Comparisons of GSwE2009 to Current Master’s Programs in Software Engineering

Version 1.0

November 10, 2009

Unlimited Global Distribution
Stevens Institute of Technology® 2009
This page intentionally left blank.
Table of Contents

PREFACE .......................................................................................................................................................... VIII

ACKNOWLEDGMENTS ........................................................................................................................................ IX

EXECUTIVE SUMMARY OF GSWE2009 ........................................................................................................ X

1 OVERVIEW OF THIS COMPARISON DOCUMENT ...................................................................................... 1

1.1 METHODOLOGY ......................................................................................................................................... 2

1.1.1 Phases of the Comparison Process ........................................................................................................... 2

1.1.2 Final Comparisons .................................................................................................................................. 2

1.1.3 Comparison Challenges ............................................................................................................................ 3

1.1.4 Data Protection ...................................................................................................................................... 4

2 CHARACTERISTICS OF PARTICIPATING PROGRAMS ................................................................................... 5

2.1 OVERVIEW ................................................................................................................................................... 5

2.2 GEOGRAPHIC LOCATION ............................................................................................................................. 5

2.3 COURSE DELIVERY STRUCTURE ................................................................................................................. 5

2.4 PROGRAM SIZE ........................................................................................................................................... 6

2.4.1 Size as Measured by Number of Graduating Students .............................................................................. 6

2.4.2 Size as Measured by Number of Course Offerings .................................................................................. 7

2.4.3 Size as Measured by Number of Faculty ................................................................................................. 7

2.5 STUDENT/FACULTY RATIO ....................................................................................................................... 10

2.6 TYPE/TITLE OF DEGREE OFFERED ........................................................................................................ 10

2.7 ENTRANCE REQUIREMENTS ...................................................................................................................... 11

2.7.1 Degree Requirements ................................................................................................................................ 11

2.7.2 Software Engineering Requirements ...................................................................................................... 11

2.7.3 Experience Requirements ........................................................................................................................ 12

3 FULFILLMENT OF GSWE2009 OUTCOMES .............................................................................................. 13

3.1 OVERVIEW ................................................................................................................................................... 13

3.2 CBOK ............................................................................................................................................................ 14

3.3 DOMAIN ...................................................................................................................................................... 15

3.4 DEPTH .......................................................................................................................................................... 15

3.5 ETHICS .......................................................................................................................................................... 15

3.6 SYS ENG ....................................................................................................................................................... 16

3.7 TEAM ........................................................................................................................................................... 17

3.8 RECONCILE ............................................................................................................................................... 17

3.9 PERSPECTIVE .............................................................................................................................................. 18

3.10 LEARN ......................................................................................................................................................... 18

3.11 TECH .......................................................................................................................................................... 19

3.12 FINDINGS AND CONCLUSIONS BASED ON STUDENT OUTCOME ACHIEVEMENT .................................. 20

3.12.1 General Conclusions and Remarks ....................................................................................................... 20

3.12.2 Conclusions Related to Hypothetical Student Achievement of Outcomes ........................................... 20

4 PROFILES AND ANALYSES OF SELECTED PARTICIPATING PROGRAMS .................................................. 21

4.1 UNIVERSITY A OUTCOME ACHIEVEMENT ................................................................................................. 22

4.2 UNIVERSITY D OUTCOME ACHIEVEMENT ................................................................................................. 24

4.3 UNIVERSITY H OUTCOME ACHIEVEMENT ................................................................................................. 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.7.1</td>
<td>Entrance Requirements</td>
<td>75</td>
</tr>
<tr>
<td>G.7.2</td>
<td>Program Architecture</td>
<td>75</td>
</tr>
<tr>
<td>G.7.3</td>
<td>Program Core</td>
<td>75</td>
</tr>
<tr>
<td>G.8</td>
<td>UNIVERSITY H PROFILE</td>
<td>77</td>
</tr>
<tr>
<td>G.8.1</td>
<td>Entrance Requirements</td>
<td>77</td>
</tr>
<tr>
<td>G.8.2</td>
<td>Program Architecture</td>
<td>77</td>
</tr>
<tr>
<td>G.8.3</td>
<td>Program Core</td>
<td>77</td>
</tr>
<tr>
<td>G.9</td>
<td>UNIVERSITY I PROFILE</td>
<td>79</td>
</tr>
<tr>
<td>G.9.1</td>
<td>Entrance Requirements</td>
<td>79</td>
</tr>
<tr>
<td>G.9.2</td>
<td>Program Architecture</td>
<td>79</td>
</tr>
<tr>
<td>G.9.3</td>
<td>Program Core</td>
<td>79</td>
</tr>
<tr>
<td>G.10</td>
<td>UNIVERSITY J PROFILE</td>
<td>80</td>
</tr>
<tr>
<td>G.10.1</td>
<td>Entrance Requirements</td>
<td>80</td>
</tr>
<tr>
<td>G.10.2</td>
<td>Program Architecture</td>
<td>80</td>
</tr>
<tr>
<td>G.10.3</td>
<td>Program Core</td>
<td>80</td>
</tr>
<tr>
<td>G.11</td>
<td>UNIVERSITY K PROFILE</td>
<td>81</td>
</tr>
<tr>
<td>G.11.1</td>
<td>Entrance Requirements</td>
<td>81</td>
</tr>
<tr>
<td>G.11.2</td>
<td>Program Architecture</td>
<td>81</td>
</tr>
<tr>
<td>G.11.3</td>
<td>Program Core</td>
<td>81</td>
</tr>
<tr>
<td>G.12</td>
<td>UNIVERSITY L PROFILE</td>
<td>82</td>
</tr>
<tr>
<td>G.12.1</td>
<td>Entrance Requirements</td>
<td>82</td>
</tr>
<tr>
<td>G.12.2</td>
<td>Program Architecture</td>
<td>82</td>
</tr>
<tr>
<td>G.12.3</td>
<td>Program Core</td>
<td>82</td>
</tr>
</tbody>
</table>
List of Figures

