presented their perspectives, which initiated discussions between the participants (attached as Appendices C-7 and C-8).

SWEBOK 2004, which has been used as a base for the As-Is analysis, relates to software development of the 1990s, and is not mindful of more recent advances, including open source, agility, and geographically distributed development. It may be too big an effort to take a completely fresh look at the competencies required. A simpler and more efficient approach will be to add and remove competencies relative to SWEBOK 2004 as appropriate.

Homework assignments submitted by David Weiss and Barry Boehm (attached as Appendices D-1 and D-2), and the document by Mary Shaw titled "Software Engineering for the 21st Century: A basis for rethinking the curriculum" were the basis

for further discussions. Participants voted as to which of 4 categories specific knowledge ("what the student is expected to know") and skills ("what the student should be able to do") should belong to: (i) pre-requisite to entering the Masters program (ii) critical for earning a Masters degree (iii) post Masters Degree (Masters Degree + 2 years) (iv) get rid of it – not a necessary skill or knowledge. Results of this voting exercise and its impact on the strawman curriculum will be published later.

Larry Bernstein raised a few questions for discussion, and provided a short list of literature that he believes every software engineer should read (attached as D-3). Thomas Hilburn posed some questions relevant to establishing requirements and also listed a set of high-level requirements (attached as D-4). Osmo Vikman expressed his views about the whole approach, based on his experience at Nokia (attached as D-5).

The following directions were generally agreed upon by all participants towards preparation of the strawman curriculum and the complete model graduate reference curriculum to follow:

1. The curriculum will identify a common foundation that all schools should include in their core curricula.
2. Students need to have knowledge of either a specific application domain (such as telecommunications or finance) or a specific disciplinary domain (such as information security).
3. Students need the experience of building real software. Some ways of doing this are: on-the-job, in a thesis, or in a school-based project.
4. Students need to learn how they can make a transition from one application or disciplinary domain to another, since so many will need to do so during their career.
5. Students need to recognize the boundaries of what they know and what they don't know, and know what to do when they encounter a situation beyond their area of competence.

3.5 Strawman Curriculum Outline

A draft strawman curriculum outline was proposed by Stevens, which was discussed and modified.

Bret Michael offered that although the As-Is analysis could be published and presented in other forms, the strawman curriculum should include an executive summary of the analysis. David Weiss stated that the rationale behind the most important decisions made in creating the curriculum should be included. Tom Hilburn added that some guidelines on infrastructure, faculty, teaching, practicum and

assessment would benefit universities whey they plan to adopt and implement the reference curriculum.

The strawman curriculum will be released near the end of 2007. The strawman curriculum will also serve as a guide to preparation of the complete graduate model reference curriculum involving a larger Core Team and many more reviewers and participants.

The agreed outline is attached in Appendix E. Participants offered to prepare various sections of the strawman curriculum and to review the contents.

3.6 Curriculum Endorsements and Dissemination

A variety of dissemination methods need to be employed – journals, conferences, presentations, publications, and the web. Suggestions included:

- iSSEc could be mentioned in the IEEE computer society meeting to be held in November 2007
- Since systems engineering will be included in phase 1 product of iSSEc (model graduate reference curriculum for software engineering), endorsement from INCOSE and the IEEE Systems Council should be possible.
- The curriculum should not be seen as a DoD publication even though DoD is providing significant funding for the effort. This constraint precludes, for example, publishing the curriculum as an SEI Technical Report. The curriculum will have broad applicability, and therefore, needs to be disseminated in media read by commercial organizations.
- A "summer institute" or the like could be organized to present the curriculum and guidelines for adoption to universities. NSF could possibly sponsor this event.
- The following associations could be contacted for endorsement:

  ACM (including SIGCSE and SIGSOFT), IEEE (including the Computer Society, Systems Council, and Software Engineering Council), ISO, ASEE, INCOSE, NDIA SE Division, AIAA, SAE, and non-US analogs of societies such as ASEE.

  In fact, the NDIA SE Division and INCOSE SE University Leadership have already endorsed this effort.
- A publication plan should be prepared, listing the journals, newsletters, websites, and magazines where iSSEc could be published

3.7 Expanding the Team

Participants expressed a number of views concerning addition of more members to the iSSEc project team. While most of them would be included at the Core Team level, some can also be brought in to the EST:

- David Weiss observed that the average age of the current EST was too high, and expressed the need to include younger participants, especially from academia. He also added that there should be some representation from Silicon Valley, especially from start-up companies.
- Paul Robitaille offered to contact BMW for representation from the auto industry.
- Many other participants agreed to identify contacts from industry who could be part of the iSSEc team.

3.8 General Points

Some general points raised by the participants of the workshop not included in the preceding sections, are mentioned below:

- Paul Robitaille cautioned that the graduate curriculum should not simply "clean up" at the graduate, failures in education at the undergraduate level.
- Larry Bernstein stated that the market demand of the software industry is high enough that students can get excellent jobs without a formal degree in software engineering. Hence, attracting students to a Master of Software Engineering (MSE) course or equivalent could be a big challenge for the universities. Paul Robitaille added that attracting students to engineering in general was a challenge, since there were less strenuous ways for students to obtain a good paycheck.
- While software engineering in most universities worldwide falls under the Computer Science (CS) department, Larry Bernstein noted that at Stevens the SwE department has recently been moved from CS to the School of Systems and Enterprises where the systems engineering program is housed.
- Art Pyster noted that the role of today's software engineer is more of assembling components than of building components.
- The IEEE's Certified Software Development Professional (CSDP) program was identified as a reasonable measure of the minimum level of software engineering a software professional has learned. It was further observed that companies are

unwilling to pay for test preparation and the cost of the test because there is no customer requirement for it. It is recommended that iSSEc define a means of identifying and tracking those professionals that take the exam.

Some thoughts on the final curriculum were discussed:

- The model curriculum should be architected to readily enable an individual university to add a domain-specific focus or other specific flavor to the curriculum.
- The curriculum must strike a balance between breadth and depth in what is presented. This will be difficult.
- A thesis or capstone project can be the means for a student to dive deeply into a specific topic such as software for real-time systems or software for safety-critical systems.
- Disciplinary domains (such as security, which applies in many application domains) and application domains (such as finance and telecommunications) are distinct. This distinction needs to be understood by all students.
- While course packaging need not be presented exhaustively, two sample course packagings could be included as examples.

4. Way Ahead

- A project plan for the strawman curriculum would be prepared by Stevens.
- Teams would interact independently towards preparing the strawman curriculum. Following a couple of intermediate versions, the final strawman curriculum is expected to be released around the end of 2007 or early 2008, probably after another meeting of the EST.
- Meanwhile, the larger Core Team would be established. The Core Team could first meet in early 2008 and then decide how to evolve the strawman curriculum into a full-fledged model graduate reference curriculum in software engineering.
- The As-Is analysis would be continued with data collection from more universities. A paper would be published on its findings, which would also feed into the strawman curriculum.

Appendix A: Workshop Objectives

1. Organize the Early Start Team (EST) in a way that best positions the team to create a strawman graduate software engineering (SwE) curriculum during the fall

2. Develop a shared understanding of the current state (as-is) of graduate SwE programs (a preliminary analysis of current graduate programs will be discussed – but a final version will be generated during the fall.)

3. Agree on a draft set of requirements for a model graduate SwE curriculum (to-be curriculum), including what minimum knowledge and skills should be expected of anyone receiving an MS in SwE (or equivalent degree). In deciding which requirements to include, the EST should think broadly, considering "soft" skills and knowledge, such as the ability to work effectively in teams, technical writing skills, ethical behavior, etc. The EST should also consider what domain-specific and systems engineering knowledge and skills they require from SwE graduates.

4. Develop a rough analysis of the gap between the as-is state of graduate curricula (understood by satisfying Objective 2) and a model curriculum (understood by satisfying Objective 3).

5. Develop a draft outline/architecture of the model curriculum

6. Agree on success conditions for iSSEc; that is, when the project is completed, what would we like to be true?

7. Agree on a high-level plan to produce a strawman curriculum

8. Identify ways in which strawman and version 1 curricula will be used

Appendix B: Workshop Agenda

Day 1: August 15th, 2007 (Wed)

8:00a *Continental Breakfast*

8.30a General Introduction of the EST and discussion of the workshop objectives and agenda

9.15a Review primary source documents, including SWEBOK, SE2004, 1991 SEI Technical Report on graduate SwE education, CSDP and the model systems engineering curriculum developed by Jain, et al.

10:30a *Break*

10.45a Review analysis of as-is state of graduate SwE programs

11:15a Develop a draft set of requirements for model curriculum, including an understanding of what students should know when entering the graduate program; results from the homework assignment to think through Objective 3 will be used to seed the discussion.

12:00p *Lunch*

1:00p Continue developing requirements

2.00p Determine the metric to be used for assessing the gap between the as-is state of SwE graduate programs and what the requirements demand. Analyze the gap using the established metric.

