

iSSEc

Integrated Software & Systems Engineering Curriculum (iSSEc) project



**Graduate Software Engineering 2009 (GSWE2009)
Companion Document**

Frequently Asked Questions on Implementing GSWE2009

Version 1.0

November 10, 2009

Unlimited Global Distribution

Stevens Institute of Technology[©] 2009

This page intentionally left blank.

Table of Contents

Preface	v
Acknowledgments	vi
Executive Summary of GSWE2009.....	vii
1. PLANNING	1
1.1. What are the first steps in planning a new program?	1
1.2. How can industrial champions be found? (Same as 4.3).....	1
1.3. How can government champions be found?	2
1.4. How can a new program identify its student market and project student enrollments? (Same as 4.1).....	2
1.5. How can a program broaden its external support?	2
1.6. How can current and future employers of program graduates be identified? (Same as 4.4).....	3
1.7. How can a program communicate with its potential student market? (Same as 4.2)	3
1.8. How can a program find start-up funding?	3
1.9. What are the options for starting up a new academic program?.....	4
1.10. What are the factors influencing where to locate the program within the university?	4
1.11. How can a new program get needed faculty resources? (Same as 3.1).....	5
2. INTERNAL COMMUNICATION.....	7
2.1. Who are the internal stakeholders of a graduate software engineering program and what are their concerns?	7
2.2. How can potential stakeholders be contacted?.....	8
2.3. How can the existing or proposed program be marketed to the internal stakeholders?	8
3. ACQUIRING RESOURCES	11
3.1. How can a new program get needed faculty resources? (Same as 1.11).....	11
3.2. How can salary differences be handled for industry-recruited faculty?.....	11
3.3. How can a new program get needed hardware and software resources?.....	12
3.4. What are good sources of courseware?	12
3.5. What other resources will be needed?.....	12
4. EXTERNAL COMMUNICATION.....	13
4.1. How can a new program identify its student market and project student enrollments? (Same as 1.4).....	13
4.2. How can a program communicate with its potential student market? (Same as 1.7)	13
4.3. How can industrial champions be found? (Same as 1.2).....	14
4.4. How can current and future employers of program graduates be identified? (Same as 1.6).....	14
4.5. How can a program gain visibility in the local community?.....	14
5. IMPLEMENTATION/EXECUTION	15
5.1. Who should maintain control over the curriculum?	15
5.2. How should the ratio of regular faculty versus external faculty be handled?.....	15
5.3. How should external faculty be integrated into the program?.....	15
5.4. What are the pedagogical considerations for a graduate SE program?	15
5.5. How can deficiencies in student preparation be addressed?.....	16
5.6. How can students with advanced preparation be accommodated?.....	16
5.7. What are the options for providing opportunities to gain work experience for students who enter the program without sufficient or appropriate backgrounds?	16
5.8. What are the options for combined undergraduate/graduate programs?	16
5.9. What are the considerations for relying on courses offered by other academic units and programs?.....	17
5.10. How are “cross cutting” topics, such as ethics, teamwork, and oral and written communication skills to be taught?	17
5.11. What kinds of laboratory facilities are required for a graduate program in software engineering?	17
5.12. What are the options for delivery of the program?.....	17
6. PROGRAM EVOLUTION	19

6.1.	How can teaming with another university help an implementer ramp up a fledging degree program to realize the educational outcomes (i.e., for when a student graduates) identified in GSwE2009?	19
6.2.	How can an implementer gradually reduce its reliance on partnering with other universities?	19
6.3.	As a program grows, how can it sustain its quality?	20
6.4.	How can a program adapt to changes in the discipline of software engineering?	20
6.5.	How can problems in areas such as course curricula, capstone projects, and student success be addressed?..	20
6.6.	How can a program deal with changes in technology?	21
7.	Bibliography	23
7.1.	Curriculum Development	23
7.2.	History and Overview of Graduate Software Engineering Programs	24
7.3.	Establishment of New Graduate Software Engineering Programs	24
7.4.	Evolution of Graduate Software Engineering Programs	25
7.5.	Collaboration between Universities and Industry in Software Engineering Education	26
7.6.	Additional Sources of Information about Graduate Software Engineering Education	27
8.	GLOSSARY	29
8.1.	Abbreviations and Codes	29
8.2.	Terminology	32

Preface

This report contains advice on implementing a program consistent with the *Graduate Software Engineering 2009 (GSWE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering* [GSWE2009]. The advice given here has been vetted by the thirty-plus members of the Curriculum Author Team (CAT) that produced GSWE2009, all of whom have had significant experience in starting up similar programs—dating back to the well-known program at the Wang Institute in the 1980s—as well as leading the current portfolio of GSWE2009 programs.

We anticipate that the contents of this FAQ will grow as implementers provide us with feedback based on their attempt to apply the GSWE2009 guidelines. Although the guidelines and the companion documents (i.e., the FAQ and Program Comparisons) are intended to transcend geo-educational systems, the majority of the participants on the CAT are involved in graduate software engineering education within North America and the United Kingdom. We especially hope to receive feedback from implementers from other regions of the globe, with the aim of making GSWE2009 truly representative of the international milieu.

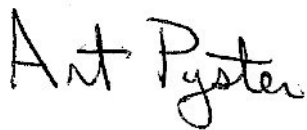
Acknowledgments

The reference curriculum, *GSwE2009*, is the product of many authors from more than 24 organizations who came together selflessly to improve global software engineering graduate education. Those authors are listed individually in the preface of *GSwE2009* along with their supporting organizations.

This report was created by a sub-team of the authors of *GSwE2009*: Mark Ardis, John Brackett, Dick Fairley, David Klappholz, Phillip Laplante, Bret Michael and Mary Shaw. Editors were Mark Ardis, Kahina Lasfer and Bret Michael. Several authors of *GSwE2009* provided helpful comments and suggestions for questions and answers.

Special thanks go to Bret Michael for leading the team and carrying more than his share of the workload. Graduate assistant Kahina Lasfer performed many enabling tasks for logistical support. Additional support was provided by Nicole Hutchison.

We are also grateful to Kristen Baldwin, Bruce Amato, Scott Lucero, and others in the U.S. Office of the Secretary of Defense for their leadership and financial support for this effort. Paramount to our success has been Ms. Baldwin's early and consistent recognition that a reference curriculum for software engineering would be of most benefit to the defense community if it were not biased toward defense applications. Finally, we thank Stevens Institute of Technology, especially the School of Systems and Enterprises and its dean, Dinesh Verma, for supporting this effort from the very beginning.

A handwritten signature in black ink that reads "Art Pyster". The signature is written in a cursive, slightly slanted style.

Art Pyster
GSwE2009 Editor and Project Leader

Executive Summary of GSwE2009

The *Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering* [GSwE2009] is a set of recommendations for a master's level graduate program in software engineering (SwE), together with implementation guidance for a university to satisfy those recommendations. Earlier versions of this work were called the *Graduate Software Engineering Reference Curriculum* (GSwERC).

The program described by GSwE2009 is for a professional master's degree, analogous in many ways to a master's of business administration. GSwE2009 is envisioned as a living document that will be revisited regularly and updated when necessary to ensure relevance to the rapidly evolving software engineering discipline. The GSwE2009 document includes the curriculum recommendations and materials describing their creation, implementation, and evolution.

GSwE2009 includes the following:

- A set of outcomes to be fulfilled by a student who successfully completes a graduate program based on the curriculum (see summary below)
- A set of student skills, knowledge, and experience assumed by the curriculum, not intended as entrance requirements for a specific program, but as the starting point for the curriculum's outcomes (see summary below)
- An architectural framework to support implementation of the curriculum
- A description of the fundamental or core skills, knowledge, and experience to be taught in the curriculum to achieve the outcomes. This is termed a *Core Body of Knowledge* (CBOK) and includes topic areas and the depth of understanding a student should achieve.