FIGURE 1. Geographic distribution of participating organizations.................................................................6
FIGURE 2. Utilization of course delivery methods across participating programs..............................................6
FIGURE 3. Program size by number of students graduating per year. The data are arranged largest to smallest.................................................................7
FIGURE 4. Program size by number of courses. The data are arranged largest to smallest.........................8
FIGURE 5. Program size by number of full-time faculty. The data are arranged largest to smallest......10
FIGURE 6. Program size by number of part-time faculty. The data are arranged largest to smallest...11
FIGURE 7. Comparison of the number of full-time and part-time faculty for participating programs, with data sorted by: a) number of full-time faculty, b) number of part-time faculty, c) the ratio of full-time to part-time faculty, and d) the total faculty count (assuming 1 full-time faculty member=4 part time faculty, calculated as F_total = F_full-time + F_part-time/4).................................................11
FIGURE 8. Ratio of the number of students to the total number of faculty (full and part-time) for each program, arranged largest to smallest.................................................................12
FIGURE 9. Required knowledge areas for program entrance........................................................................12
FIGURE 10. Student achievement of CBOK outcome. ..............................................................................16
FIGURE 11. Student achievement of Domain outcome. ...........................................................................17
FIGURE 12. Student achievement of Depth outcome. ..............................................................................17
FIGURE 13. Student achievement of Ethics outcome. ..............................................................................18
FIGURE 14. Student achievement of Sys Eng outcome...........................................................................18
FIGURE 15. Student achievement of Team outcome. ..............................................................................19
FIGURE 16. Student achievement of Reconcile outcome. .......................................................................19
FIGURE 17. Student achievement of Perspective outcome. ..................................................................20
FIGURE 18. Student achievement of Learn outcome. .............................................................................20
FIGURE 19. Student achievement of Tech outcome. .............................................................................21
FIGURE 20. Average fulfillment of each outcome, ordered from highest to lowest fulfillment (axis scale contains only M- to M+ to better show differences).................................................20
FIGURE 21. Fulfillment of outcomes by University A students.................................................................24
FIGURE 22. Outcome achievement at entry and completion for University A students..........................23
FIGURE 23. Fulfillment of outcomes by University D students.................................................................26
FIGURE 24. Outcome achievement at entry and completion for University D students.........................25
FIGURE 25. Fulfillment of outcomes by University H students.................................................................28
FIGURE 26. Outcome achievement at entry and completion for University H students..........................27
List of Tables

Table 1. Definitions of Full-time for Program Faculty ........................................................................ 8
Table 2. Types of Degrees Offered by Participating Programs .................................................................. 11
Table 3. University A’s Hypothetical Student Profiles ............................................................................. 22
Table 4. University D’s Hypothetical Student Profiles ............................................................................. 24
Table 5. University H’s Hypothetical Student Profiles ............................................................................. 27
Table 6. Activities Requires for Collaboration between Participating Faculty and the Curriculum
Author Team (CAT) ................................................................................................................................. 45
Table 7. Abbreviated Description of GSWE2009 Outcomes ................................................................... 61
Table 8. Abbreviated Description of the GSWE2009 CBOK ................................................................... 63
Preface

This is a companion document to *Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering*. The purpose of this companion document is to report on a series of comparisons between the recommendations of GSwE2009 and several currently-offered graduate programs in software engineering. To provide context for the comparisons, this document begins with a summary of the GSwE2009 recommendations.
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, Larry Bernstein, Dennis Frailey, Barrie Thompson, Massood Towhidnejad, Joe Urban and Mary Jane Willshire. Editors were Mark Ardis, Dennis Frailey and Nicole Hutchinson.

Special thanks go to Dennis Frailey for leading the team and carrying much more than his share of the workload. Graduate assistant Nicole Hutchison provided extensive technical assistance in analyses and performed many enabling tasks for logistical support. Additional support was provided by Kahina Lasfer.

Several authors of GSwE2009 provided helpful comments and suggestions for participating schools and contacts. In addition we thank the anonymous contributors of comparison data for their valuable input.

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.

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 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 GSwE2009 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 SE2004

### Summary of outcomes

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CBO</strong></td>
<td>Master the CBOK.</td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Depth</strong></td>
<td>Master at least one KA or sub-area from the CBOK to at least the Bloom Synthesis level.</td>
</tr>
<tr>
<td><strong>Ethics</strong></td>
<td>Be able to make ethical professional decisions and practice ethical professional behavior.</td>
</tr>
<tr>
<td><strong>Sys Eng</strong></td>
<td>Understand the relationship between SwE and SE and be able to apply SE principles and practices in the engineering of software.</td>
</tr>
<tr>
<td><strong>Team</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Reconcile</strong></td>
<td>Be able to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.</td>
</tr>
<tr>
<td><strong>Perspective</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Learn</strong></td>
<td>Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.</td>
</tr>
<tr>
<td><strong>Tech</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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 form a foundation for achieving the 10 outcomes previously described. However, it is recognized that individual programs may start with a student population that has characteristics different from what GSwE2009 presumes here. Such programs will likely have to augment their master’s programs with supplementary courseware or other means in order for their students to achieve all 10 outcomes—or they will deliberately choose not to adopt some outcomes. Furthermore, programs may choose to add other outcomes to favor their particular markets and institutional emphases. GSwE2009 is not intended as a basis for certifying or accrediting either programs or individuals. It is, rather, a set of recommendations to 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 several 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 to 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* [this
document] and *Frequently Asked Questions on Implementing GSwE2009* – are being prepared concurrently with the release of GSwE2009. They will be available in Fall 2009 at [www.GSwE2009.org](http://www.GSwE2009.org) and updated regularly. These are not under ACM or IEEE-CS stewardship but, rather, are maintained by Stevens Institute.
1 Overview of This Comparison Document

When creating a new reference curriculum model it is natural to ask how it compares with current programs. The purpose of this document is to address that question. The document presents the results of comparing the new GSwE2009 curriculum guidelines with several existing graduate level software engineering (SwE) programs. The results are presented in three ways:

- **Section 2** presents a summary of the demographic information provided by the twelve participating programs/universities. The goal is to show the range and breadth of the SwE programs used for this comparison. Certain observations are made regarding faculty size and composition, entrance requirements, program size and other demographic characteristics.

- **Section 3** presents a series of analyses and top-level conclusions based on the expected outcome achievements of 34 hypothetical students pursuing graduate SwE degrees at these various universities. The purpose is to assess how well typical students attending currently-offered SwE programs would achieve the objectives of GSwE2009. The section concludes with several observations based on these analyses.

- **Section 4** provides high-level profiles and analyses of a selection of the participating programs and makes observations based on the specific characteristics of each participating program. This provides insight into the effect of each program on student achievement. This section concludes with several overall observations and identifies a number of issues for further research.

To obtain the hypothetical student descriptions and other information the GSwE2009 team worked with representatives of each participating graduate software engineering program. Each representative conducted structured comparisons between their current program and the recommendations of GSwE2009. The most important aspect of those comparisons is how well students would achieve the ten GSwE2009 outcomes. A tenet of GSwE2009 is that a program satisfying all of its recommendations should graduate students who achieve all ten outcomes at a relatively high level.