3:00p *Break*

3:15p Continue gap analysis

4:15p Begin developing an outline/architecture of the model curriculum

5:00p *Adjourn for the day*

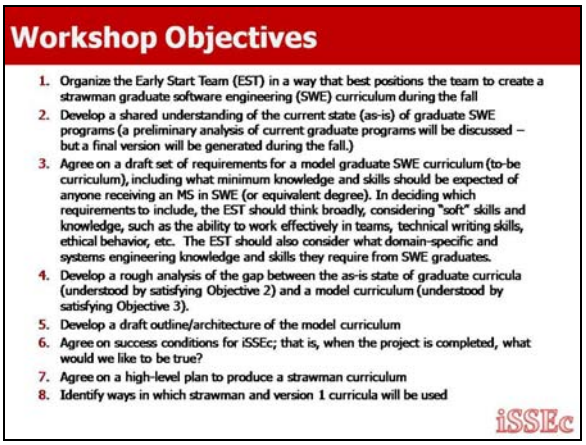
6:00p Group supper for those interested

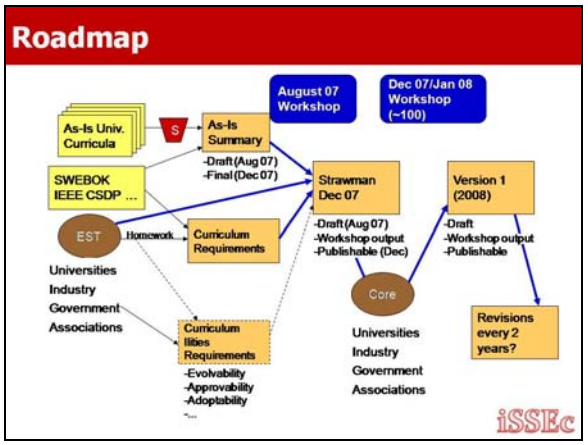
Day 2: August 16th, 2007 (Thu)

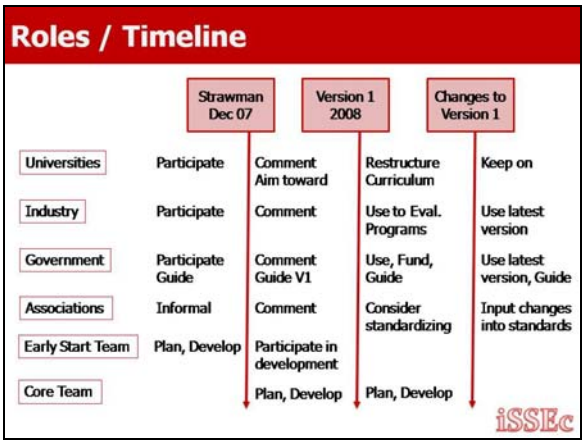
8:00a	*Continental breakfast*
8.30a	Review the previous day and make any mid-course corrections
9:00a	Continue outline/architecture of the model curriculum
11.00a	Develop & agree on iSSEc Project success conditions
11.30a	*Lunch*
12:15p	Develop high-level implementation plan for strawman curriculum
1:30p	Wrap up and action item review
2:00p	Adjourn workshop

Appendix C: Presentations

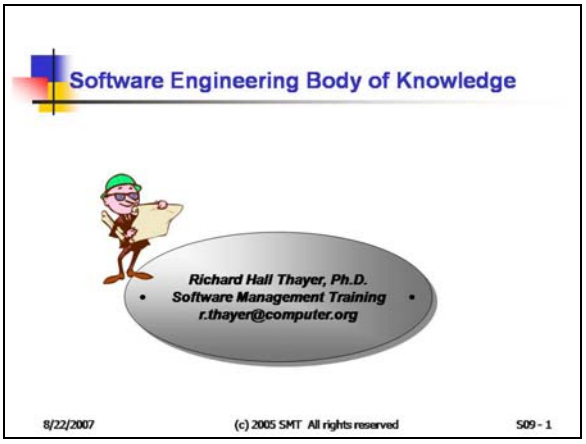
C-1: "iSSEc project"; Art Pyster

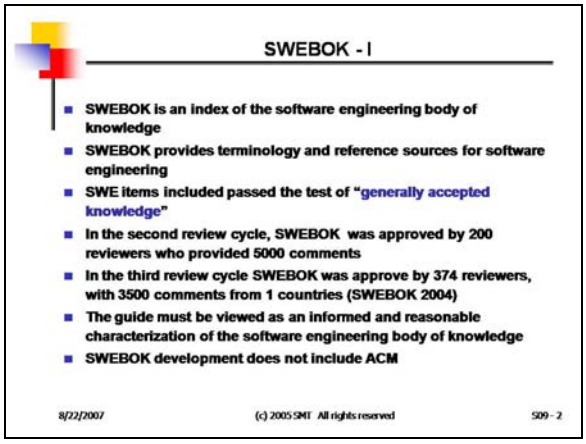


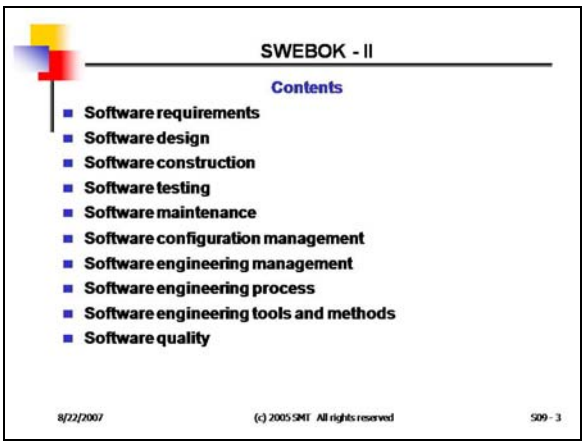


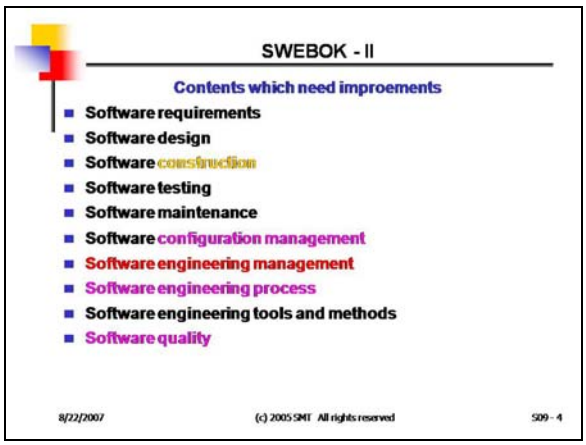


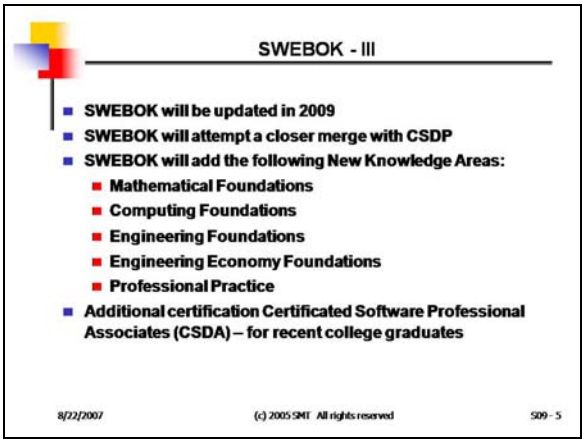
C-2: "SWEBOK"; Richard Thayer











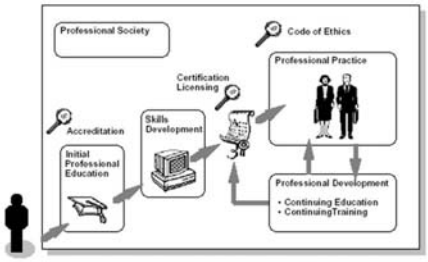
C-3: "SE-2004"; Thomas Hilburn

Software Engineering 2004
Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (SE2004)

integrated Software and Systems Engineering curriculum (iSSEc) Project
Early Start Team (EST) Workshop
August 15-16, 2007

1

Ford-Gibbs Model



Ford, G. and Gibbs, N. E., *A Mature Profession of Software Engineering*, CMU/SEI-96-TR-004, CMU, 1996.

2

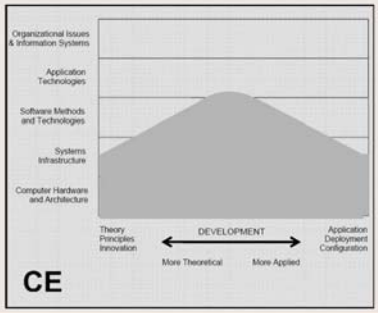
Computing Curricula 2001

- Beginning in 1968, the Association for Computing Machinery (ACM) – later joined by the IEEE-CS – has developed guidelines for computer science programs.
- In 1998, the ACM and the IEEE-CS convened a joint-curriculum task force called Computing Curricula 2001(CC2001).
  - goal to develop curricular guidelines that would "match the latest developments of computing technologies in the past decade and endure through the next decade"
  - computer science volume in 2001
  - additional volumes in computer engineering, information systems, and software engineering since 2001
  - information technology currently in review
  - *Overview Report* discusses and defines computing and its various disciplines. It uses graphical figures to describe similarities and differences in computing disciplines