Additional materials included in the document:

- The fundamental philosophy for GSwE2009 development as described in a set of guiding principles (see summary below)
- A discussion of how GSwE2009 will evolve to remain effective
- A mapping of expected outcomes to the CBOK and to the total GSwE2009 program recommendations

- A description of Knowledge Areas (KAs) discussed in GSwE2009 that are not yet fully integrated into the current version of the Software Engineering Body of Knowledge (SWEBOK)
- Glossary, references, and other supporting material.

Summary of Guidance for Creating GSwE2009

The following guidance, established early in the development of GSwE2009, came primarily from SE2004¹ and the deliberations of the GSwE2009 authors.

- Create a set of recommendations for developing and improving curricula for master's level software engineering education.
- Target a professional degree for practicing software engineers.
- Require about as many credit hours as typical programs do now.
- Recognize software engineering as a distinct discipline with a rich body of knowledge, practice, and theory.
- Recognize that software engineering is founded on a wide variety of disciplines, with deepening ties to Systems Engineering (SE).
- Require that all software engineering students be able to integrate theory and practice.
- Foster ongoing review and revision of the curriculum because of rapid evolution in software engineering.
- Be sensitive to changes in technologies, practices, applications, pedagogy, and the importance of lifelong learning.
- Offer significant guidance in individual curriculum components through a CBOK that all students are expected to master.
- Identify fundamental skills and knowledge that all software engineering master's program graduates must possess.

¹ ACM/IEEE-CS Joint Task Force on Computer Curricula, *Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, August 2004.

- Use a flexible curriculum architecture and recognize existing bodies of knowledge, modified and enhanced as required.
- Limit common knowledge required for *all* students to no more than 50% of total knowledge taught.
- Be broadly based and international in scope.
- Include exposure to aspects of professional practice as an integral component of the graduate curriculum.
- State strategies and tactics for implementation.
- Distinguish clearly between SE2004 and GSwE2009.
- Identify expected knowledge and experience for students to enter a master's program in software engineering.

Summary of Outcomes

Graduates of a master's program that satisfies GSwE2009 recommendations will:

- Master the CBOK.
- Master software engineering in at least one application domain, such as finance, medical, transportation, or telecommunications, and one application type, such as real-time, embedded, safety-critical, or highly distributed systems. That mastery includes understanding how differences in domain and type manifest themselves in both the software itself and in its engineering, and includes understanding how to learn a new application domain or type.
- Master at least one KA or sub-area from the CBOK to at least the Bloom Synthesis level.
- Be able to make ethical professional decisions and practice ethical professional behavior.
- Understand the relationship between SwE and SE and be able to apply SE principles and practices in the engineering of software.

- Be an effective member of a team, including teams that are international and geographically distributed, effectively communicate both orally and in writing, and lead in one area of project development, such as project management, requirements analysis, architecture, construction, or quality assurance.
- Be able to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.
- Understand and appreciate feasibility analysis, negotiation, and good communications with stakeholders in a typical software development environment, and be able to perform those tasks well; have effective work habits and be a leader.
- Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.
- Be able to analyze a current significant software technology, articulate its strengths and weaknesses, compare it to alternative technologies, and specify and promote improvements or extensions to that technology.

Summary of Expected Background

GSwE2009 presumes that an entering student has:

- The equivalent of an undergraduate degree in computing or an undergraduate degree in an engineering or scientific field and a minor in computing,
- The equivalent of an introductory course in software engineering, and
- At least two years of practical experience in some aspect of software engineering or software development.

These presumptions about entering students are designed to achieve the 10 outcomes previously described. However, it is recognized that individual schools may start with a student population that has characteristics that are different from what GSwE2009 presumes here. Such schools will likely have to lengthen their master's programs in order for their students to achieve all 10 outcomes—or the schools will deliberately choose not to adopt some outcomes. In fact, schools may even add other outcomes to favor their particular markets and institutional emphases. GSwE2009 is not intended for use to

certify or accredit either programs or individuals. It is a set of recommendations that must be tailored by each adopting university.

The process of developing this report has been highly inclusive. It has been widely reviewed by academics and practitioners through a public draft process. We have also held feedback sessions at conferences and meetings, including the annual American Society for Engineering Education (ASEE) meeting, the International Symposium of the International Council on Systems Engineering (INCOSE), the International Conference on Software Engineering (ICSE), the Conference on Software Engineering Education and Training (CSEET), and various smaller meetings in Europe, Asia, and various parts of the United States. These meetings have provided us with critically important feedback that we have used to shape the final report.

From the beginning it was intended for GSwE2009 to be a living document, with a broad, responsible, and knowledgeable community of practice. It was anticipated that after Version 1.0 was published, Stevens Institute of Technology, which has managed the original development, would identify a steward who would assume responsibility for maintaining and refining the model and expanding and focusing implementation guidance based on experience and feedback from the supporting community and academia, industry, and students. Effort is now underway for a combination of the ACM and the IEEE Computer Society to become that steward. As of the writing of this document, discussions are underway for those two organizations to take over maintenance responsibility for GSwE2009 within the first 6 months of the release of Version 1.0, with INCOSE playing a supporting role.

To support and enable wide acceptance of GSwE2009, two companion documents - *Comparisons of GSwE2009 to Current Master's Programs in Software Engineering* and *Frequently Asked Questions on Implementing GSwE2009* [this document] – are being prepared concurrently with the release of GSwE2009. They will be available in Fall 2009 at www.GSwE2009.org and updated regularly. These are not under ACM or IEEE-CS stewardship but, rather, are maintained by Stevens Institute.

This page intentionally left blank.

1. PLANNING

The planning process will differ significantly when starting a new graduate software engineering program, as compared to modifying an existing one to incorporate the major aspects of the GSWE2009 curriculum guidelines. If there is significant external support for the potential program, a business plan and an implementation plan should be created in order to obtain university support for launching the program. This section includes issues that arise during the planning stages for a new program. Other advice for starting a new program may be found in the papers cited in Section 7.3 of the Bibliography.

1.1. What are the first steps in planning a new program?

ANSWER: We assume that one or more champions for a new graduate software engineering program exist. They are essential to developing a new program, and they could be in industry, government, or within the university. A university champion might be a dean or a senior faculty member who strongly supports the creation of a graduate software engineering program. An industrial champion should have the credibility to influence technical leaders in other companies to support the formation of the new graduate program. Ideally an industrial or government champion would also lead a campaign to provide start-up financial support for the program, but they might only promise to encourage their employees to apply to the program. For example, Boston University received strong support from Digital Equipment Corporation and Hewlett-Packard Corporation, both of whom wanted a local software engineering program for their employees. Southern Methodist University received support in the form of students and adjunct faculty from local industry.

In some cases a consortium of companies might act as champions. In the late 1980s a consortium of 9 companies in Florida sponsored a new program at Florida Atlantic University that used materials from the Video Dissemination Project of the Software Engineering Institute [Florida Atlantic University 1994]. As another example, the Portuguese government along with regional, national, and international companies championed the creation of a joint master's program in Software engineering between Carnegie-Mellon University and the University of Coimbra. All of the champions provided significant monetary and in-kind support. The industry members have permitted seasoned engineers and managers to teach courses and mentor students in the software engineering program, in addition to financially sponsoring individuals and cohorts of students.

1.2. How can industrial champions be found? (Same as 4.3)

ANSWER: One of the best places to look for champions is the institutional alumni list. While they may not be alumni of the new software engineering program, they are often

interested in helping establish new programs and in improving current offerings at the school. Another place to turn is the base of employers of current graduates of similar programs, such as computer science and computer engineering. A third place to look is the software engineering literature in order to identify technical leaders in the geographical area from which the university typically recruits graduate students. Other examples of industry-academia relationships may be found in section 7.5 of the Bibliography.