One might ask why we did not simply compare the curricula in these current programs with the curriculum recommendations of GSwE2009. Although we do make curriculum observations at several points, several factors make curriculum comparisons a poor choice for understanding how current programs compare with GSwE2009:

- GSwE2009 focuses on outcomes and leaves much room for choice in how to achieve those outcomes. The curriculum recommendations of GSwE2009 are very flexible, focusing on core topics to be covered rather than defining courses and leaving room for a large number of elective and/or program-specific courses.

- The curricula of the programs examined are also very flexible.
• The curricula and other characteristics of the programs examined vary substantially from one another.

GSwE2009 was created to “set the bar” for the global community – by intention, a higher bar than is common today. If students in existing programs already achieve nearly all GSwE2009 outcomes at a high level, that could indicate GSwE2009 has set the bar too low. On the other hand, if students in successful, well-established programs fall dramatically short of attaining the majority of GSwE2009 objectives, it could signal that GSwE2009 is unreasonably demanding and will have little impact on the academic community. As will be seen throughout this document, GSwE2009 is somewhere between these extremes.

1.1 Methodology

The following is the approach used to develop and collect the comparisons.

1.1.1 Phases of the Comparison Process

There were three phases to the comparison process.

1. Prior to development of the GSwE2009 model, an assessment was made of 28 current, graduate-level software engineering programs [Pyster 2009]. This helped the GSwE2009 Curriculum Author Team (CAT) understand, at an overall level, the current state of graduate software engineering curricula. The information was of significant value in helping the CAT devise a model that met the objective of raising the bar but not too high.

2. An early draft version of the model (GSwERC 0.25) was compared in detail with six existing programs, both as a way to calibrate the model (how different is it from current practice) and to assess the comparison methods initially proposed. This enabled the CAT to refine the comparison approach.

3. The final draft of the model and an improved comparison method were used by those same six programs to refine their comparisons. They were also used to develop comparisons with several additional programs, so as to provide a richer set of data.

1.1.2 Final Comparisons

In its refined form, each comparison was created as a collaborative effort between faculty representing a particular university’s SwE graduate program and representatives of the GSwE2009 Curriculum Author Team (CAT). The CAT provided faculty with a template and rubric for assessing their programs and, when possible, followed a standard collaboration protocol (please see Appendices A and B). After this initial exercise, CAT members discussed the comparisons with the faculty to ensure a common baseline for evaluation. This information was then used to create the profiles and descriptions reported in this document and formed the basis of the analyses described herein.
Each comparison has five elements:

1. Description of the program, including what makes it distinctive.
2. Description of program admission requirements.
3. Description of two or three typical students with diverse backgrounds and interests. Note that these are hypothetical students, not actual students, although they are expected to be representative of common student profiles for a given program. Among other things, the description included an assessment of how well each student would have attained the GSwE2009 outcomes upon entry to the software engineering program.
4. A set of courses each student might take to complete their master’s degree, including required and elective courses.
5. How well each student taking the courses identified in (4) above would be expected to satisfy each of the ten outcomes upon completion of the software engineering program.

The decision to focus on comparing hypothetical students for each program, rather than programs themselves, was influenced by the lessons learned from earlier comparison efforts (GSwERC versions 0.25 and 0.5). It was observed that many current programs present a variety of options for completing a degree. Typically, some options are more likely to foster attainment of the GSwE2009 objectives than others.

1.1.3 Comparison Challenges

The comparisons herein are constructed only with respect to diverse, hypothetical, but typical, students; not all the students who enter a master’s program or all the options available to a typical student are represented. The more heterogeneous the student population and the available options for a single program, the less comprehensive the sample of students will be. Nevertheless, this comparison is significantly richer and more insightful than the one performed for GSwERC 0.25 or GSwERC 0.5. It is planned that future versions of this comparison document will provide even more insightful comparisons between existing programs and the GSwE2009 recommendations.

Comparison information for each program was provided by a faculty representative, many of whom were also members of the CAT. The CAT members who provided this information engaged in a leveling exercise to improve the consistency of the data. In cases where the university was not represented on the CAT, our process called for a CAT member to review the data with the faculty representative (to maintain consistency). Despite these efforts, a number of inconsistencies remain, driven in part by the many differences among universities. In future work, the data collection methods will be improved based on this experience.

Generally, submitting programs developed the data set through collaborative efforts with multiple faculty members. However, the information presented here is not an “official” product of any reporting university and represents only the opinion of the authoring professor(s).
1.1.4 Data Protection

Another important decision for this version of the comparison has to do with anonymity. The comparisons include two or three typical students for the purpose of gaining insight on the GSwE2009 model. The purpose of these comparisons was not to form a value judgment regarding any specific university or academic program. These efforts benefit greatly from the data supplied by participating universities, and this document seeks both to assure that the data are representative and to avoid the risk of a too-broad brush that might reflect unfairly or inaccurately on the quality of a university’s software engineering program. Therefore, universities or programs have not been identified by name in this report.
2 Characteristics of Participating Programs

2.1 Overview

There are twelve programs included in this first version of the GSwE2009 comparison analysis. These span the spectrum from very large programs partnered with industry to very small programs offered within other specialties, such as computer science. This section characterizes the sampled programs according to many demographic characteristics, including size, student/faculty ratio and entrance requirements.

Individuals examining this analysis in order to understand how their programs compare with the GSwE2009 model are encouraged to examine the information below and determine the program(s) most similar to their own. If a faculty member would like to complete a full comparison with the GSwE2009 recommendations, he or she may visit the comparison page on the GSwE2009 website (www.gswe2009.org).

2.2 Geographic Location

Of the twelve programs in our comparison, 9 were located in North America, 2 in South America, and 1 in Australia.

In future versions of the Comparisons, the CAT hopes to include many additional examples, including more from outside North America.

2.3 Course Delivery Structure

There were five main delivery methods for the programs examined: classroom, distance, executive, off-site, and distributed:

- Classroom, face-to-face delivery is a traditional course structure, where students meet with professors for several hours a week over the course of a semester or quarter. These may include laboratories as well as lectures.

- Distance learning encompasses both traditional online courses as well as other distance learning methods, such as video teleconferencing and interactive television.

- Executive course delivery is face-to-face instruction provided over an abbreviated time scale. For example, a university may provide 40 hours of instruction over 5 8-hour days or five weekends. This delivery style is generally suited to professionals who would have a difficult time...
attending a traditional semester or quarter course.