http://www.computer.org/education/cc2001/

3

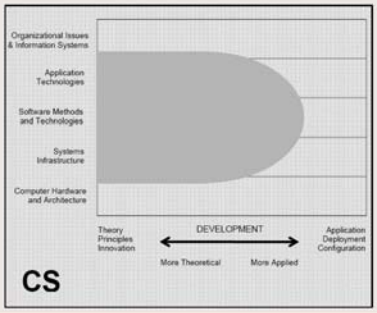
Computer Engineering



CE

4

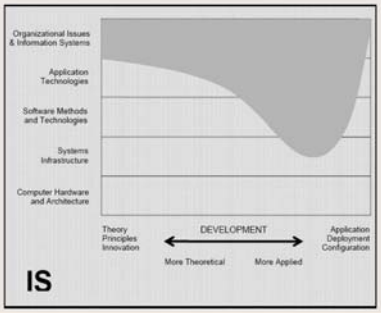
Computer Science



CS

5

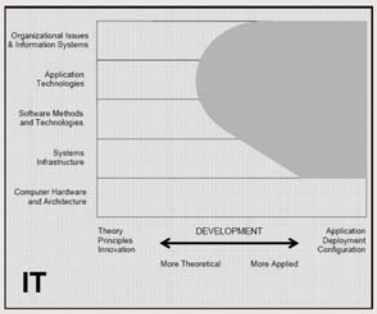
Information Systems



IS

6

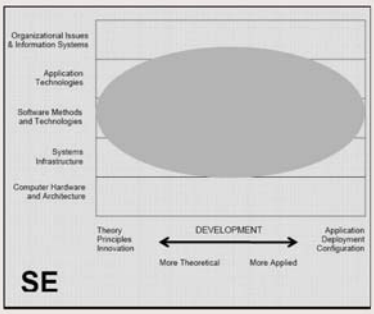
Information Technology



IT

7

Software Engineering



SE

8

SE2004 Project

- Spring of 2002, the SE2204 Steering Committee
  - determined the project scope and developed a high-level plan and schedule
  - established guiding principles for the project
  - formulated expected outcomes for graduates of BSSE programs
  - solicited volunteers and set up two working groups:
    - the Education Knowledge Area Group to develop an initial body of Software Engineering Education Knowledge (SEEK)
    - the Pedagogy Focus Area Group to produce the curriculum recommendations using the SEEK as a foundation.
- Fall 2002, Spring 2003
  - SEEK was developed and submitted for public review.
  - Curriculum guidance was developed and reviewed
- Summer and Fall of 2003, Spring 2004
  - full draft of the SE2004 volume was developed and submitted for public review
  - revised draft was completed and reviewed in the winter and spring of 2004.
- Summer 2004
  - SE2004 volume submitted to the education boards of the ACM and IEEE-CS
  - approved for distribution in August 2004.

9

SE2004 Guiding Principles

1. Computing is a broad field that extends well beyond the boundaries of any one computing discipline.
2. Software Engineering draws its foundations from a wide variety of disciplines.
3. The rapid evolution and the professional nature of software engineering require an ongoing review of the corresponding curriculum.

…

11. SE2004 must include discussions of strategies and tactics for implementation, along with high-level recommendations.

10

SE2004 Student Outcomes

Graduates of an undergraduate SE program must be able to:

1. Show mastery of the software engineering knowledge and skills, and professional issues necessary to begin practice as a software engineer.
2. Work as an individual and as part of a team to develop and deliver quality software artifacts.
3. Reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.
4. Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.
5. Demonstrate an understanding of and apply current theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification, and documentation.
6. Demonstrate an understanding and appreciation for the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.
7. Learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.

11

SEEK Organization

- The SEEK (Software Engineering Education Knowledge) describes the body of knowledge that is appropriate for an undergraduate program in software engineering.
  - The term "knowledge" is used to describe the whole spectrum of content for the discipline: information, terminology, artifacts, data, roles, methods, models, procedures, techniques, practices, processes, and literature.
  - The initial selection of the SEEK knowledge areas was based on the SWEBOK knowledge areas and discussions with the Education Knowledge Area Group volunteers.
  - Subsequent modifications of the SEEK were based on an two-day SEEK workshop, review by a group of experts, and extensive public review.
  - The SEEK is organized into three hierarchical levels:
    - Knowledge Areas
    - Units
    - Topics.

12

SEEK Knowledge Areas

KA	Title	Hours
CMP	Computing Essentials	172
FND	Mathematical & Engineering Fundamentals	89
PRF	Professional Practice	35
MAA	Software Modeling & Analysis	53
DES	Software Design	45
VAV	Software V & V	42
EVL	Software Evolution	10
PRO	Software Process	13
QUA	Software Quality	16
MGT	Software Management	19
	total	494

13

Curriculum Guidelines

The CCSE volume offers a set of guidelines for the design and delivery of an SE curriculum.

1. Curriculum designers and instructors must have sufficient relevant knowledge and experience and understand the character of software engineering.
2. Curriculum designers and instructors must think in terms of outcomes.
3. Curriculum designers must strike an appropriate balance between coverage of material, and flexibility to allow for innovation.
4. Many SE concepts, principles, and issues should be taught as recurring themes throughout the curriculum to help students develop a software engineering mindset.
5. Learning certain software engineering topics requires maturity, so these topics should be taught towards the end of the curriculum, while other material should be taught earlier to facilitate gaining that maturity.

14

Curriculum Guidelines

6. Students must learn some application domain (or domains) outside of software engineering.
7. Software engineering must be taught in ways that recognize it is both a computing and an engineering discipline.
8. Students should be trained in certain personal skills that transcend the subject matter.
9. Students should be instilled with the ability and eagerness to learn.
10. Software engineering must be taught as a problem-solving discipline.
11. The underlying and enduring *principles* of software engineering should be emphasized, rather than *details* of the latest or specific tools.

15

Curriculum Guidelines

12. The curriculum must be taught so that students gain experience using appropriate and up-to-date tools, even though tool details are not the focus of the learning.
13. Material taught in a software engineering program should, where possible, be grounded in sound research and mathematical or scientific theory, or else widely accepted good practice.
14. The curriculum should have a significant real-world basis.
15. Ethical, legal, and economic concerns, and the notion of what it means to be a professional, should be raised frequently.

16

Curriculum Guidelines

16. In order to ensure that students embrace certain important ideas, care must be taken to motivate students by using interesting, concrete and convincing examples.
17. Software engineering education in the 21st century needs to move beyond the lecture format: It is therefore important to encourage consideration of a variety of teaching and learning approaches.
18. Important efficiencies and synergies can be achieved by designing curricula so that several types of knowledge are learned at the same time.
19. Courses and curricula must be reviewed and updated regularly.

17

SE2004 Curriculum Material

- Using the SEEK, the curriculum guidelines, the CC2001 CS volume, and ideas and recommendations from the SE2004 volunteers, the following curriculum materials were developed for the SE2004 volume:
  - a set of curriculum "patterns" that outlined various curriculum structures that would serve the needs of different constituencies. Each pattern includes a set of courses that cover the core SEEK material
  - descriptions of courses, including a short description, learning objectives and a list of topics

18

North American - CS Pattern

SE201 Introduction to Software Engineering
SE211(A) Software Construction
SE212 (B) Software Engineering Approach to Human Computer Interaction
SE311 (D) Software Design and Architecture
SE321 (C) Software Quality Assurance and Testing
SE322 (E) Software Requirements Analysis
SE323 (F) Software Project Management
SE400 Software Engineering Capstone Project

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
CS101	CS102	CS103	CS220	CS226	CS270T	SE400	SE400
Calc 1	*Calc 2*	CS106	SE A	MA271	SE D	SE F	*Tech elect*
NT181	CS105	SE201	SE B	SE C	SE E	*Tech elect*	*Tech elect*
Physics	*Any science*	NT272	*Linear Alg*	NT291	*Tech elect*	*Tech elect*	*Tech elect*
Gen ed	*Gen ed*	--	*Gen ed*	*Gen ed*	*Gen ed*	*Gen ed*	*Gen ed*

19

Two Related Efforts

- ABET Accreditation of Software Engineering programs (http://www.abet.org/)
  - Student Outcomes (a through k)
  - Program Specific Criteria
- Certified Software Development Professional (CSDP) (http://www.computer.org/)
  - CSDP related to SWEBOK
  - Certified Software Development Associate (CSDA)

20

Famous Software Engineering Educator

"90% of a software engineering curriculum should be devoted to fundamentals; the other half should be in a practicum."

21

C-4: "SEI-1991 Curriculum"; Mark Ardis

1991 SEI Report on Graduate Software Engineering Education

Mark Ardis
Rochester Institute of Technology

Main contents of Report

- Model Curriculum
- Video Dissemination Project Courses
- Survey of Graduate Programs

2

Model Curriculum

- 1985-87: Requirements Gathering
- 1987: Specification - 20 Topics
- 1988: Design - 6 Core Courses
- 1988-90: Implementation at CMU

3

20 Specified Topics

- Soft. Operational Issues
- Requirements Analysis
- System Design
- System Integration
- Human Interfaces
- Specification
- Software Design
- Embed. Real-Time Sys.
- Software Generation
- Software Maintenance
- Soft. Implementation
- Software Quality Issues
- Software Qual. Assurance
- Software Testing
- SE Process
- Software Evolution
- Tech. Communication
- Software Config. Mgmt.
- Software Qual. Assurance
- Soft. Proj. Organ. & Mgmt.
- Soft. Proj. Economics

4

5 Groups

Soft. Operational Issues
Requirements Analysis
System Design
System Integration
Human Interfaces

Specification
Software Design
Embed. Real-Time Sys.