1.3. How can government champions be found?

ANSWER: Champions can be found at the local and national levels of government. Champions can be found in agencies involved in: implementing education policy; creating a source of new hires; and keeping the people in technical career paths moving forward in the human-capital pipeline, that is, advancing them through the ranks from apprentices to masters of their disciplines.

1.4. How can a new program identify its student market and project student enrollments? (Same as 4.1)

ANSWER: The recommended way to identify potential students is by conducting a market survey of local companies and government establishments currently employing software engineering personnel. Most schools have a marketing organization as part of, or supporting their admissions office. They can usually help to develop an initial market study.

The survey goal would be to estimate: (a) how many students would be expected to apply in the first year of the program, (b) how many students would be expected to attend such a program in its steady state, (c) which employers would be most likely to encourage their employees to apply to the program, and (d) where potential students currently apply to graduate school. This last piece of information can help to identify competing programs. Keep in mind that some of the students may currently attend other programs in the institution.

1.5. How can a program broaden its external support?

ANSWER: If there appears to be adequate demand for future graduates of the program then the champions can seek to broaden the base of external support for creating the program. One possible approach is to create an industry advisory board of key senior technical employees and managers from companies whose business requires software engineering professionals. Having an industrial champion will greatly facilitate formation of the advisory board.

1.6. How can current and future employers of program graduates be identified? (Same as 4.4)

ANSWER: Local employers will often already be hiring graduates of similar programs, such as computer science and computer engineering. One place to look for employers is at meetings of local high-tech professional organizations. For example, the IEEE holds regional meetings regularly for members, who are usually working professionals. Some of the attendees may be potential students, and others may be potential employers of graduates. Some regions have high-tech trade organizations whose meetings can serve a similar purpose. Many schools hold career fairs where company representatives are available for informal conversations. This is a good way to find out what skills and knowledge they look for in new graduates.

1.7. How can a program communicate with its potential student market? (Same as 4.2)

ANSWER: It is not always true that “if you build it they will come.” Communicating with potential students is important to the success of any program in the initial years. A market study will reveal where applicants may be drawn. It is helpful to establish a relationship with the Human Resources (HR) departments of local high-tech employers. HR departments administer company benefits, such as tuition reimbursement, and they often have lists of schools for their employees to use in considering professional development. An industrial champion should be able to arrange in-house presentations to potential students. Large companies sometimes have “benefits fairs” where vendors, including academic programs, present material or set up booths.

Another avenue to consider is local professional organizations. Trade organizations provide networking for local professionals and often have social events sponsored by local companies. There often are opportunities to give a short presentation or set up a booth at some of these meetings. Better yet, a school may host one of these on campus. The computing departments at Rochester Institute of Technology host an annual meeting of their local high tech organization where the departments show off their degree programs and take advantage of networking opportunities. Professional societies, like the IEEE Computer Society, also have local organizations. These groups welcome talks by faculty on technical subjects at their meetings.

1.8. How can a program find start-up funding?

ANSWER: Assuming there is industrial support for the potential program, a business case and an implementation plan should be created in order to obtain university support for launching the program. A new program needs seed funding to launch itself, and its business plan should address this issue. The level of start-up funding needed may be high over the first few years, perhaps too high for a university to self-fund. In addition, continued funding is also needed to sustain recruitment of students, along with excellence

in teaching, research and continued professional growth of the faculty. A graduate software engineering program may be able to share some capital resources with computer science and other disciplines, so there may not be as many start-up costs as there might be for a stand-alone program.

The university might obtain financial and in-kind sponsorship from government agencies and industry. For example, an organization can encourage its employees to continue their professional development by (a) offering to pay the tuition to attend a graduate software engineering program, (b) giving employees a sabbatical (i.e., partial or full salary while attending graduate school) or (c) giving employees time off from work to attend classes. An organization might also donate equipment or software to the university for instructional purposes. The university should create a close working relationship with the sponsoring government or industrial organizations, with the aim of obtaining long-term financial and in-kind sponsorship.

1.9. What are the options for starting up a new academic program?

ANSWER: One way to start is by gradual build-up. Some schools have started by offering software engineering courses as electives in their computer science programs. Those courses can also be packaged in a certificate program to attract attendees from local industry who might not want to commit to an entire graduate program. Another alternative is the Big Bang approach, launching the whole program at once. Boston University [Boston University 1988] and Southern Methodist University [Southern Methodist University 1994] each did that successfully with strong support from their local industries. The risk of that approach, of course, is that a lot of energy and resources will be committed before receiving any feedback from the market.

1.10. What are the factors influencing where to locate the program within the university?

ANSWER: When a new graduate software engineering program is to be established within a university, a key consideration in evaluating alternative locations is to assess where a terminal degree graduate program is likely to thrive. The academic unit should value professional graduate education. One of the reasons hypothesized for why there are so few graduate software engineering programs at prestigious research universities is that most science and engineering academic units do not value terminal degree graduate education. However, the business schools at these universities do focus on terminal degrees.

In an academic unit that values professional education, there are likely to be senior faculty with applied experience in the field who do not have a doctorate or tenure. For example, in a business school it is not unusual to find a former corporate chief executive officer (CEO) as a full-time faculty member. The recruitment process for a senior

software engineering professional in a graduate software engineering program is more similar to recruiting a CEO for a business school than it is to recruiting a tenure-track faculty member. A graduate software engineering program will likely be handicapped in recruiting if it cannot hire non-tenure full-time senior faculty, some of whom do not have doctorates.

1.11. How can a new program get needed faculty resources? (Same as 3.1)

ANSWER: There are two common sources of faculty for software engineering programs: (a) faculty from related areas that have knowledge and interest in software engineering, and (b) software engineers from industry who are interested in teaching. The former are often working in computer science academic units, but they may be found in almost any discipline that uses computing. Although they may have good teaching skills, they may need some help adjusting to the professional nature of the program. Some of their students will have considerable software development experience and expect to learn about the latest methods and tools. Staying current in the field is important. Consulting is one good way to do this.

The second type of faculty candidate (from industry) may need some help making the transition to teaching. If they work part-time as adjunct faculty they will need to balance the demands of two jobs. If they become full-time faculty there may be some discomfort in taking a salary cut. In either case it is important to remind them of the benefits of an academic position.

Both types of faculty may need help using new teaching technologies, especially if you have a distance program. Hopefully the school has support for learning and mastering this style of teaching. At the Naval Postgraduate School, for example, faculty members are required to complete a three-month intensive course on how to teach via distance techniques. The university also provides ongoing distance-learning support to faculty members through its Center for Educational Design, Development, and Distribution.

It is prudent to ramp up faculty at a pace consistent with the growth of the program. This means that some part-time faculty will be needed early on before there is enough demand to justify hiring full-time faculty. External faculty from industry are often used for this, but do not forget to consider faculty from other academic units at your institution.

This page intentionally left blank.

2. INTERNAL COMMUNICATION

Internal stakeholders are those who will affect or be affected by a GSwE2009-based graduate program in software engineering; they include students, faculty members, administrators, and facilities providers. Issues related to establishing and maintaining communication among these internal stakeholders are discussed in this section.

2.1. Who are the internal stakeholders of a graduate software engineering program and what are their concerns?

ANSWER: Internal stakeholders are those who will affect or be affected by a graduate software engineering program; they include students, faculty members, administrators, and facilities providers. Students enrolled in the program will have the expectation that the program will advance their careers by providing practical, industry-current education. Students enrolled in other programs of study may want to take supporting courses or perhaps a minor supporting program in software engineering. These students will be concerned that they have the necessary background for the software engineering classes they want to take. New graduate students who are entering the university may be considering a major program of study in software engineering. They, too, will be concerned about having the necessary backgrounds and the career opportunities provided by graduate education in software engineering.