- Off-site delivery is face-to-face instruction provided outside of the traditional academic environment, often in an industrial setting.

- Distributed learning connects teams of students from different locations and encourages teamwork and collaboration across geographic and time differences, generally via the internet and often using specialized tools for this purpose.

Of the programs examined, all offer traditional, in-class, face-to-face course delivery. Five of the programs offer distance learning opportunities. The executive, off-site, and distributed options are less common, with only 2 schools offering executive and off-site courses and only 1 school consistently using distributed learning.

2.4 Program Size

2.4.1 Size as Measured by Number of Graduating Students

The size of a program’s student body was categorized using the average number of program graduates each year. This measure was used to provide a consistent standard without having to distinguish between full-time and part-time enrollment.

For discussion of the size of programs, the following scale was used:

- Small: Fewer than 15 students graduating per year
- Medium: 16-30 students graduating per year
- Large: 31-45 students graduating per year
- Very large: More than 45 students graduating per year

Using this scale, three of the universities included have large programs, graduating at least 31 students per year. Four of the universities fall into the medium category, while four are small, with fewer than 15 students graduating each year. One school did not provide graduation information for this first version of the Comparisons.
2.4.2 Size as Measured by Number of Course Offerings

The size of a program may also be measured by the number of software engineering course offerings taught per year. This includes only courses for which credit is given toward the software engineering degree and does not include leveling or articulation courses. If multiple sections or offerings of a course are taught each year, each offering is counted as a separate course offering.

Of the twelve programs examined, eleven provided data on the number of course offerings taught per year. Only three of these programs offer 10 or fewer courses per year. There appears to be no clear correlation between the number of courses offered and the size of the student body. One of the programs with few courses is classified as “large” and the other as “small,” as measured by the number of graduates per year.

The other eight programs providing data all offer between 15 and 40 courses per year. As with the programs offering fewer courses, there seems to be little correlation between the number of courses offered per year and the number of students graduating each year. This suggests that the programs vary widely in the number of options available to students. For example, a large program with few courses might be one where all students take mostly the same courses or take many non-SwE courses, whereas a medium program with many courses might be one that offers students a rich array of SwE options.

2.4.3 Size as Measured by Number of Faculty

2.4.3.1 Definition of Full-Time

Each program defines “full-time” a little differently. The assumption used for development of the comparisons was that a full-time instructor would provide “six courses meeting for at least 150 minutes each week for 12-14 weeks” per year. If the university defined full-time differently, the faculty members providing the comparisons were asked to provide definitions.
Table 1 provides some of the different definitions received. There was no overlap in the definitions provided. Full-time status appears to be affected by the academic term (semester or quarter) and the faculty’s status as tenure- or non-tenure track. A university may define full-time in terms of the number of hours spent teaching, the number of courses taught, the number of credit hours taught, or a percentage of the time dedicated to a program.

Table 1. Definitions of Full-time for Program Faculty

<table>
<thead>
<tr>
<th>Definition of Full-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faculty who teach 4 courses each year and who do the equivalent of 4 courses of research each academic year.</td>
</tr>
<tr>
<td>Faculty who teach 9 credit hour course load per semester (Fall and Spring).</td>
</tr>
<tr>
<td>A professor with 6 or more courses taught per year.</td>
</tr>
<tr>
<td>A faculty member that works 75% or more for the software engineering program.</td>
</tr>
<tr>
<td>A faculty member teaching a full load of software engineering courses only.</td>
</tr>
<tr>
<td>There are two different definitions. For tenure-track faculty, those who teach 4 courses in 3 quarters. For non-tenure-track, those who teach 8 courses in 3 quarters.</td>
</tr>
<tr>
<td>Since all faculty must perform research as well as teach, most teach only two to four courses per year.</td>
</tr>
<tr>
<td>A faculty member who teaches two courses per year.</td>
</tr>
<tr>
<td>A faculty member with 40 hrs per week totally dedicated to the program. All projects by the faculty member are developed under the department umbrella.</td>
</tr>
</tbody>
</table>

2.4.3.2 Full-Time Faculty

Obviously, with such dramatic differences in the definition of “full-time,” it is difficult to infer much from the number of full-time faculty included in a program. However, the CAT did want to understand the landscape of programs in terms of the number of faculty whom they consider full-time.

Of the twelve programs providing data on full-time faculty, ten have fewer than eight full-time faculty members. However all but one have at least three and most have four or more. These programs span the spectrum in terms of number of graduates per year (small to large programs) and offer substantially different numbers of courses per year. The program reporting only one full-time software engineering faculty member has a software engineering specialty within another degree, and most of the faculty teach both full-time and part-time courses.

Figure 5. Program size by number of full-time faculty. The data are arranged largest to smallest.
2.4.3.3 Part-Time Faculty

All faculty who are not considered to be full-time, but who support the SwE program, are considered to be adjunct or part-time faculty.

Of the twelve participating programs, eleven provided data on the number of part-time faculty used by their programs. Two of these programs do not use any part-time faculty, relying entirely on the full-time professorial staff. Most programs use a small contingent of part-time faculty, and this seems to be independent of the number of graduates per year or the number of courses taught per year. One program, however, has a very high number of part-time faculty. This particular program’s paradigm is focused on having a large number of faculty with close ties to industry. Because many of these faculty members are also still working in industry, they teach few courses.

2.4.3.4 Ratio of Full-Time to Part-Time Faculty

There appears to be no standard ratio of full-time to part-time faculty. Some universities use only a few full-time faculty to help oversee the program, while instruction is provided primarily by adjunct faculty. At the other end of the spectrum, some use only full-time positions. There does not appear to be a correlation between the number of graduates per year or the number of courses offered per year and the ratio of full-time to part-time faculty.

Figure 6. Program size by number of part-time faculty. The data are arranged largest to smallest.

Figure 7. Comparison of the number of full-time and part-time faculty for participating programs, with data sorted by: a) number of full-time faculty, b) number of part-time faculty, c) the ratio of full-time to part-time faculty, and d) the total faculty count (assuming 1 full-time faculty member=4 part time faculty, calculated as $F_{\text{total}}=F_{\text{full-time}}+F_{\text{part-time}}/4$).
The only commonality among these data is that most programs seem to operate with a total of at least four and at most ten full-time equivalent SwE faculty members. Although no other common findings could be gained from this profile, the data do indicate that the sample of programs currently included provides a view across the spectrum of faculty utilization.