Software Generation
Software Maintenance
Soft. Implementation

Software Quality Issues
Software Qual. Assurance
Software Testing

SE Process
Software Evolution
Tech. Communication
Software Config. Mgmt.
Software Qual. Assurance
Soft. Proj. Organ. & Mgmt.
Soft. Proj. Economics

5

Sizing of Topics

Soft. Operational Issues-s
Requirements Analysis-M
System Design-s
System Integration-s
Human Interfaces-s

Specification-L
Software Design-L
Embed. Real-Time Sys.-M

Software Generation-s
Software Maintenance-M
Soft. Implementation-M

Software Quality Issues-M
Software Qual. Assurance-M
Software Testing-M

SE Process-s
Software Evolution-s
Tech. Communication-M
Software Config. Mgmt.-s
Software Qual. Assurance-s
Soft. Proj. Organ. & Mgmt.-M
Soft. Proj. Economics-s

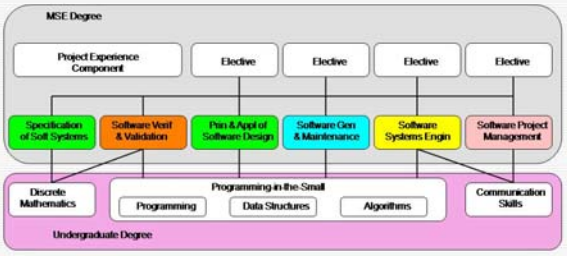
6

6 Core Courses

- Software Systems Engineering
- Specification of Software Systems
- Principles and App.s of Software Design
- Software Generation and Maintenance
- Software Verification and Validation
- Software Project Management

7

Curriculum Structure



8

Video Dissemination Project

- Core courses for CMU MSE
- Video-taped and distributed to other schools

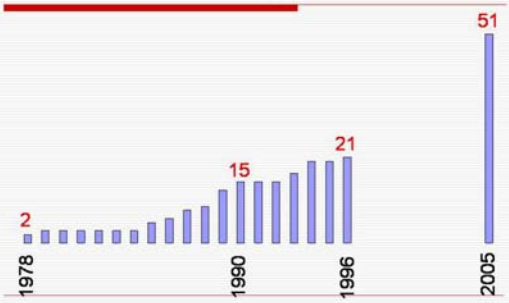
9

Survey of Graduate Programs

- Seattle University
- Texas Christian University
- Wang Institute
- University of St. Thomas
- National University
- Monmouth University
- Andrews University
- Rochester Inst. of Tech.
- Boston University
- Carnegie Mellon University
- George Mason University
- Florida Atlantic University
- University of Pittsburgh
- University of West Florida
- Univ. of Houston-Clear Lake
- University of Scranton

10

Growth in MSE Programs



11

Estimated Graduation Rates (21/51 programs)

12

C-5: "Systems Engineering Curriculum"; Art Pyster

Summary of "Reference Curriculum for a Graduate Program in Systems Engineering" by Jain, et al.

Art Pyster

Stevens Institute of Technology

Background

- Analogous to what we are doing, but for Systems Engineering -- though at a much higher-level
- Supported by SE Curriculum Working Group within INCOSE
- Studied 36 universities in US and Europe
- Methodology analogous to ours
- Product is at a high-level. Course descriptions in total are 4 pages
- Competency model, based on an amalgam of sources, including INCOSE Handbook, is 3 pages.

Schools Surveyed

#	Institution	B*	M*	P*	Degree
1	Air Force Institute of Technology		X	X	M.S., Ph.D. in Systems Engineering
2	Boston University		X	X	M.S., Ph.D. in Systems Engineering
3	California State University- Fullerton		X		M.S. Option in Systems Engineering
4	Colorado School of Mines		X	X	M.E., M.S., Ph.D. in Engineering Systems
5	Cornell University		X		M.E., Systems Engineering Option
6	Florida Institute of Technology		X		M.S. in Systems Engineering
7	George Mason University	X	X		B.S., M.S. in Systems Engineering
8	George Washington University	X	X	X	B.S., M.S., Ph.D. in Systems Analysis and Engineering
9	Iowa State University		X		M.S. in Systems Engineering
10	Johns Hopkins University		X		M.S. in Systems Engineering, Post Masters Certificate
11	Loyola Marymount University		X		M.S. in Systems Engineering
12	Naval Postgraduate School		X		M.S. in Systems Engineering
13	Oakland University	X	X	X	B.S., M.S., Ph.D. in Systems Engineering
14	Old Dominion University		X		M.S. in Systems Engineering
15	Pennsylvania State University- Valley		X		M.E. in Systems engineering
16	Polytechnic University- Farmingdale		X		M.S. in Systems Engineering and Integration
17	Portland State University		X		M.S. in Systems Engineering
18	Rochester Institute of Technology		X		M.S. in Systems Engineering
19	Southern Methodist University		X		M.S. in Systems Engineering
20	Southern Polytechnic State University		X		M.S. in Systems Engineering
21	State University of NY - Binghamton		X		M.S. in Systems Engineering
22	Stevens Institute of Technology		X	X	M.S., Ph.D. in Systems Engineering
23	University of Alabama - Huntsville		X	X	M.S.E., Ph.D. in Systems Engineering
24	University of Arizona	X	X	X	B.S., M.S., Ph.D. in Systems Engineering
25	*University of Arkansas - Little Rock*	*X*			*B.S. in Systems Engineering*
26	University of Idaho		X		M.E. Systems Engineering & Systems Thinking
27	University of Maryland		X	X	M.E., M.S., Ph.D. in Systems Engineering
28	University of Missouri - Rolla		X		M.S. in Systems Engineering
29	University of Pennsylvania	X	X	X	B.S., M.S.E., Ph.D. in System Science and Engineering
30	University of Southern California		X		M.S. in Systems Architecture and Engineering
31	University of Virginia	X	X	X	B.S., M.S., Ph.D. in Systems Engineering
32	*U.S. Military Academy*	*X*			*B.S. in Systems Engineering*
33	*U.S. Naval Academy*	*X*			*B.S. In Systems Engineering*
34	Virginia Tech		X		M.E., M.S. in Systems Engineering
35	Washington University		X		M.S. in Systems Science and Engineering

Sample SE Competencies

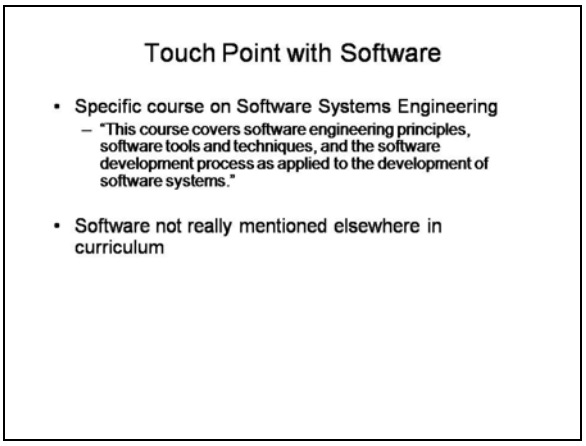
SE Competency	Definition
Systems Thinking	Systems Thinking contains the underpinning systems concepts and the system/super-system skills including the business and technological environment.
Systems concepts	The application of the fundamental concepts of systems thinking to systems engineering. These include understanding what a system is, its context within its environment, its boundaries and interfaces and that it has a lifecycle.
Super-system capability issues	An appreciation of the role the system plays in the super system of which it is a part.
Business and technology environment	The definition, development and production of systems within an enterprise and technological environment.
Holistic Lifecycle view	Holistic Lifecycle View contains all the skills associated the systems lifecycle from need identification, requirements through to operation and ultimately disposal.
Determine and manage stakeholder requirements	To analyze the stakeholder needs and expectations to establish and manage the requirements for a system.
System Requirements	To translate the stakeholder needs and expectations for the system into system requirements such that it reflects the true needs of the stakeholders.
System Design	
Architecture Design	The definition of the system architecture and derived requirements to produce a solution that can be implemented to enable a balanced and optimum work that considers all stakeholder requirements (business, technical(IE.).
Concept generation	The generation of potential system solutions that meet a set of needs and demonstration that one or more could be feasible within a cost.
Design for requirements of total life cycle stages	Ensuring that the requirements of later lifecycle stages are addressed within the context point in the system design. During the design process considerations should be given to manufacturability, testability, reliability, maintainability, safety, security, flexibility, interoperability, capability growth, disposal, etc.
Functional analysis	Analysis is used to determine which functions are required by the systems to meet the requirements. It transforms the requirements into a coherent description of system functions and the interfaces that can be used to guide the design activity that follows. It consists of the decomposition of higher level functions to lower levels and the traceability allocation of requirements to those functions.

Gap Analysis in Current SE Programs

- SE competencies that are not adequately addressed in current courses:
  - System concepts, architectural design, modeling and simulation; system requirements; determine and manage stakeholder requirements; super-system capability issues
- Competencies covered in intro courses, but not well-developed in follow on courses:
  - General project management; finance, economics, and cost estimation; organizational leadership

Course Recommendations

- Level 0: Foundation Courses (Ramp-up)
  - General Mathematics; Probability and Statistics
- Level 1: Introduction Courses (Take first)
  - Fundamentals of SE; Intro to SE Management
- Level 2: Core Courses (Required)
  - Systems Design/Architecture; Systems Integration and Test; Quality, Safety, and Systems Suitability; Modeling, Simulation, and Optimization; Decisions, Risks, and Uncertainty; Software Systems Engineering
- Level 3: Specialization Courses
  - General PM; Finance, Economics, and Cost Estimation; Manufacturing, Production, and Operations; Organizational Leadership; Engineering Ethics and Legal Considerations; Masters Project or Seminar


Touch Point with Software
• Specific course on Software Systems Engineering
– "This course covers software engineering principles, software tools and techniques, and the software development process as applied to the development of software systems."
• Software not really mentioned elsewhere in curriculum

C-6: "Analysis of existing SwE programs"; Art Pyster

iSSEc

Integrated Software and Systems Engineering Curriculum

Analysis of Existing SwE programs

Aug 15th & 16th — Arlington, Virginia

STEVENS Institute of Technology

ASysT INSTITUTE

SwE Programs (Schools)

Validated

1. Air Force Institute of Technology
2. California State University - Sacramento
3. Embry-Riddle Aeronautical University
4. George Mason University
5. Monmouth University
6. Naval Postgraduate School
7. Seattle University
8. Stevens Institute of Technology
9. University of Alabama - Huntsville
10. University of Southern California
11. University of York (UK)