Faculty members may be directly affiliated with the program or may participate in the program while primarily associated with other programs and academic units. These faculty members will be concerned with issues of starting and maintaining a high quality graduate program in software engineering. Some faculty members in other programs may be concerned that the program will siphon students and resources from their programs of study into software engineering. Other faculty members may fail to see the importance or relevance of a graduate software engineering program.

Administrators will establish and enforce the policies under which a graduate software engineering program will be conducted. They will allocate the necessary resources (faculty positions, laboratory facilities, classrooms, etc), during the start-up period for the program. Their concerns will be the compatibility of a graduate software engineering program with their organizational charters, the relative priority of software engineering with respect to the other programs they administer, the importance and influence of the external clients to be served (e.g., local industry), and the prospects for long-term viability of the program.

Facilities providers are those who will provide the hardware, software, classroom technology and other computing resources needed to start and maintain a graduate program in software engineering. They will also provide on-going support for students

and faculty members. Their concerns will be adequacy of budget and personnel to fulfill their mission.

2.2. How can potential stakeholders be contacted?

ANSWER: The usual means of institutional communications can be used; for example, campus newspapers and newsletters, electronic and physical bulletin boards, and email lists. In addition, special interest groups such as student chapters of professional societies provide opportunities to contact potential students and faculty participants in a graduate software engineering program.

Most importantly, an internal champion is needed; i.e., a person who has the enthusiasm and personality to be an evangelist for the program. This individual should be a respected faculty member or administrator who can gain audiences with and effectively communicate with students, potential students, faculty members, administrators, facilities providers, and other stakeholders. The champion should work on a continuing basis to build and maintain internal support for the program. If a program is to flourish it needs commitment from faculty or administrators who believe in it and have a vision for its future.

2.3. How can the existing or proposed program be marketed to the internal stakeholders?

ANSWER: On-going effort should be expended to inform students and potential students of the relevance of the program of study for their careers and the career opportunities a graduate degree in software engineering will afford them. They will be interested to learn of the kinds of projects, course assignments, and software tools to which they will be exposed. Students will be attracted to the opportunities for collaborative interactions with other students and faculty members, both inside and outside the classroom. The most effective marketing will occur by word of mouth from enthusiastic students to other students and potential students.

Faculty members from other academic units and programs of study may be interested in opportunities that do not exist within their programs to engage with students and faculty members who have interest in, and experiences with, current and advanced methods, tools, and techniques of software engineering.

Administrators may be interested in the current and potential student population for the program, the interest and support of local industry, compatibility of a graduate program in software engineering with their organizational charters, increased status and expanded realm of their organizational units, and a financially viable business plan. They should receive on-going status reports of enrollments, finances, achievements, and plans for

future developments in order to provide visibility of the program to them, and to maintain their interest in it.

Facilities personnel should welcome the increased responsibilities and scope of work, contingent on clear guidelines and assurances that accompanying resources will be provided. They should also welcome the opportunity to learn and support the new and advanced systems and software tools used by a graduate program in software engineering.

This page intentionally left blank.

3. ACQUIRING RESOURCES

This section discusses how to acquire resources for a program. One of the most important tasks to accomplish in starting a new program is to find the right faculty to teach the courses. This is especially difficult in professional software engineering programs, where the best teachers have both academic and industrial experience. Other resources, hardware and software are usually easy to obtain through industrial contacts.

3.1. How can a new program get needed faculty resources? (Same as 1.11)

ANSWER: There are two common sources of faculty for software engineering programs: (a) faculty from related areas that have knowledge and interest in software engineering, and (b) software engineers from industry who are interested in teaching. The former are often working in computer science academic units, but they may be found in almost any discipline that uses computing. Although they may have good teaching skills, they may need some help adjusting to the professional nature of the program. Some of their students will have considerable software development experience and expect to learn about the latest methods and tools. Staying current in the field is important. Consulting is one good way to do this.

The second type of faculty candidate (from industry) may need some help making the transition to teaching. If they work part-time as adjunct faculty they will need to balance the demands of two jobs. If they become full-time faculty there may be some discomfort in taking a salary cut. In either case it is important to remind them of the benefits of an academic position.

Both types of faculty may need help using new teaching technologies, especially if you have a distance program. Hopefully the school has support for learning and mastering this style of teaching. At the Naval Postgraduate School, for example, faculty members are required to complete a three-month intensive course on how to teach via distance techniques. The university also provides ongoing distance-learning support to faculty members through its Center for Educational Design, Development, and Distribution.

It is prudent to ramp up faculty at a pace consistent with the growth of the program. This means that some part-time faculty will be needed early on before there is enough demand to justify hiring full-time faculty. External faculty from industry are often used for this, but do not forget to consider faculty from other academic units at your institution.

3.2. How can salary differences be handled for industry-recruited faculty?

ANSWER: One way to handle this is through corporate or other donations to partly offset their salary. Some companies have been known to partially underwrite academic

positions for their early retirees. While not fitting the conventional definition of a “chair,” naming rights to such positions might attract industrial support.

Another way is to point out advantages not available to these faculty candidates in their current positions, such as tuition breaks for their families, low-cost access to university events, summers off, sabbaticals, or other perks.

A third method is to allow faculty to perform technical consulting on a part-time basis in areas that keep them active in their field. Consulting income can help close the pay gap. The experience gained from consulting is usually relevant to the courses they are teaching, and it helps make faculty more effective in the classroom.

3.3. How can a new program get needed hardware and software resources?

ANSWER: The computing industry has provided many academic units with donations of equipment for laboratories and classrooms. Some programs offer free equipment, while others offer steep discounts. Fortunately, software engineering is not particularly hardware-intensive, and most students have their own computers.

Large companies like IBM and Microsoft have academic alliance programs that provide free software for academic use. Be sure to keep open source solutions in mind, also. Open source software can offer benefits both in application and in study: availability of the source code and supporting documents makes it a good source of examples and exercises in several software engineering areas.

3.4. What are good sources of courseware?

ANSWER: One good source of software engineering courseware is SWENET, a web-based community for software engineering education <<http://www.swenet.org>>. SWENET has sample lecture materials, exercises and guidelines for assessing student performance on several software engineering topics. Of course, many publishers of software engineering textbooks have websites with lecture notes and exercises for faculty who adopt their books. These can be especially helpful to faculty teaching a course for the first time.

3.5. What other resources will be needed?

ANSWER: It is very helpful for a program to have its own space on campus, even if it is part of another academic unit. Students appreciate having a place to gather and commune with their peers. A small conference room is enough to get started.

4. EXTERNAL COMMUNICATION

It is important for a program to identify its potential market of students and keep in touch with them as the program evolves. This does not require a complicated marketing strategy, but it does require attention each year. The other important stakeholders to identify are the potential employers of graduates of your program. Methods for identifying and communicating with external stakeholders are discussed in this section.

4.1. How can a new program identify its student market and project student enrollments? (Same as 1.4)

ANSWER: The recommended way to identify potential students is by conducting a market survey of local companies and government establishments currently employing software engineering personnel. Most schools have a marketing organization as part of, or supporting their admissions office. They can usually help to develop an initial market study.

The survey goal would be to estimate: (a) how many students would be expected to apply in the first year of the program, (b) how many students would be expected to attend such a program in its steady state, (c) which employers would be most likely to encourage their employees to apply to the program, and (d) where potential students currently apply to graduate school. This last piece of information can help to identify competing programs. Keep in mind that some of the students may currently attend other programs in the institution.