2.5 Student/Faculty Ratio

There was no standard ratio of the number of graduates per year to the total number of faculty (full- and part-time). A counter-intuitive observation is that some of the large programs have a low student-faculty ratio, while some of the small programs have a much higher ratio. The program with the largest student-faculty ratio, however, is a large program, with an average of 45 graduates per year, and the smallest student-faculty ratio is a small program (fewer than 15 graduates per year). For this latter program, the number of total faculty is actually higher than the average number of graduates per year.

2.6 Type/Title of Degree Offered

Table 3 provides an overview of all the degree types reported in the comparisons. The most common master's degree title is “Master of Science in Software Engineering.” The other titles occurred only once each. It should be noted that some universities that consider themselves to have software engineering programs offer degrees that do not actually refer to “software engineering” in the title. In our sample there were three such examples. The schools offering the master of information technology and master of informatics degrees are from Australia and South America, respectively. The combined degree in systems engineering and computer science is offered by a South American school.
Table 2. Types of Degrees Offered by Participating Programs

<table>
<thead>
<tr>
<th>Degree Title</th>
<th>Number of Schools Offering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master of Science in Software Engineering</td>
<td>7</td>
</tr>
<tr>
<td>Master of Software Engineering</td>
<td>1</td>
</tr>
<tr>
<td>Master of Computer Science Software Engineering</td>
<td>1</td>
</tr>
<tr>
<td>Master of Information Technology</td>
<td>1</td>
</tr>
<tr>
<td>Master of Informatics</td>
<td>1</td>
</tr>
<tr>
<td>Master of Systems Engineering and Computer Science</td>
<td>1</td>
</tr>
</tbody>
</table>

2.7 Entrance Requirements

2.7.1 Degree Requirements

GSwE2009 expects that any student entering a master’s program in SwE will have a bachelor’s degree in an engineering field or other technical or scientific discipline. Of the twelve schools providing data, nine have a similar acceptance criterion. These nine schools require a bachelor’s degree in computer science, information technology, engineering, or science.

The three programs that do not require this type of bachelor’s degree all require prerequisite knowledge, such as formal mathematics, computer languages, and basic SwE and CS. For these programs, if a student does not possess the prerequisite knowledge, they may take (undergraduate) leveling courses before being formally accepted to the SwE master’s program.

2.7.2 Software Engineering Requirements

All twelve of the schools participating in the comparisons require some level of SwE knowledge prior to acceptance into the program. However, the knowledge areas required differ among schools, as seen in Figure 9. (Due to inconsistent descriptions, the figure reflects the judgment of the authors.) Though not all programs specify the knowledge areas directly in their data, it was assumed that a program requiring a bachelor’s degree in CS or SwE would require knowledge in the areas of software design, software development/maintenance, programming, and data structures.
Figure 9. Required knowledge areas for program entrance.

All programs require some prerequisite knowledge in software development and maintenance and most (ten) require some understanding of software design and advanced mathematics. Approximate three quarters of the programs surveyed require understanding in data structures and computer programming, and half require computer science. A few explicitly stated a requirement for prerequisite knowledge in computer engineering or object-oriented design. However, it should be noted that the comparison instrument did not explicitly ask for a list of prerequisite knowledge areas, so the numbers may change as this information is supplemented in future comparisons.

Most schools allow students to apply for admission into the SwE program even without the prerequisite knowledge, but require that they obtain the knowledge through (undergraduate) leveling courses prior to formal program acceptance.

2.7.3 Experience Requirements

Most of the schools surveyed indicated that they preferred students to have some experience in the workplace with software engineering prior to attending the program, and most of the hypothetical students have such experience. However, only three of the schools require this as a criterion for acceptance into the program. Of these, two require at least one year of experience, and the other requires at least two years. One school noted that it formerly required two years of experience, but it dropped that requirement after finding that appropriate leveling courses and internship opportunities enabled students without prior experience to succeed in their program. This suggests that with appropriate support from a program, lack of experience may be overcome.
3 Fulfillment of GSwE2009 Outcomes

3.1 Overview

GSwE2009 stipulates 10 outcomes that a student from a GSwE2009-aligned program should be able to achieve upon graduation. A brief overview of the outcomes is provided in Appendix D of this document. (Please see Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering for the full discussion of the outcomes.) Each program provided up to three scenarios via their typical student profiles, and developed a list of the courses each typical student would take, including both core and elective courses. Ideally, these courses, combined with the capstone experience for the program (if any) and the expected skills upon entrance, would be used to predict the outcome achievement for the students. In the most recent comparison process, each program was asked to provide a chart of expected outcome achievement, for each hypothetical student, as follows:

1. upon entry to the program (after completing all entry criteria, including any leveling courses);
2. upon completion of the core courses;
3. upon completion of elective courses; and
4. overall achievement from all of the above plus any capstone experience upon completion of the program.

Due to some confusion in the data collection process, we were unable to obtain all of this data from all participating schools. However, each of them provided at least #3, so this section uses #3: the “expected outcome achievement upon completion of elective courses” to compare students across each of the 10 GSwE2009 outcomes. Section 4 provides a more in-depth assessment of three programs that did supply the above information.

It should be noted that fulfillment of outcomes is highly dependent not only on the program, but on the background of the individual student, the level of student experience, and the student’s choices in elective courses. Therefore, as previously explained, the CAT decided that, for this first version of the Comparisons, individual students’ achievements of the GSwE2009 outcomes will be compared, and that when looking at programs (next section) there would be no attempt to characterize a program solely on the basis of the hypothetical students provided.

For each outcome, a chart is provided that depicts the individual student achievements upon completion of elective courses. Each data point has a university signifier (A-L) as well as a student ID number (1-3) which identifies the student background associated with that data point. The chart shows achievement on the scale of “0” to “H”, which correspond to the following values:
3.2 CBOK

The CBOK outcome deals with a student’s ability to master the Core Body of Knowledge (CBOK). (A brief description of the CBOK Knowledge Areas (KAs) and the topics for each area can be found in Appendix F. For more information, please see Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering.)

Most students currently attain the CBOK outcome at the Medium to High level. This suggests that the CBOK outcome is attainable for most students in current SwE programs. Because the CBOK is based on currently-used standard bodies of knowledge (such as the SWEBOK), this is not entirely unexpected.

It should be noted that, in some programs, different students achieve significantly different levels of attainment of the CBOK knowledge. This is generally associated with programs that offer a richer variety of courses and/or options.