Under Process

1. James Madison University
2. Blekinge Institute of Technology (Swe)
3. Carnegie Mellon University
4. Dublin City University (UK)
5. Kansas State University
6. Kingston University (UK)
7. Southern Methodist University
8. Texas Tech University
9. University of Melbourne (Aus)
10. University of Scranton
11. University of Sunderland (UK)

Others to be determined

2 iSSEc

Data Collection & Analysis

- Select Program / University
- Collect Data (from Internet):
  - School, Degree
  - Courses & Descriptions
  - SWEBOK Competency Mapping
- Validate with University
- Analyze

3 iSSEc

Spreadsheet for Data Collection

- School, Degree

4 iSSEc

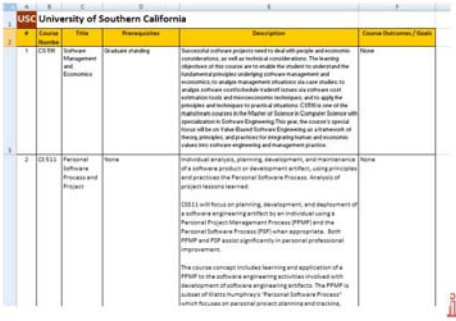
Spreadsheet for Data Collection

- Degree, Courses

5 iSSEc

Spreadsheet for Data Collection

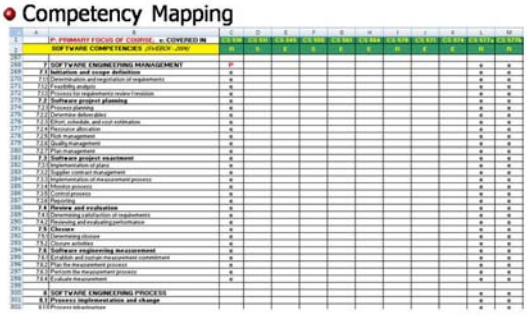
- Course Descriptions



6 iSSEc

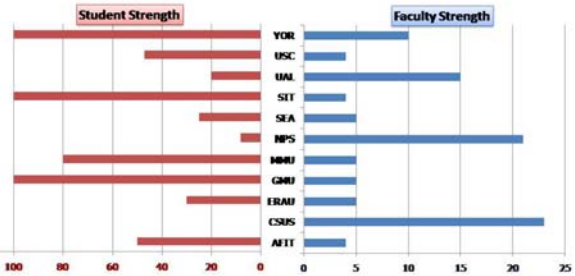
Spreadsheet for Data Collection

- Competency Mapping



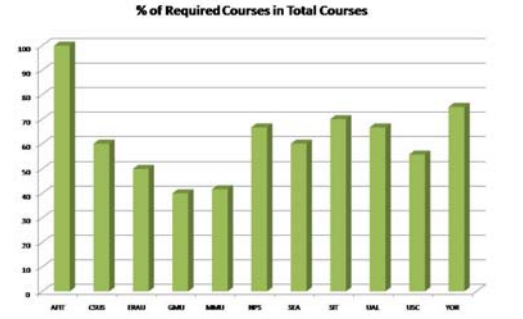
7 iSSEc

Program



8 iSSEc

Courses



9 iSSEc

Program Specialty

Air Force Institute of Technology	Defense Systems
California State University – Sacramento	
Embry-Riddle Aeronautical University	Embedded Real-time Software
George Mason University	
Monmouth University	
Naval Postgraduate School	Acquisition of Defense Systems
Seattle University	Project Experience
Stevens Institute of Technology	Quantitative Software Engineering
University of Alabama – Huntsville	
University of Southern California	Quantitative; Software Economics
University of York (UK)	Safety Critical Systems

10 iSSEc

Program Focus

Air Force Institute of Technology	Develop professionals to develop and manage increasingly complex software
California State University – Sacramento	
Embry-Riddle Aeronautical University	How to engineer high-performance sofwtare embedded in aircraft, space and medical systems
George Mason University	Developing and modifying large, complex software systems. Emphasis both technical and management aspects
Monmouth University	Effective Member of software development team
Naval Postgraduate School	Enable acquisition professionals to procure highly dependable, trustworthy software-intensive systems
Seattle University	Understand and apply advanced software engineering principles vital to industry
Stevens Institute of Technology	Realizing software products on time, within budget and with known quality
University of Alabama – Huntsville	Provide fundamentals of software development for members of software development teams
University of Southern California	Prepare students for an industrial leadership career in software engineering and serve as introduction to researchers
University of York (UK)	Software systems with a high requirement for dependability.

11 iSSEc

Target Student

Air Force Institute of Technology	PMs and Sw developers from DoD & other agencies
California State University – Sacramento	UG degree with CS courses
Embry-Riddle Aeronautical University	Strong academic record
George Mason University	UG degree
Monmouth University	UG degree in CS, SwE or Engineering related
Naval Postgraduate School	Acquisition professionals with 2+ years in Software development
Seattle University	UG degree in CS or equivalent 2+ years in Software development
Stevens Institute of Technology	Experienced computer professionals seeking leadership positions
University of Alabama – Huntsville	UG courses in CS, Math & Statistics
University of Southern California	UG degree in CS, Math or Engineering with courses in computing and math
University of York (UK)	UG degree in CS or related with math

12 iSSEc

Project / Thesis / Practicum

Air Force Institute of Technology	practicum
California State University – Sacramento	
Embry-Riddle Aeronautical University	practicum or project
George Mason University	optional (2 course equivalent)
Monmouth University	2 course thesis or practicum required
Naval Postgraduate School	thesis required beyond courses
Seattle University	project required
Stevens Institute of Technology	no thesis
University of Alabama – Huntsville	thesis (2 course equivalent) or comprehensive exam
University of Southern California	thesis optional
University of York (UK)	project required

13 iSSEc

Observations (Programs)

1. SwE is largely viewed as a specialization of CS
2. Faculty size is small; few dedicated SwE Professors
3. Student enrollments are generally small compared to CS and to other engineering disciplines
4. Many programs have specialties such as Defense systems or safety critical systems
5. Target student characteristics vary widely
6. Program focus and desired outcomes vary

14 iSSEc

Observations (Content)

1. SWEBOK alone does not represent the breadth of many program's required courses
2. Some significant topics rarely mentioned
   - Agility, SW economics, systems engineering
3. Some topics are ubiquitous
   - Formal methods, architecture
4. OO is the standard development paradigm that is taught

15 iSSEc

Introduction to Software Engineering

Schools

AFIT	Air Force Institute of Technology	CSE 481
CSUS	California State University - Sacramento	
ERAU	Embry-Riddle Aeronautical University	
GMU	George Mason University	-
MMU	Monmouth University	SE 504
NPS	Naval Postgraduate School	SW 3460
SEA	Seattle University	
SIT	Stevens Institute of Technology	CS 540
UAL	University of Alabama - Huntsville	CS 650
USC	University of Southern California	CS 577a, CS 577b
YOR	University of York (UK)	-

Scale

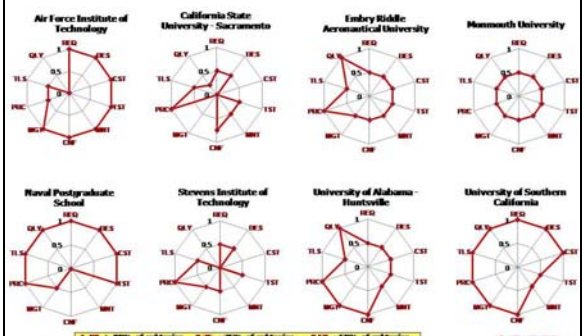
1.00 > 90% of subtopics
0.75 ~75% of subtopics
0.50 ~50% of subtopics
0.25 ~25% of subtopics
0.00 No Coverage

SWEBOK

REQ	Software Requirements
DES	Software Design
CST	Software Construction
TST	Software Testing
MNT	Software Maintenance
CNF	Software Configuration Management
MGT	Software Engineering Management
PRC	Software Engineering Process
TLS	Software Engineering Tools and Methods
QLY	Software Quality