4.2. How can a program communicate with its potential student market? (Same as 1.7)

ANSWER: It is not always true that “if you build it they will come.” Communicating with potential students is important to the success of any program in the initial years. A market study will reveal where applicants may be drawn. It is helpful to establish a relationship with the Human Resources (HR) departments of local high-tech employers. HR departments administer company benefits, such as tuition reimbursement, and they often have lists of schools for their employees to use in considering professional development. An industrial champion should be able to arrange in-house presentations to potential students. Large companies sometimes have “benefits fairs” where vendors, including academic programs, present material or set up booths.

Another avenue to consider is local professional organizations. Trade organizations provide networking for local professionals and often have social events sponsored by local companies. There often are opportunities to give a short presentation or set up a booth at some of these meetings. Better yet, a school may host one of these on campus. The computing departments at Rochester Institute of Technology host an annual meeting

of their local high tech organization where the departments show off their degree programs and take advantage of networking opportunities. Professional societies, like the IEEE Computer Society, also have local organizations. These groups welcome talks by faculty on technical subjects at their meetings.

4.3. How can industrial champions be found? (Same as 1.2)

ANSWER: One of the best places to look for champions is the institutional alumni list. While they may not be alumni of the new software engineering program, they are often interested in helping establish new programs and in improving current offerings at the school. Another place to turn is the base of employers of current graduates of similar programs, such as computer science and computer engineering. A third place to look is the software engineering literature in order to identify technical leaders in the geographical area from which the university typically recruits graduate students. Other examples of industry-academia relationships may be found in section 7.5 of the Bibliography.

4.4. How can current and future employers of program graduates be identified? (Same as 1.6)

ANSWER: Local employers will often already be hiring graduates of similar programs, such as computer science and computer engineering. One place to look for employers is at meetings of local high-tech professional organizations. For example, the IEEE holds regional meetings regularly for members, who are usually working professionals. Some of the attendees may be potential students, and others may be potential employers of graduates. Some regions have high-tech trade organizations whose meetings can serve a similar purpose. Many schools hold career fairs where company representatives are available for informal conversations. This is a good way to find out what skills and knowledge they look for in new graduates.

4.5. How can a program gain visibility in the local community?

ANSWER: Another form of visibility that can help a new program may come from events that are sponsored by the unit that administers the program. Wang Institute held short courses in the summer that attracted potential students to campus. Butler University conducts a colloquium series for local industry that brings invited speakers to their campus. These types of events reach both potential students and their employers. They may also encourage alumni to stay in touch with the program.

5. IMPLEMENTATION/EXECUTION

This section addresses issues of importance for the faculty members, students, and facilities providers who will be involved in implementing and executing a GSWE2009-based graduate program in software engineering.

5.1. Who should maintain control over the curriculum?

ANSWER: Faculty members who are regular employees of the school should control the curriculum (i.e., not external faculty members, such as visiting faculty and adjuncts). These regularly appointed faculty members should have the authority for the creation, delivery, evaluation and modification of the curriculum. They are responsible for the consistency and quality of its courses.

5.2. How should the ratio of regular faculty versus external faculty be handled?

ANSWER: There should be a core of regularly appointed faculty members who control the curriculum, and each course should have a course director, a regularly appointed faculty member, who monitors course deliveries by external faculty. Under these conditions, there is no “magic quota” for the ratios of regular faculty versus external faculty, provided course outcomes are achieved and the course outcomes support the program outcomes.

5.3. How should external faculty be integrated into the program?

ANSWER: External faculty members (industry visitors and adjuncts) should be treated as first class members of the faculty. They should be invited to social functions, meetings of the academic unit, and committee meetings in which they have an interest. They should be consulted on issues such as course outcomes, textbooks, and delivery mechanisms for the courses they teach, but they should not have authority to change these or other factors related to their courses without approval of the course director, who should be a regularly appointed faculty member.

A regular faculty course director should mentor newly appointed external faculty. External faculty members who have no teaching experience may benefit from attending a teaching seminar offered by the school. They may also want to attend classes taught by other faculty in the program.

5.4. What are the pedagogical considerations for a graduate SE program?

ANSWER: Pedagogical considerations should be tailored to the outcomes and objective of the program and to the preparation of students upon entry. Adult learners grow impatient with traditional lecture formats; they are better served by discussions and exercises that build on and extend their experiences.

5.5. How can deficiencies in student preparation be addressed?

ANSWER: There are two kinds of deficiencies to be addressed: knowledge deficiencies and experience deficiencies. Knowledge deficiencies can be addressed by leveling courses; such as an overview course on software and systems engineering, or a survey course in current topics in software engineering. Experience deficiencies can be partially overcome by internships in industry and assistantships within the school. Special team projects in various aspects of industrial practice can be offered for cohorts of students who lack sufficient experience (e.g., a project course on the use of software tools for software development and maintenance; or a project course on procurement, integration, and testing of open source software packages).

5.6. How can students with advanced preparation be accommodated?

ANSWER: Courses for which students have sufficient knowledge can be replaced with: (a) advanced courses in software engineering; (b) courses in application domains such as aerospace computing, information security, or cloud computing; or (c) supplementary courses in other academic units. For example, math or physics courses might be useful for students interested in scientific computing, and courses in human factors could be useful for students who are interested in visual computing.

Students with significant work experience might be permitted to replace a team project with a directed study course or a thesis as their capstone requirement.

5.7. What are the options for providing opportunities to gain work experience for students who enter the program without sufficient or appropriate backgrounds?

ANSWER: Experience deficiencies can be partially overcome by internships in industry and assistantships within the school. Special team projects in various aspects of industrial practice can be offered for cohorts of students who lack sufficient experience (e.g., a project course on use of software tools for software development and maintenance, or a project course on procurement, integration, and testing of open source software packages).

5.8. What are the options for combined undergraduate/graduate programs?

ANSWER: Given the current state of software engineering education a student with a bachelor's degree in software engineering might not receive maximum benefit from a GSwE2009-based graduate program. However, given the broad applicability of software engineering, a student might combine an undergraduate program in any number of disciplines (electrical engineering, biology, mathematics, visual arts, systems engineering) with a graduate program in software engineering. Combined programs can be designed to reduce the number of courses for a combined program to be less than the

number of courses for separate programs, thereby shortening the time required to obtain both degrees.

Alternatively, a student with an undergraduate degree in software engineering might be required to take more advanced courses in the graduate program. Embry Riddle University waives up to three core courses and requires the students who enter under this option to do a master's thesis rather than a practicum project, based on the fact that they have done a project as part of their BS program. Further, the students must take two pairs of sequential MS elective courses, so that they go into more depth in two domains than is required of other MS students.

5.9. What are the considerations for relying on courses offered by other academic units and programs?

ANSWER: Supplemental and elective courses offered by other academic units and programs can broaden and enrich a software engineering curriculum. However, required courses should be taught by faculty members who have expertise in and an orientation toward the applied nature of the topic.

5.10. How are “cross cutting” topics, such as ethics, teamwork, and oral and written communication skills to be taught?

ANSWER: These topics can be included as outcomes in subject matter courses that emphasize topics such as requirements, design and implementation. A capstone experience is often a particularly appropriate place to assess these skills. However, rubrics should be developed and a percentage of course points should be placed on these topics lest they be neglected in pursuit of the other desired outcomes of the courses.

5.11. What kinds of laboratory facilities are required for a graduate program in software engineering?

ANSWER: Laboratory facilities for a graduate software engineering program should support the domain, or domains, of emphasis of the program. In addition to the software tools needed to support a GSwE2009-based graduate program, (i.e., requirements traceability, version control, testing, etc.) a program that emphasizes real-time system development will need workstations, software tools, and perhaps hardware benches to support the course projects and capstone project. Similarly, a program that emphasizes distributed systems development will need a dedicated network to support experimentation with various protocols and network configurations.