In order to ensure better coverage of the CBOK, universities would not necessarily need to redesign their programs. Conversations with data providers suggest that many gaps could be closed by adding or changing materials in some courses.
3.3 Domain

The Domain outcome deals with mastery of an application domain (e.g., finance, medical, transportation, or aerospace) and one application type (e.g., real-time, embedded, safety critical, or highly distributed). This mastery includes student understanding of how differences in domain and type manifest themselves in both the software itself and in its engineering, and includes understanding of how to learn a new application domain or type.

Of the hypothetical students assessed, only three scored High in this area, although several students scored in the Medium-High range. The 15 students who scored above Medium were from programs that have an overarching program focus on either a domain area or an application type. Nine of the hypothetical students scored Medium, and ten scored either Medium-Low or Low.

3.4 Depth

The Depth outcome deals with mastery of some component of the CBOK at the Bloom’s Synthesis level, which requires an individual to use existing ideas and software components to create new knowledge. Most programs had requirements for developing this type of in-depth knowledge in at least one area. Programs that do not require this type of in-depth analysis generally had students with lower achievement levels in Depth.

3.5 Ethics

The Ethics outcome requires an individual to clearly demonstrate an ability to make ethical professional decisions and to practice ethical behavior.
Although each program surveyed has an honor code applicable to their graduate programs and requires ethical conduct on all program work, few focus instruction time on professional ethics or decision-making. Despite this, as judged by the program faculty, over half of the hypothetical students would have attained this objective at a level of Medium or higher upon graduation.

Some faculty members indicated in follow-up conversation that their undergraduate SwE programs provide training in ethics and that they considered ethics a subject that should be taught and understood at the undergraduate level. As they expect students to enter with this knowledge and ability, they do not provide course materials specifically designed to address ethics issues associated with software engineering.

### 3.6 Sys Eng

The *Sys Eng* outcome requires a student to understand the relationship between software engineering (SwE) and systems engineering (SE) and to apply SE principles to SwE. Most universities indicated that this knowledge was gained tangentially in their programs, primarily through the examination of software lifecycles and embedded systems, which require an understanding of the greater operational system. Because they do not focus on the interaction of SwE and SE, however, most hypothetical students scored only Medium on this outcome. University E differentiates its program by the inclusion of SE principles, and two of its hypothetical students scored High. At least one other program has explicit courses requiring students to learn about SE.

From a GSwE2009 implementation perspective, this is an area where improvements are generally needed.
3.7 Team

The Team outcome requires a student to demonstrate the ability to work in a team setting, including the ability to lead in at least one area of project development. Development of appropriate SwE artifacts is also required.

Most participating programs require teamwork on projects in at least a few of their core courses. Program D is an exception – because most of its students are distance-education students, team-based project work is difficult to accomplish. Therefore, its entrance requirement of one year of industrial software engineering experience is intended, in part, to assure that a basic level of teamwork skill has been attained by the entering student. Several programs also require all students to participate in a large-scale team project or capstone project, which includes the development of traditional SwE artifacts. Overall, most students scored Medium or higher on the Team outcome.

The faculty from some of the programs claim that their mostly-industrial students do not need instruction or coursework on teamwork, because their students get their teamwork experience during the course of their day-to-day work activities.

3.8 Reconcile

The Reconcile outcome requires an individual to reconcile project objectives and find compromises within the limitations of a software system.

Some programs have a high number of software projects that require the student to perform activities requiring reconciliation of objectives, such as requirements generation, risk management and design tradeoffs; thus students in these programs generally had a High level of achievement for the Reconcile outcome. Other programs provide a more structured approach that does not require the students to work with conflicting project objectives to the same extent. For example, software
requirements are provided upfront and conflicts need not be resolved by the students. Students in programs using this latter approach generally scored Medium or Medium-High for the Reconcile outcome.

3.9 Perspective

The Perspective outcome requires an individual to understand and appreciate effective work habits, leadership, and communication with stakeholders in a software development environment.

In general, program faculty representatives indicated that their capstone experiences—typically moderate-scale software development projects requiring stakeholder interaction and project management—served to support this outcome. Programs not requiring a capstone experience, however, often required enough in-class projects that their students tend to satisfy this outcome at a Medium level.

Programs that do not require a capstone generally have students who attain lower achievement levels for this outcome. However, for many of these programs the students’ backgrounds provided them with the ability to achieve the Perspective outcome at a higher level. In general, it appears that professional/practical experience was the most consistent component in the student backgrounds when students attained high achievement of this outcome without a capstone experience. This suggests that one of the benefits of a strong experience entry requirement is that the Perspective outcome is more readily attained.

3.10 Learn

The Learn outcome deals with an individual’s understanding of and ability to accomplish professional development. Specifically, a graduate should be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.
There is a high degree of variability in achievement of the Learn outcome, although there does seem to be consistency within each program. Only three programs had students who achieved the Learn outcome at the High level. All of those students have practical experience greater than 2 years and/or participated in a capstone project. Experience in a professional setting may help students understand the importance of professional development and may require them to learn many new techniques, models, and technologies on the job. The capstone experience often requires that students perform additional work outside of the curriculum, which includes leaning new models, techniques, and technologies.

3.11 Tech

The Tech outcome requires an individual to be able to analyze a new or emerging software technology, including understanding its strengths and weaknesses, and to identify and promote opportunities to improve that technology.

Students in programs that require analysis of a software technology as part of their curriculum tended to score High on this outcome. Students that are required to perform a technology assessment that drives the choices of technologies used in their capstone projects also scored High on this outcome. Programs with students scoring in the Medium or Medium-High range did not explicitly require technology assessment as part of their courses. However, some of their core courses included projects that required some technology analysis, perhaps not as in-depth as measured by the outcome.

Figure 19. Student achievement of Tech outcome.
3.12 Findings and Conclusions based on Student Outcome Achievement

3.12.1 General Conclusions and Remarks

By the GSwE2009 standard, the programs examined here clearly do a reasonable job of satisfying the outcomes to a Medium level, at least for the “typical” students described. Two factors tend to bias things in their favor. For one thing, the universities that agreed to participate in this comparison offer graduate software engineering programs that, by and large, are among the better ones available today. Most have been around for some time and have established stakeholders, whom they serve at least adequately and often quite well. Few weak programs agreed to a comparison. A second factor that might bias the comparison is that the faculty representatives might have chosen hypothetical students who would be more likely to satisfy GSwE2009 outcomes than other students in their programs. Thus, truly typical students from truly typical programs might have lower outcome achievement than is reported here.

3.12.2 Conclusions Related to Hypothetical Student Achievement of Outcomes

The goal of GSwE2009 is to help raise the typical SwE master’s program to a higher standard that is more consistent with what employers and the SwE profession need. In order to do this the GSwE2009 CAT sought to develop a set of outcomes that went beyond the outcomes of current programs, but not so far that few programs would be able to achieve them.