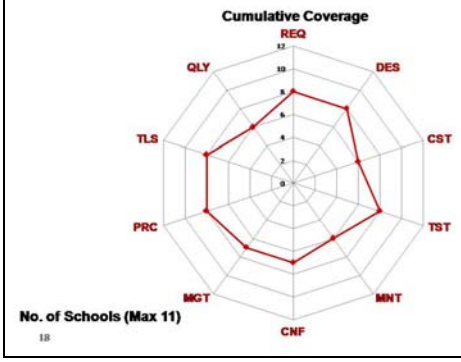
16 iSSEc

Introduction to Software Engineering

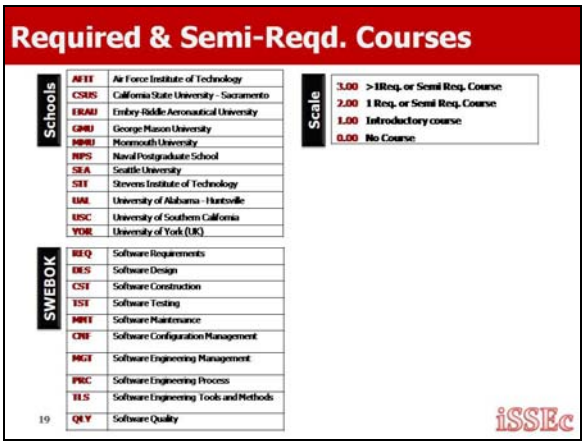


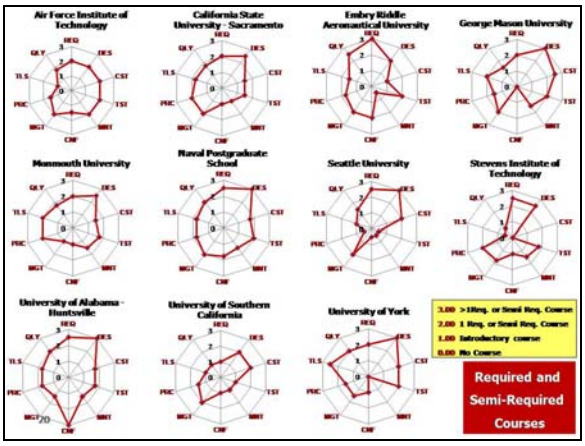
17 iSSEc

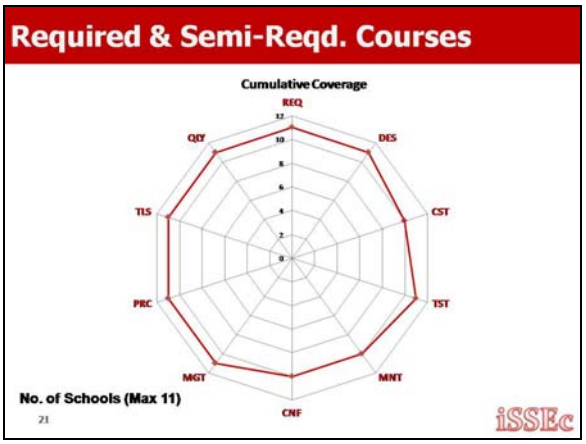
Introduction to Software Engineering

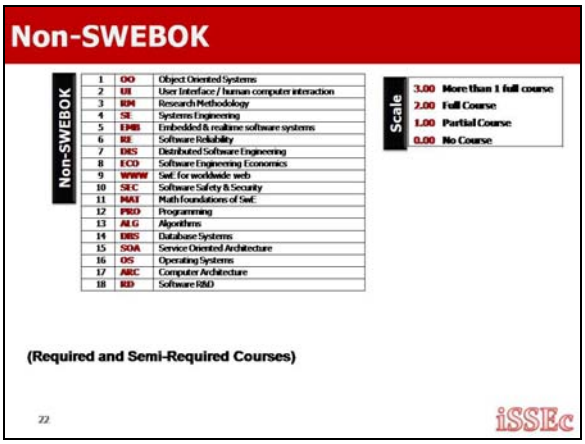


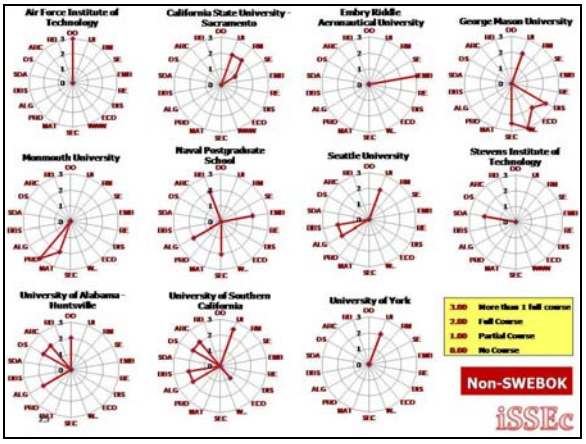
18 iSSEc

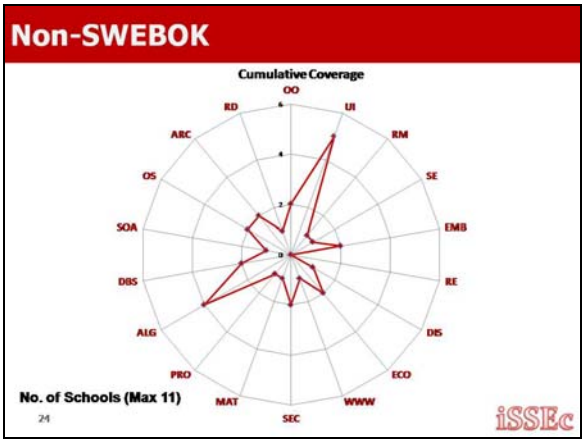

Required & Semi-Reqd. Courses
Schools
AFIT Air Force Institute of Technology
CSUS California State University - Sacramento
ERAU Embry-Riddle Aeronautical University
GMU George Mason University
MMU Monmouth University
NPS Naval Postgraduate School
SEA Seattle University
SIT Stevens Institute of Technology
UAH University of Alabama - Huntsville
USC University of Southern California
YOR University of York (UK)
SWEBOK
REQ Software Requirements
DES Software Design
CST Software Construction
TST Software Testing
MNT Software Maintenance
CNF Software Configuration Management
MGT Software Engineering Management
PRC Software Engineering Process
TLS Software Engineering Tools and Methods
QLY Software Quality
Scale
3.00 >1Req. or Semi Req. Course
2.00 1 Req. or Semi Req. Course
1.00 Introductory course
0.00 No Course
19
iSSEc


Air Force Institute of Technology
California State University - Sacramento
Embry Riddle Aeronautical University
George Mason University
Monmouth University
Naval Postgraduate School
Seattle University
Stevens Institute of Technology
University of Alabama - Huntsville
University of Southern California
University of York
3.00 >1Req. or Semi Req. Course
2.00 1 Req. or Semi Req. Course
1.00 Introductory course
0.00 No Course
Required and Semi-Required Courses
20


Required & Semi-Reqd. Courses
Cumulative Coverage
REQ
DES
CST
TST
MNT
CNF
MGT
PRC
TLS
QLY
No. of Schools (Max 11)
21
iSSEc


Non-SWEBOK
Non-SWEBOK
1 OO Object Oriented Systems
2 UI User Interface / human computer interaction
3 RM Research Methodology
4 SE Systems Engineering
5 EMB Embedded & realtime software systems
6 RE Software Reliability
7 DBS Distributed Software Engineering
8 ECO Software Engineering Economics
9 WWW SwE for worldwide web
10 SEC Software Safety & Security
11 MAT Math foundations of SwE
12 PRO Programming
13 ALG Algorithms
14 DBS Database Systems
15 SOA Service Oriented Architecture
16 OS Operating Systems
17 ARC Computer Architecture
18 RD Software R&D
Scale
3.00 More than 1 full course
2.00 Full Course
1.00 Partial Course
0.00 No Course
(Required and Semi-Required Courses)
22
iSSEc


Air Force Institute of Technology
California State University - Sacramento
Embry Riddle Aeronautical University
George Mason University
Monmouth University
Naval Postgraduate School
Seattle University
Stevens Institute of Technology
University of Alabama - Huntsville
University of Southern California
University of York
3.00 More than 1 full course
2.00 Full Course
1.00 Partial Course
0.00 No Course
Non-SWEBOK
iSSEc


Non-SWEBOK
Cumulative Coverage
OO
UI
RM
SE
EMB
RE
DIS
ECO
WWW
SEC
MAT
PRO
ALG
DBS
SOA
OS
ARC
RD
No. of Schools (Max 11)
24
iSSEc

C-7: "Model Graduate SwE Curriculum - An Industry Perspective"; Gary Hafen

Model Graduate SWE Curriculum – An Industry Perspective

iSSCe Workshop
Arlington, VA
15 August 2007

LOCKHEED MARTIN

Gary Hafen
Corporate Fellow, SW Eng.
Lockheed Martin Corp.

Overall Capability Expectations

- Increased Comprehension or Application of skills above that attained in Undergraduate Curriculum
- Interdisciplinary Skills
  - Systems Thinking (SE Fundamentals)
  - Computer Engineering/Design
  - Organization/Team Building
- Working Experience (Meaningful Internship or apprenticeship as a minimum)

Specific Capabilities

- System Requirements Engineering - K
- System Design (Architecting) - C
- SW Product Management – C
  - Estimating/Planning - C
  - Risk Management - C
  - Baseline Management - C
  - Measurement - C
- SW Development Process and Life Cycle – C
  - Quantitative Management - K
  - Process Improvement - K

K = Knowledge, C= Comprehension, A = Application

Specific Capabilities (Cont.)

- SW Requirements Engineering - A
- SW Arch./Design - A
- SW Implementation - A
- SW Test - A
- SW Integration - A
- System Integration - C
- System Verification/Validation – C
- SW Maintenance - C
- SW CM and QA - C
- SW Database Management - A

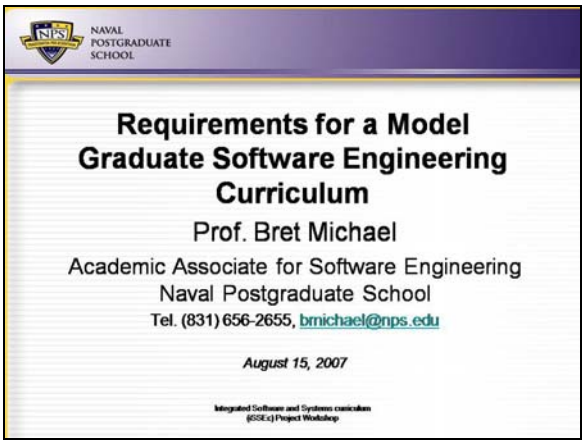
K = Knowledge, C= Comprehension, A = Application

Specific Capabilities (Cont.)