5.12. What are the options for delivery of the program?

ANSWER: Traditional on-campus courses are still a popular delivery method, but many schools offer alternative approaches that better serve working professionals. Stevens Institute of Technology and Southern Methodist University both offer on-campus

courses, distance courses, remote delivery (classes offered on-site), and executive formats that compress instructor-student contact time to a few days or a week at a time. Distance courses require the most preparation, and they should only be attempted if there is good infrastructure and support for this format at your school.

6. PROGRAM EVOLUTION

While no degree program can remain static, even with the most stable underlying principles, software engineering is a particularly fluid discipline. Software engineering technologies, techniques and professional practices are constantly evolving, and the software industry itself is rapidly changing. Software engineering theories also evolve as new discoveries and experiences are brought to bear. Any graduate program that supports this discipline, therefore, needs to be dynamic. In this section techniques for adapting the graduate software engineering program for changes from these many sources are discussed. See also the citations on evolution of programs in Section 7.4 of the Bibliography.

6.1. How can teaming with another university help an implementer ramp up a fledging degree program to realize the educational outcomes (i.e., for when a student graduates) identified in GSWE2009?

ANSWER: One way it can help is by giving the start-up program access to the people and other necessary in-place resources for students to master the core body of knowledge of software engineering in addition to software engineering in an application domain. There are many options for partnering. For example, the University of Coimbra established a joint professional master degree program with Carnegie-Mellon University (CMU). The two schools share the teaching of the courses and conducting of the capstone team project. The University of Coimbra leverages CMU's long-established curriculum and program resources.

A less formal arrangement is simply to reuse course materials that were prepared at another school. For example, the SEI distributed course materials to schools so that they could start teaching software engineering courses on their own campuses. A short tutorial before each course and some consulting during the course were often enough to get them off the ground. Although these informal arrangements do not include some of the other benefits of a formal partnership, they provide networking opportunities for all schools involved.

6.2. How can an implementer gradually reduce its reliance on partnering with other universities?

ANSWER: Once the new degree program becomes established in its own right the start-up program can consider dropping or deemphasizing the partnering relationship with the other university. This allows the start-up program to regain autonomy in administering its own degree program and to reduce its costs. However, the implementer should also consider other factors in making such a decision, such as the value of branding. For

example, the brand name and high-quality of instruction and support provided by the partnering university may be an important selling point for recruiting students, retaining or attracting staff, and courting current and potential sponsors of the implementer's degree program.

6.3. As a program grows, how can it sustain its quality?

ANSWER: It is impossible to anticipate every kind of problem that may trouble an evolving graduate program, even a successful one. But as with any program, an ongoing monitoring system can help to short-circuit problems before they become too severe.

A vibrant (and not just a superficial, rubber-stamping) industrial advisory board can ensure that the curricula are relevant. A system of active student feedback monitoring, including course evaluations, exit examinations, and ongoing collection of data from alumni, are crucial in ensuring that the program is meeting its goals.

Furthermore, there should be a clearly defined, documented, and monitored set of course objectives mapping to the program learning objectives. Appropriate measurement of achievement of these objectives should be done for each course, and just prior to graduation from the program. The capstone course can provide a partial methodology for measuring overall learning objectives, but exit exams and professional certification exams can also be used.

6.4. How can a program adapt to changes in the discipline of software engineering?

ANSWER: A vibrant industrial advisory board can help keep a program current through annual review. Vigilant monitoring of the discipline and program revision at an appropriate rate is also needed. Three to five years is a generally accepted cycle for program revision of professional programs. In addition, the GSwE2009 guidelines themselves are intended to be continuously maintained in order to keep pace with changes in the discipline.

6.5. How can problems in areas such as course curricula, capstone projects, and student success be addressed?

ANSWER: In order to deal with programmatic issues, there should be a mechanism in place to rapidly identify these problems. The answers to the previous 2 questions provide guidance for monitoring the ongoing health of a program and dealing with evolutionary changes. Dealing with spike problems requires that the academic head of the program be vigilant and have the ability to make rapid changes in the event of a problem. In short, someone or some entity should be empowered to make changes rapidly in order to correct short-term problems, and the curriculum evolution structure should be in place to deal with long-term problems.

6.6. How can a program deal with changes in technology?

ANSWER: It should be the case that a graduate software engineering program is relatively insensitive to changes in hardware and software technologies. The issue of acquiring new or replacement hardware and software is covered in §3 (Acquiring Resources) of this document.

This page intentionally left blank.

7. Bibliography

This is not intended to be a comprehensive bibliography of all published work on software engineering education. It contains those sources that should be most helpful in creating and modifying graduate programs, and in comparing existing programs.

Citations are arranged in the following categories:

- Curriculum references and standards
- Historical summaries and surveys of programs
- Descriptions of new programs
- Evolutionary trends and program updates
- Notable collaborations between industry and academia, especially in the creation of new programs

7.1. Curriculum Development

[Bloom 1956]

Bloom, B. S. (Ed.), *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I, Cognitive Domain*, Longmans, 1956.

[Fairley 1978a]

Fairley, R.E., "Toward Model Curricula in Software Engineering," Proceedings of the Ninth SIGCSE Technical Symposium on Computer Science Education, *ACM SIGCSE Bulletin 10(3)*, 77-79, August 1978.

[Fairley 1978b]

Fairley, R.E., "Educational Issues In Software Engineering," *Proceedings of the 1978 Annual Conference of the ACM*, 58-62, December 1978.

[Freeman 1976]

Freeman, P., Wasserman, A.I., and Fairley, R.E., "Essential Elements of Software Engineering Education," *Proceedings of the 2nd International Conference on Software Engineering*, October 1976.

[GSWE2009]

Pyster, A. (Ed.), *Graduate Software Engineering 2009 (GSWE2009): Curriculum Guidelines for Graduate Software Engineering Education*, Stevens Institute of Technology, September 30, 2009.

[SE2004]

ACM/IEEE-CS Joint Task Force on Computing Curricula, *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, August 2004,
<<http://www.acm.org/education/curricula-recommendations>>.

[SEI 1989]

Ardis, M. and Ford, G., *SEI Report on Graduate Software Engineering Education*, CMU/SEI 89-TR-21, Software Engineering Institute, Carnegie Mellon University, June 1989.

[SEI 1991]

Ford, G., *SEI Report on Graduate Software Engineering Education*, CMU/SEI 91-TR-002, Software Engineering Institute, Carnegie Mellon University, April 1991.

[SWEBOOK]

P. Bourque and R. Dupuis (Eds.), *Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society Press, 2004.

7.2. History and Overview of Graduate Software Engineering Programs

[Bagert 2005]

Bagert, D. and Mu, X., "Current State of Software Engineering Master's Degree Programs in the United States," *Proceedings of 35th Annual Conference on Frontiers in Education*, 2005.

[Mead 2009]

Mead, N., "Software Engineering Education: How Far We've Come and How Far We Have to Go," *Journal of Systems and Software* 82, 571-575, 2009.

[Pyster 2009]

Pyster, A., Lasfer, K., Turner, R., Bernstein, L. and Henry, D., "Master's Degrees in Software Engineering: An Analysis of 28 University Programs," *IEEE Software* 26(5), 94-101, September-October 2009.

[Tomayko 1998]

Tomayko, J., "Forging a Discipline: An Outline History of Software Engineering Education," *Annals of Software Engineering* 6, 3-18, April 1998.

7.3. Establishment of New Graduate Software Engineering Programs

[Andrews University 1990]

Bidwell, D.R., "Master of Science in Software Engineering," *Proceedings of the 1990 SEI Conference on Software Engineering Education*, 148-149, April 2-3, 1990.