The exercise of comparing current programs to the GSwE2009 outcomes provided some insight into this goal. In no outcomes did all students in all programs score either High or Low. For most students and most outcomes, achievement ran in the Medium-Low to Medium-High range. In addition, no single program’s students scored all Lows or all Highs. The participating programs’ hypothetical students collectively had a mid-level achievement of all outcomes, and individually each had some room for improvement to fully meet each GSwE2009 outcome. Therefore, the GSwE2009 outcomes as currently structured, if attained, would most likely raise the standard of master’s level SwE education, while still being attainable for the majority of programs.

Looking at the average achievement levels over all students, we find that all outcomes were fulfilled at levels between Medium-Low and Medium-High. When we focus on that narrow range (see Figure 20) we see that there is some variation across outcomes, with Ethics scoring lowest.

![Average Outcome Fulfillment](image)

Figure 20. Average fulfillment of each outcome, ordered from highest to lowest fulfillment (axis scale contains only M- to M+ to better show differences).
4 Profiles and Analyses of Selected Participating Programs

This section contains in-depth analysis of the student outcome fulfillments for three participating programs: University A, University D, and University H. These three programs were selected for detailed discussion because they had the most consistent and detailed data. In particular, their faculty representatives provided outcome attainment data upon entry as well as upon completion of the programs. The author team is still working to improve the quality of data from the other participating schools. Profiles from all 12 schools can be found in Appendix G.
4.1 University A Outcome Achievement

The students at University A achieve in the Medium to Medium-High range for almost all outcomes (see Figure 21). The only exception is hypothetical student 2, who received a Medium-Low score for the Ethics outcome. In the student profiles, University A listed similar educational experience for each student (a bachelor’s degree in a computer-science related field). The primary difference, then, appears to be in the level of practical experience. Both Students 1 and 3 have experience in industry prior to entering the program, while Student 2 has no practical experience and enters the program directly from an undergraduate education.

<table>
<thead>
<tr>
<th>Student</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>Student 2</td>
<td>Student 3</td>
<td></td>
</tr>
<tr>
<td>Student 1 has a bachelor’s degree in computer engineering and has been employed as a software engineer in the aviation industry for two years.</td>
<td>Student 2 is an international student from India, with a B.S. in Computer Science. The student has no industry experience.</td>
<td>The student has a bachelor’s degree in aerospace engineering with a minor in computer science. The student has been employed for one year as a programmer in a research center.</td>
<td></td>
</tr>
</tbody>
</table>

For the three hypothetical students in University A, the level of outcome fulfillment at the end of the program can be seen below in Figure 22.
Figure 22. Outcome achievement at entry and completion for University A students.
4.2 University D Outcome Achievement

Students in University D’s program achieve Medium or higher on all outcomes (see Figure 23). Achievement levels are closely coupled, with all three students reaching the same levels of achievement on almost all outcomes.

The minor differences in outcome achievement may be attributable to student backgrounds, which can be found in Table 7. Although the three hypothetical students focus in entirely different areas, their achievement levels are remarkably similar and relatively high, perhaps in part because this program offers a large selection of SwE courses and requires a minimum of seven SwE courses in each student's program.

The hypothetical students from program D had similar outcome achievement levels upon program entry, which helps to explain the relatively tight clusters of outcome achievement. For a comparison of outcome achievement upon entry versus achievement upon program completion for each student, please see Figure 24.

![Figure 23. Fulfillment of outcomes by University D students.](image)

Table 4. University D’s Hypothetical Student Profiles

<table>
<thead>
<tr>
<th>Student 1</th>
<th>Student 2</th>
<th>Student 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first student enters the master's program about a year after earning a Bachelor's Degree in Computer Science. He has no significant practical experience in software development other than what he gained as an undergraduate and what he has learned on the job in his first year of employment as a software developer in the telecommunications industry. He seeks to obtain a master's degree,</td>
<td>This student graduated from college with a degree in electrical engineering, went to work in the defense industry, and has been working for two years. During that time, she took courses in various computer science topics so she could move into a software engineering career, and she has project experience in software engineering. Her employer has established and sponsored an on-</td>
<td>This individual graduated with a math degree and entered the US Air Force, where he initially worked on algorithms for certain high performance flight applications. He migrated into broader software development activities and decided to take advantage of the opportunity offered by the Air Force to get a master’s degree in software engineering. In order to gain entrance, he had to take three junior</td>
</tr>
</tbody>
</table>

24
<table>
<thead>
<tr>
<th>Student 1</th>
<th>Student 2</th>
<th>Student 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>part-time, to further his career. At work he is a member of a software project team and over the course of his master's degree program, he is expected to have different work assignments within the software engineering field.</td>
<td>site, &quot;executive style&quot; master's program for software engineering. She has selected most electives in the field of information security.</td>
<td>college courses in basic computer science topics.</td>
</tr>
</tbody>
</table>

**Figure 24. Outcome achievement at entry and completion for University D students.**
4.3 University H Outcome Achievement

Students in University H’s program achieved Medium-Low to Medium-High on most outcomes (see Figure 25). The main exceptions are the Domain outcome, in which the program fosters lower achievement, and the Ethics and Reconcile outcomes, in which the program fosters higher achievement. Achievement levels for students 2 and 3 are generally closely coupled, with both students reaching the same levels of achievement on most outcomes.

The differences in outcome achievement may be attributable to student backgrounds, which can be found in Table 7. Student 1 is the only one of the students that has industrial experience in SwE. Students 2 and 3 have workforce experience, but have not been involved in software engineering efforts. Students 1 and 2 have undergraduate degrees in computer-intensive fields, while student 3 has an unrelated graduate degree.

For the CBOK outcome, student 3 has the lowest level of achievement. This is expected because student 3 has had less experience in the software engineering field. For the Depth outcome student 3 has a considerably lower achievement level than students 1 and 2. Again, this is most likely explained by the differences in abilities upon entry to the program.

In general, the hypothetical students 2 and 3 from program H had similar outcome achievement levels upon program entry, which helps to explain the relatively tight clusters of outcome achievement. In contrast, the outcome achievement upon entry is considerably different for student 1, which helps to explain why student 1 consistently achieves each outcome at a higher level than students 2 or 3. For a comparison of outcome achievement upon entry versus achievement upon program completion for each student, please see Figure 26.