- SW Work Product Realization - A
  - Documentation - A
  - Requirements/Design models - A
  - Software Product Files - A
  - Operational Manuals - A
- Technical Communication Skills - A
  - Technical Reviews - A
  - Presentation/Writing Skills - A
- Professional Ethics - C
  - SW Licensing - C
  - SW Assurance - C

K = Knowledge, C= Comprehension, A = Application

C-8: "Requirements for a Model Graduate SwE Curriculum"; Bret Michael



NAVAL POSTGRADUATE SCHOOL

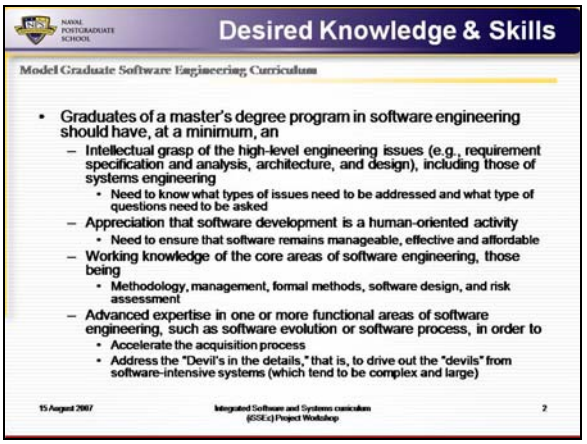
Requirements for a Model Graduate Software Engineering Curriculum

Prof. Bret Michael

Academic Associate for Software Engineering
Naval Postgraduate School
Tel. (831) 656-2655, bmichael@nps.edu

August 15, 2007

Integrated Software and Systems curriculum (iSSEc) Project Workshop

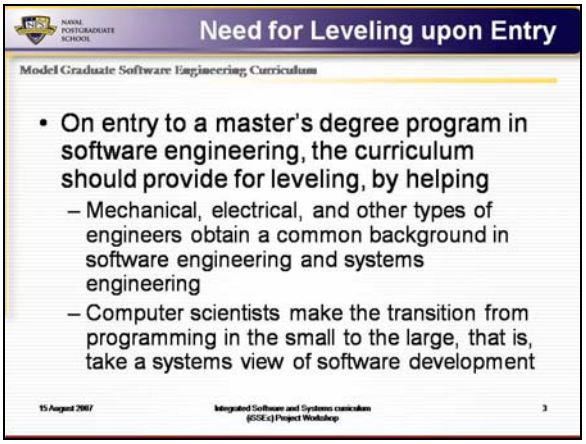


Desired Knowledge & Skills

Model Graduate Software Engineering Curriculum

- Graduates of a master's degree program in software engineering should have, at a minimum, an
  - Intellectual grasp of the high-level engineering issues (e.g., requirement specification and analysis, architecture, and design), including those of systems engineering
    - Need to know what types of issues need to be addressed and what type of questions need to be asked
  - Appreciation that software development is a human-oriented activity
    - Need to ensure that software remains manageable, effective and affordable
  - Working knowledge of the core areas of software engineering, those being
    - Methodology, management, formal methods, software design, and risk assessment
  - Advanced expertise in one or more functional areas of software engineering, such as software evolution or software process, in order to
    - Accelerate the acquisition process
    - Address the "Devil's in the details," that is, to drive out the "devils" from software-intensive systems (which tend to be complex and large)

15 August 2007 — Integrated Software and Systems curriculum (iSSEc) Project Workshop — 2

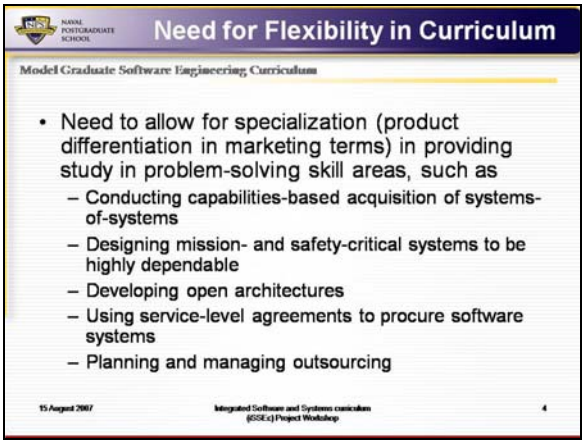


Need for Leveling upon Entry

Model Graduate Software Engineering Curriculum

- On entry to a master's degree program in software engineering, the curriculum should provide for leveling, by helping
  - Mechanical, electrical, and other types of engineers obtain a common background in software engineering and systems engineering
  - Computer scientists make the transition from programming in the small to the large, that is, take a systems view of software development

15 August 2007 — Integrated Software and Systems curriculum (iSSEc) Project Workshop — 3

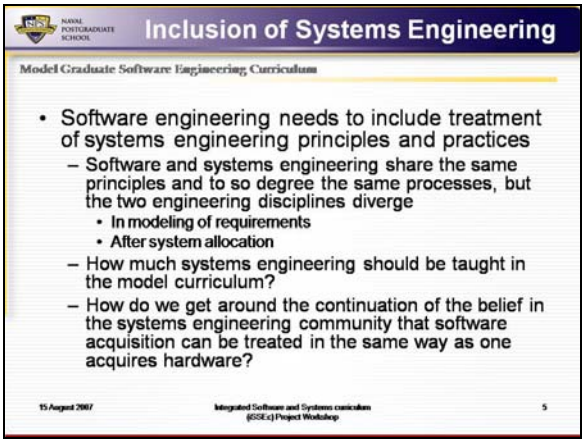


Need for Flexibility in Curriculum

Model Graduate Software Engineering Curriculum

- Need to allow for specialization (product differentiation in marketing terms) in providing study in problem-solving skill areas, such as
  - Conducting capabilities-based acquisition of systems-of-systems
  - Designing mission- and safety-critical systems to be highly dependable
  - Developing open architectures
  - Using service-level agreements to procure software systems
  - Planning and managing outsourcing

15 August 2007 — Integrated Software and Systems curriculum (iSSEc) Project Workshop — 4

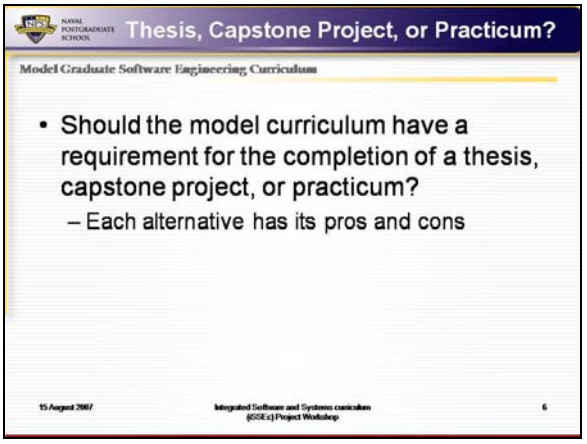


Inclusion of Systems Engineering

Model Graduate Software Engineering Curriculum

- Software engineering needs to include treatment of systems engineering principles and practices
  - Software and systems engineering share the same principles and to so degree the same processes, but the two engineering disciplines diverge
    - In modeling of requirements
    - After system allocation
  - How much systems engineering should be taught in the model curriculum?
  - How do we get around the continuation of the belief in the systems engineering community that software acquisition can be treated in the same way as one acquires hardware?

15 August 2007 — Integrated Software and Systems curriculum (iSSEc) Project Workshop — 5

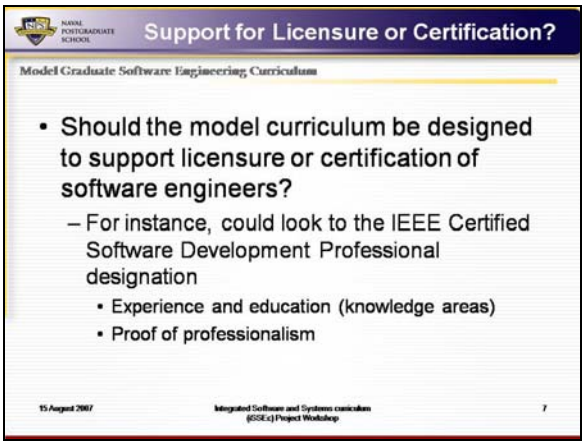


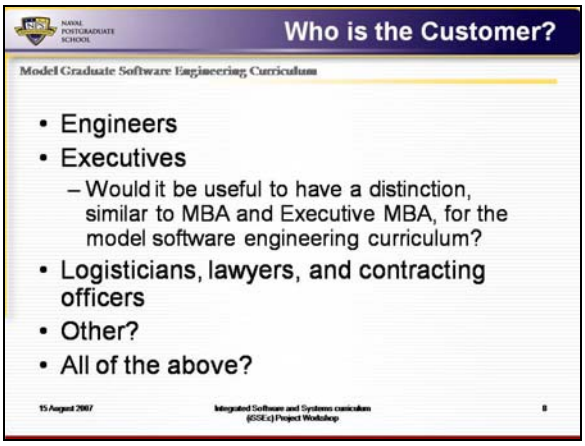
Thesis, Capstone Project, or Practicum?