[Boston University 1988]

Brackett, J., Kincaid, T., and Vidale, R., "The Software Engineering Graduate Program at the Boston University College of Engineering," *Proceedings of the SEI Conference on Software Engineering Education*, 56-63, April 28-29, 1988.

[Carnegie-Mellon University 1990]

Gibbs, N., Ardis, M., Habermann, A.N., and Tomayko, J.E., "The Carnegie Mellon University Master of Software Engineering Degree Program," *Proceedings of the 1990 SEI Conference on Software Engineering Education*, 152-154, April 2-3, 1990.

[Florida Atlantic University 1994]

Coulter, N.S., and Dammann, J.E., "Current Practices, Culture Changes, and Software Engineering Education," *Computer Science Education* 5, 211-227, 1994.

[Frailey 1998]

Frailey, D., "Opportunities for Software Engineering Education," *Annals of Software Engineering* 6, 131-144, 1998.

[George Mason University 1988]

Fairley, R.E., "The Software Engineering Programs at George Mason University," *Proceedings of the SEI Conference on Software Engineering Education*, 64-69, April 28-29, 1988.

[National University 1990]

Olivier, D.P., and Hayward, R.R., "Master of Science in Software Engineering Program at National University," *Proceedings of the 1990 SEI Conference on Software Engineering Education*, 155-156, April 2-3, 1990.

[University of Saint Thomas 1990]

Folz, B.M., "Master of Software Design and Development," *Proceedings of the 1990 SEI Conference on Software Engineering Education*, 150-151, April 2-3, 1990.

[Seattle University 1978]

Stucki, L.G., and Peters, L.J., "A Software Engineering Graduate Curriculum," *Proceedings of the 1978 ACM Annual Conference*, 63-67, December 1978.

[Seattle University 1981]

Lee, K.Y., "Status of Graduate Software Engineering Education," *Proceedings of the 1981 ACM Annual Conference*, 179-183, January 1981.

[Southern Methodist University 1994]

Coyle, F., Forrest, E., Frailey, D.J., and Tanik, M., "Meeting the Needs of Industry: SMU's Master's Degree Program in Software Engineering," *Proceedings of the 7th SEI Conference on Software Engineering Education*, January 1994.

[Texas Christian University 1978]

Hoffman, A.A.J., "A Proposed Masters Degree In Software Engineering," *Proceedings of the 1978 ACM Annual Conference*, 54-57, December 1978.

[Texas Christian University 1985]

Comer, J.R., Conn, H.C., and Schember, K.A., "Software Design and Development: A Graduate Curriculum in Software Engineering," *Proceedings of the Sixteenth SIGCSE Technical Symposium on Computer Science Education*, *ACM SIGCSE Bulletin 17(1)*, 1985.

[Texas Christian University 1987]

Comer, J.R., and Rodjak, D.J., "Adapting to Changing Needs: A New Perspective on Software Engineering Education at Texas Christian University," *Proceedings of Software Engineering Education: The Educational Needs of the Software Community*, 149-171, February 27-28, 1986.

[Wang Institute 1983]

Fairley, R.E., and Martin, N., "Software Engineering Programs at the Wang Institute of Graduate Studies," *Proceedings of the 1983 Annual Conference on Computers: Extending the Human Resource*, January 1983.

7.4. Evolution of Graduate Software Engineering Programs

[Carnegie-Mellon 1995]

Garlan, D., Brown, A., Jackson, D., Tomayko, J., and Wing, J., "The CMU Master of Software Engineering Core Curriculum," *Proceedings of the 8th SEI Conference on Software Engineering Education*, 65-86, March 29 - April 1, 1995.

[Carnegie-Mellon 1996]

Tomayko, J.E., "Carnegie Mellon's Software Development Studio: A Five Year Retrospective," *Proceedings of the 9th Conference on Software Engineering Education and Training*, 119-129, April 21-24, 1996.

[George Mason 1994]

Ammann, P., Gomaa, H., Offutt, J., Rine, D., and Sanden, B., "A Five Year Perspective on Software Engineering Graduate Programs at George Mason University," *Proceedings of the 7th Conference on Software Engineering Education and Training*, 473-488, January 1994.

[University of Houston Clear-Lake 1995]

Atkinson, C., Eichmann, D., and McKay, C., "An Evolution of a Software Engineering Curriculum," *Proceedings of the 8th SEI Conference on Software Engineering Education*, 99-112, March 29 - April 1, 1995.

[Monmouth University 1987]

Drucker, H., Kuntz, R.A., and Swartz, G.B., "Software Engineering at Monmouth College," *Proceedings of the 1987 SEI Conference on Software Engineering Education*, 385-395, April 30 - May 1, 1987.

[Monmouth University 2003]

Rosca, D., Tepfenhart, W., and McDonald, J., "Necessary Metamorphoses of a Software Engineering Program," *Proceedings of the 16th Conference on Software Engineering Education and Training*, 129-139, March 2003.

[Seattle University 1987]

Mills, E.E., "The Master of Software Engineering at Seattle University after Six Years," *Proceedings of Software Engineering Education: The Educational Needs of the Software Community*, 182-200, February 27-28, 1986.

[Wang Institute 1987]

Ardis, M., "The Evolution of Wang Institute's Master of Software Engineering Program," *IEEE Transactions on Software Engineering*, 13(11), 1149-1155, November 1987.

[Wang Institute 1988]

Fairley, R. E., "A Post-Mortem Analysis of the Software Engineering Programs at Wang Institute of Graduate Studies," *Issues in Software Engineering Education*, Springer Verlag, 1988.

7.5. Collaboration between Universities and Industry in Software Engineering Education

[Beckman 1996]

Beckman, K., Coulter, N., Khahenoori, S., and Mead, N., "Collaborations: Closing the Industry-Academia Gap," *IEEE Software* 14(6), 49-57, 1996.

[Ellis 2002]

Ellis, H., Mead, N., Moreno, A., Tanner, C., and Ramsey, D., "Characteristics of Successful Collaborations to Produce Educated Software Engineering Professionals," *Computer Science Education* 12(1-2), 119-140, 2002.

[Frailey 2002]

Frailey, D. and Mason, J., "Using SWEBOK to Enhance Software Engineering Education Programs in Academia and Industry," *Proceedings of the 15th Conference on Software Engineering Education and Training*, February 2002.

[McCabe 1996]

McCabe, N., O'Mary, G., and Powel, K., "Stretching the McDonnell Douglas Software Training Budget: Striking a Balance between In-House and Outsourcing," *Proceedings of the 9th Conference on Software Engineering Education and Training*, 199-213, 1996.

7.6. Additional Sources of Information about Graduate Software Engineering Education

[CSEET]

Conference on Software Engineering Education and Training,
<<http://conferences.computer.org/cseet/>>

[FASE]

Forum for the Advancing of Software engineering Education (FASE),
<<http://sepe.wetpaint.com/page/FASE+Newsletters>>

[GSWE2009]

Graduate Software Engineering 2009 (GSWE2009) website,
<<http://www.gswe2009.org/>>

This page intentionally left blank.