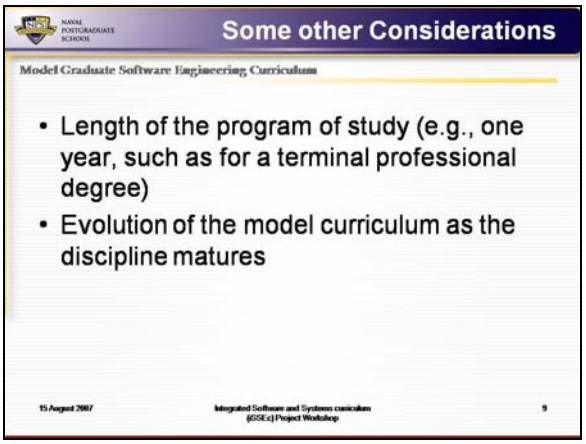
Model Graduate Software Engineering Curriculum

- Should the model curriculum have a requirement for the completion of a thesis, capstone project, or practicum?
  - Each alternative has its pros and cons

15 August 2007 — Integrated Software and Systems curriculum (iSSEc) Project Workshop — 6


Support for Licensure or Certification?
Model Graduate Software Engineering Curriculum
• Should the model curriculum be designed to support licensure or certification of software engineers?
– For instance, could look to the IEEE Certified Software Development Professional designation
• Experience and education (knowledge areas)
• Proof of professionalism
15 August 2007
Integrated Software and Systems curriculum (iSSEc) Project Workshop
7


Who is the Customer?
Model Graduate Software Engineering Curriculum
• Engineers
• Executives
– Would it be useful to have a distinction, similar to MBA and Executive MBA, for the model software engineering curriculum?
• Logisticians, lawyers, and contracting officers
• Other?
• All of the above?
15 August 2007
Integrated Software and Systems curriculum (iSSEc) Project Workshop
8


Some other Considerations
Model Graduate Software Engineering Curriculum
• Length of the program of study (e.g., one year, such as for a terminal professional degree)
• Evolution of the model curriculum as the discipline matures
15 August 2007
Integrated Software and Systems curriculum (iSSEc) Project Workshop
9

Appendix D: Homework Assignments

D-1: David Weiss

1. What should recipient know?

1.1. Basic mathematics needed to understand techniques for and limitations in constructing and verifying and validating software and system architectures and code,

1.2. Limitations prescribed by computational theory.

1.3. Underlying principles of software engineering needed to construct well-engineered systems, e.g., meaning of structure, information hiding, object orientation, definition of interface, role of specification, i.e., what distinguishes software engineering from simply writing code.

1.4. Meaning of engineering and what constitutes good engineering practice. A few examples from other industries.

1.5. Architecture and its role.

1.6. Architecture, algorithms, and a few tricks of the trade in at least one domain, such as telecommunications, finance, medicine, control systems.

1.7. Basic theory and use of operating systems, databases, language parsers, performance analysis, human-computer interfaces (some of this should be a precondition).

1.8. Basics of project planning and types of development processes, e.g., waterfall, iterative, product-line.

1.9. Principles underlying the specification and design of a product line.

1.10. Principles underlying configuration and change control systems.

1.11. Problems of scale, i.e., what problems dominate as systems become larger.

1.12. Basic theory of teamwork and how it is manifested in software development teams, what different manifestations are, and what their strengths and weaknesses are, e.g., distributed development, open source development, distributed (including agile) development, Conway's Law.

1.13. Theory of software measurement, e.g., goal-question-metric paradigm, CMMI scale, complexity measures.

1.14. Strunk & White and why it's important to apply it.

1.15. Ethical guidelines for engineering in general and software engineers in particular.

1.16. Economic principles underlying cost and value analyses for software systems, and return on investment calculations.

1.17. Finance and marketing principles applied to software engineering, e.g., the quality-interval-cost cycle.

2. What should recipient be able to do?

2.1. Write a program, in at least two different languages, and give a convincing argument that it is correct.

2.2. Review an architecture. Review a requirements specification.

2.3. Operate as a member of a software development team in at least two different roles, e.g., developer, system tester, including knowing the process for making a change to the code to add a new feature or to correct an error.

2.4. Construct a project plan.

2.5. Check code into and out of a configuration control system.

2.6. Conduct a performance analysis of a small system.

2.7. Conduct code inspections.

2.8. Write a good interface specification. Review an interface specification.

2.9. Estimate how long it will take to implement a module.

2.10. Use a software development environment such as Eclipse.

2.11. Distinguish between well-founded research and poorly conceived or executed research, particularly be able to read journal and books relevant to software engineering and understand what's useful and what's not.

2.12. Be able to cite at least two examples of good software architectures and two examples of failed software architectures and explain why they succeeded or failed.

2.13. Review both cost and value analyses of software systems.

2.14. Review the design of a graphical user interface.

2.15. Design a small product line.

D-2: Barry Boehm

1. **What should recipient know?**
    a. Software ergonomics beyond user interface design: task analysis, work context analysis, computer supported cooperative work, operational concept formulation.
    b. Software/systems quality attributes and attribute tradeoff analysis
    c. Software risk analysis and risk mitigation strategies (buying information, risk avoidance, reduction, transfer, or assumption)
    d. COTS/NDI (non-developmental item) assessment, tailoring, and integration
    e. Legal aspects of software: intellectual property, contracting, governance
    f. Presentation and communication skills and variation by audience
2. **What should recipient be able to do?**
    a. Use prototypes, scenarios, personas, etc. to engineer a non-programmer, collaborative-user interface and operational concept
    b. Conduct a tradeoff analysis between performance and another quality attribute, e.g., accuracy, security, availability
    c. Perform a risk analysis of a given software project plan
    d. Perform a COTS/NDI selection analysis
    e. Assess the relative vulnerabilities of the supplier and consumer for a shrink-wrapped software product's terms of use
    f. Tailor a project proposal briefing to a prospective financial sponsor and to an end-user representative

D-3: Larry Bernstein

Some questions that will define directions:

1. What programming skill level and experience is expected of a software engineer?
2. Should the curricula be established for the minimum understanding a software engineer needs before being consider a software engineer or should it be targeted to an 'average' professional in the field?
3. Is there a mandatory reading list of a few essential classic software engineering materials? Do you agree that every software engineer can be expected to have read or know:
   - Brooks Mythical Man Month
   - Parnas on Modularization
   - Boehm on Spiral Model
   - Read Dijkstra on Goto less programming
   - Agile Manifesto
   - IEEE/ACM Ethics- short form
   - Linger/Mills on Structured programming
   - McIlroy 1968 NATO paper
   - ACM video showing the 1957 Leo development
   - Read the GAO report on the failure of the Patriot in Desert Storm.
4. How will the reference curricula be kept current?
5. Should the curricula cover people intensive techniques such as performance reviews, innovation, team building and guru management?
6. How commonly does a MS program require a thesis? How commonly does it optionally allow a thesis?
7. How commonly does a MS program require domain-specific application of SwE concepts, such as requiring a course on "software in the telecommunications domain"? How commonly does it offer such courses as an elective?

D-4: Thomas Hilburn

Some questions relevant to establishing requirements:

1. What kind of prior background and preparation for admission to the degree program is expected? Undergraduate degree in computing or any engineering degree or and any B.S./B.A.? Experience in Software engineering practice?
2. What will be the general nature of the degree? Will it be a "Master of SwE" professional degree or a "MS in SwE" research based/preparation degree?
3. How will the graduate degree relate to or overlap with undergraduate degrees in Software Engineering? Computer Engineering?
4. What is the market of such degrees?

The following is an initial set of high-level requirements (graduate outcomes) for a Model Graduate Curriculum in Software Engineering:

1. Show mastery of the software engineering knowledge and skills, and professional issues necessary to practice as a software engineer. This mastery will include
    a. At least comprehensive level competency across the ten SWEBOK knowledge areas (not including the KA on "Knowledge Areas of the Related Disciplines").
    b. Application level competency, or above, in six of the ten SWEBOK knowledge areas.
    c. At least comprehensive level competency of "System engineering Fundamentals" as described in the Guide to Systems Engineering Body of Knowledge (G2SEBoK - http://g2sebok.incose.org/) .
2. Work effectively as part of a team to develop and deliver quality software artifacts.
3. Reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.
4. Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.
5. Demonstrate an understanding and appreciation for the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.
6. Possess the ability to learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.

D-5: Osmo Vikman

Osmo's requirements for an ideal software curriculum:

1. It should not be constrained within a specific implementation technology "silo" (i.e. software, electronics or mechanicalengineering/design)
2. It should emphasize systems thinking, innovation and problem solving skills and competences
3. Architectural thinking should be one of the core competences (i.e. service, system, product, platform, software, hardware,...architectures)
4. It should develop collaboration capabilities for extended enterprise (i.e. spanning the whole value chain/network of a business domain)
5. It should introduce stakeholder need analysis methods and tools for the whole lifecycle of a service, system or product
6. It should teach Product Family (i.e. Product Line) approach instead of development of one-off products
7. It should teach complex system-of-systems development based on network-centric approach
8. Change management and change tolerance (resilience) skills will be essential in facing the ever more uncertain future

Appendix E: Strawman Curriculum Outline

Size estimate 25-50 pp

Section	**Page count**
Executive Summary	1
Introduction	3
(Addressable markets, spectrum of degrees, project rationale)	
Student capabilities	
- Masters program entrance requirements *(expected knowledge and skills)*	2
- Masters graduate Capabilities *(expected knowledge and skills)*	5
Curriculum	
- Requirements *(for curriculum "-ilities")*	1
- Body of Knowledge *(Deltas + and - to SWEBOK)*	10
- Curriculum Architecture *(to meet requirements)*	3
- Course Packagings *(order, content, texts, readings, radical packaging)*	10
(alternative packagings, such as integrative vs. discrete)	
- Summary of As-Is analysis	2
Discussion of Teaching Methods *(philosophical?)*	2
Curriculum implementation (*guidance)*	2
Appendices *(as necessary)*	
- Rationale *(on detailed decisions)*	
- Program support *(faculty, infrastructure, scope – evolution & growth)*	
- Mappings of requirements to courses, etc.	
References/Bibliography	