8. GLOSSARY

8.1. Abbreviations and Codes

CAT	Curriculum Author Team of GSwE2009
CBOK	Core Body of Knowledge; <i>CBOK</i> is also the code for the outcome of mastering the Core Body of Knowledge
CS	Computer Science
DEPTH	<i>DEPTH</i> is the code for the outcome that requires a student to master an aspect of the Core Body of Knowledge to the Bloom's Synthesis level.
DOMAIN	<i>DOMAIN</i> is the code for the outcome that requires a student to master software engineering in at least one application domain (such as finance, medical, transportation, or telecommunications) and one application type (such as real-time, embedded, safety-critical, or highly distributed systems).
ETHICS	<i>ETHICS</i> is the code for the outcome that requires a student be able to make ethical professional decisions and practice ethical professional behavior.
GPA	Grade Point Average
GSwE	Graduate Software Engineering
GSwERC	Graduate Software Engineering Reference Curriculum, the name of earlier drafts of GSwE2009
GSwE2009	<i>Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering</i>
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering

iSSEc	Integrated Software and Systems Engineering Curriculum project
IT	Information Technology
KA	Knowledge Area
LEARN	<i>LEARN</i> is the code for the outcome that requires a student be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.
MI	Master of Informatics
MS	Master of Science
MSCS-SE	Master of Science in Computer Science-Software Engineering
MSE	Master of Software Engineering
MSSE	Master of Science in Software Engineering
MSWE	Master of Software Engineering
NDIA	U.S. National Defense Industrial Associate—Systems Engineering Division
PERSPECTIVE	<i>PERSPECTIVE</i> is the code for the outcome that requires a student to understand and appreciate feasibility analysis, negotiation, and good communications with stakeholders in a typical software development environment; and further requires that a student be able to perform those tasks well, have effective work habits and be a leader.
RECONCILE	<i>RECONCILE</i> is the code for the outcome that requires a student to be able to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.
SE	Systems Engineering
SE2004	<i>Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering,</i>

SwE	Software Engineering
SWEBOK	Software Engineering Body of Knowledge
SYS ENG	<i>SYS ENG</i> is the code for the outcome that requires a student to understand the relationship between SwE and SE and be able to apply SE principles and practices in the engineering of software.
TEAM	<i>TEAM</i> is the code for the outcome that requires a student to be an effective member of a team, effectively communicate both orally and in writing, and lead in one area of project development.
TECH	<i>TECH</i> is the code for the outcome that requires a student be able to analyze a current significant software technology, articulate its strengths and weaknesses, compare it to alternative technologies, and specify and promote improvements or extensions to that technology.
V&V	Verification and Validation

8.2. Terminology

Academic unit: A grouping of faculty for administrative purposes, such as a department, division, group, school, or college.

Academic term: Any breakdown or sub-division of the academic year, such as semester, quarter, or non-traditional self-contained instructional units.

Adjunct Faculty. See *External Faculty*.

Admission Requirements. Admission requirements are the minimum standards an individual must meet in order to enter an academic program. These requirements are generally mandatory, and waivers require justification. Admission requirements are not specified in GSwE2009. (See *Entrance Expectations*)

Architecture. Architecture refers to the framework used to develop software, which is specifically covered in the Core Body of Knowledge.

Bloom Taxonomy. A categorization of the intellectual activities associated with learning [Bloom 1956]. The taxonomy has six levels of activity: Knowledge (K), Comprehension (C), Application (AP), Analysis (AN), Synthesis (SYN), and Evaluation (EV). These levels are used to describe the depth to which curricula should cover specific elements in the Core Body of Knowledge (CBOK). The GSwE2009 Curriculum is focused primarily at the K, C, AP, and AN levels, with recommendation of SYN level understanding in an elective area. (Please see Appendix B of GSwE2009 for more information.)

Bridging Course. See *Leveling Course*.

Capstone Experience. A detailed and work-intensive endeavor that demonstrates the application of knowledge and skills gained in a program to a specific problem. Capstone projects have traditionally been in the form of a thesis. More recently, capstone projects that handle problems relevant to a particular industry segment or area of expertise and develop potential solutions have been included.

Certificate program: A collection of courses, delivered by an appropriate authority, leading to a certificate in a specialty area. Completion of courses in a certificate program may, depending on the authority, also be counted toward fulfilling the requirements for a degree.

Core Body of Knowledge (CBOK). The recommended knowledge areas that should be obtained within a software engineering master's degree program. In addition, the

CBOK provides a recommendation as to the appropriate Bloom's level for each knowledge area. (The CBOK is described in Section 6 of GSwE2009.)

Core materials. Fundamental skills and knowledge that all students must master within a given program.

Course. A collection of material, exercises and assessment for which academic credit is awarded, which may be part of a number of programs.

Course Credits. See *Credit Hours*.

Credit Hours. A unit used to indicate the amount of in-class time for a given course. Generally, this refers to one hour of class time per week per term. This may be affected by the types of terms used (e.g., semesters vs. quarters) and by the instructional mode (e.g., on-line vs. traditional classroom). (Also referenced as course credits.)

Curriculum. All the courses associated with a specific course of study. The curriculum will depend on the level (e.g., graduate or undergraduate) and specificity (i.e., discipline or specialty) of the course of study.

Degree Program. A collection of courses, delivered by an appropriate authority, leading to an academic degree.

Elective Materials. A set of courses to accommodate different interests and goals of individual students that may include special topics.

Entrance Expectations. Knowledge and skills expected of students when they enter an academic program. These are often prerequisites to the topics they will study.

External faculty: Members of the faculty who have a short-term position within the academic unit, as opposed to regular faculty who have a permanent appointment. Visiting faculty and adjuncts are considered external.

Faculty. Academic or teaching staff. These may include both full-time permanent staff who are employed in an academic unit and external staff attached to the program, such as adjuncts.

GSwE2009-Satisfying Program. A university program that offers a master's in software engineering with a curriculum that largely satisfies GSwE2009 recommendations. Reasonable deviation from those recommendations for individual university or program constraints is expected. There is no precise measure of how much deviation is "reasonable."

Leveling Course. A course designed to allow students who do not meet entrance expectations to enroll in an academic program. In general, these are courses

designed to ensure that students have the requisite knowledge, skills, and abilities to succeed in the program. These may also be referred to as *bridging courses* or *preparatory courses*.

Master's Degree. A graduate or professional-level degree intended to follow an undergraduate course of study. Within GSwE2009, a master's degree in software engineering is focused on developing knowledge, skills, and abilities to meet the current and future challenges of complex systems that require software in order to operate properly.

Module: A self-contained (i.e., can stand alone) distinguishable unit of instruction in a course, such as a lecture, exercise, or case study in a course on software design. Modules, although self-contained, can build upon one another. For instance, in a course on software testing one could introduce the topic of modularity in a module and then present a follow-on module on the ramifications of modularity on the testability of software.

Outcomes. The expected accomplishments of an individual who has completed an academic program.

Pedagogy. The style of instruction and strategies used within a specific course of study.

Preparatory Course. See *Leveling Course*.

Program. See *Degree Program*.

Program track. A specific set of courses within a program that emphasizes different areas of study such as telecommunications, real-time systems, and information systems.

Practical experience. Professional experience that allows a student to be exposed to a team environment and the product life cycle in the context of software engineering.

Reference Curriculum. A set of outcomes, entrance expectations, architecture, and a body of knowledge that provide guidance for faculty who are designing and updating their programs. That guidance is intentionally flexible so that faculty can adopt and adapt it based on local programmatic needs. A reference curriculum is not intended to be used directly for program certification or accreditation.

Regular faculty: Members of the faculty who have a permanent position within the academic unit, as opposed to external faculty who have a short-term appointment.

Senior faculty: Members of the teaching staff who have been recognized in the institution as having a level of seniority and experience sufficient to be responsible for the quality of a program. The definition will vary between

institutions, but is usually a mixture of teaching experience (possibly formally assessed via mentoring or peer review), along with a track record of publications in the subject area and a level of recognition or esteem in the field. The rank of senior faculty is often linked to the concept of tenure used in some countries (e.g., the United States).

Software Engineering (SwE). A systematic approach to the development of operational software, and the maintenance of that software.

SWEBOK. The IEEE's *Software Engineering Body of Knowledge* [SWEBOK].

Track. See *Program Track*.

Visiting Faculty. See *External Faculty*.