University H offers a SwE specialty within a Computer Science program, whereas A and D offer SwE degrees. This may explain why H has lower outcome achievement in several areas.
Table 5. University H’s Hypothetical Student Profiles

<table>
<thead>
<tr>
<th>Student 1</th>
<th>Student 2</th>
<th>Student 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1 is a 30 year old male who has an undergraduate degree in computer science and has been working for a high tech company doing software development for more than 5 years.</td>
<td>Student 2 is a 28 year old woman who has her undergraduate degree in information technology.</td>
<td>Student 3 is a 50+ year old male who wants to change fields. He has a non-technical degree and has done some work programming, but not too much. He has had computers at home and messed around with them enough that he feels comfortable and thinks he knows enough to do a MSCS.</td>
</tr>
</tbody>
</table>

Figure 26. Outcome achievement at entry and completion for University H students.
4.4 Observations from Selected Program Comparisons

The data from the three selected programs clearly show certain characteristics of these programs. While three data points are not sufficient to draw conclusions, several observations are worthy of note.

1. **Observation:** These graduate programs clearly make substantial improvements in student outcome achievement, as shown by the “program influence on outcome achievement” charts.

2. **Observation:** Students with SwE experience in industry clearly do better in these programs.

3. **Observation:** The programs that offer full SwE degrees clearly do better in outcome achievement.

4.5 Observations and Issues from Overall Program Comparisons

Although data from the remaining nine programs are less complete, certain observations and issues can be noted based on examination of the data from all twelve (see Appendix G):

4. **Observation:** Similar entrance requirements (required degrees, levels of experience, etc.) do not always correspond to similar levels of outcome attainment, even when the students appear to have similar backgrounds. This suggests several hypotheses:

   a. The content and nature of the program has a significant effect on outcome achievement by graduates. (In other words, the data from the remaining nine programs tend to reinforce Observation 1)

   b. The individuals performing the comparisons were subjective in their assessments and varied somewhat in their application of the comparison rubric. (In other words, some rated themselves more highly than others with comparable circumstances, despite the CAT’s efforts to reduce this type of bias.)

   c. Background descriptions used in this study and experience requirements imposed by universities do not sufficiently reflect student experience, skill, and capability. While this may be a truism for any academic program, one might hope to find better ways of predicting student success.

**Issue:** Additional research will be required to determine whether any of these hypotheses are accurate and, in the case of the first, to identify which specific
program characteristics correlate to higher outcomes. There are some initial observations on the latter in what follows.

5. **Observation:** Individual programs vary from one another in the overall outcome attainment levels of their students. This suggests at least two hypotheses:
   a. Current SwE programs offer students significantly different academic experiences.
   b. Different programs attract different types of students, who achieve the outcomes at different levels.

**Issue:** Additional research will be required to explore this subject in more depth.

6. **Observation:** Within several of the programs, hypothetical students with similar backgrounds vary in their levels of outcome attainment. From a GSwE perspective, this implies the student’s choices of electives and tracks can make a significant difference in outcome attainment when programs offer a variety of options.

7. **Observation:** In most, but not all cases, students with industry experience attain notably higher levels of outcome achievement. This supports Observation 2. However, at least one program found that students lacking such experience can do well if suitable efforts are made to correct this deficiency and, thus, has recently removed its requirement for industry experience.

**Issue:** Experience clearly correlates to outcome attainment but there is insufficient evidence regarding what courses or other devices effectively compensate for lack of such experience.

8. **Observation:** The most commonly required (core) courses are software project management (7), software architecture and design (7), software requirements (7), and testing or verification and validation (6). Several programs (6) require broad-based software engineering or software process courses that cover multiple topics, including the first three mentioned above. Thus, virtually all programs have curricula that cover those three topics in some fashion or another. By contrast, relatively few require courses in construction (2), metrics (2), ethics (2) or systems engineering (1).

9. **Observation:** Programs that require courses in ethics or systems engineering tend to have higher outcome attainment, respectively, in Ethics and Sys Eng.

10. **Observation:** Although there was international representation on the GSwE2009 CAT, the data from three non-US programs suggest that there are significant differences of perspective, and that the GSwE2009 model is more US-centric than originally intended. Faculty from universities in South America and Australia reported difficulty applying the comparison process to their programs, and several
other international universities were unable to complete the comparison in time for this report.

**Issue:** Further research is needed to understand the differences in international graduate programs and to refine the GSwE comparison process appropriately.
5 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

5.1 Curriculum Development

[Bloom 1956]

[Fairley 1978a]

[Fairley 1978b]

[Freeman 1976]

[GSwE2009]

[SE2004]
[SEI 1989]

[SEI 1991]

[SWEBOK]

History and Overview of Graduate Software Engineering Programs

[Bagert 2005]

[Mead 2009]

[Pyster 2009]

[Tomayko 1998]

5.2 Establishment of New Graduate Software Engineering Programs

[Andrews University 1990]

[Boston University 1988]

[Carnegie-Mellon University 1990]

[Florida Atlantic University 1994]

[Frailey 1998]

[George Mason University 1988]

[National University 1990]

[University of Saint Thomas 1990]

[Seattle University 1978]

[Seattle University 1981]

[Southern Methodist University 1994]

[Texas Christian University 1978]

[Texas Christian University 1985]
5.3 Evolution of Graduate Software Engineering Programs

[Carnegie-Mellon 1995]

[Carnegie-Mellon 1996]

[George Mason 1994]

[University of Houston Clear-Lake 1995]

[Monmouth University 1987]

[Monmouth University 2003]

[Seattle University 1987]
Mills, E.E., “The Master of Software Engineering at Seattle University after Six Years,”

[Wang Institute 1987]

[Wang Institute 1988]

5.4 Collaboration between Universities and Industry in Software Engineering Education

[Beckman 1996]

[Ellis 2002]

[Frailey 2002]

[McCabe 1996]

5.5 Additional Sources of Information about Graduate Software Engineering Education

[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/>
6 GLOSSARY

6.1 Abbreviations and Codes

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT</td>
<td>Curriculum Author Team of GSwE2009</td>
</tr>
<tr>
<td>CBOK</td>
<td>Core Body of Knowledge; CBOK is also the code for the outcome of mastering the Core Body of Knowledge</td>
</tr>
<tr>
<td>CS</td>
<td>Computer Science</td>
</tr>
<tr>
<td>DEPTH</td>
<td>DEPTH 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.</td>
</tr>
<tr>
<td>DOMAIN</td>
<td>DOMAIN 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).</td>
</tr>
<tr>
<td>ETHICS</td>
<td>ETHICS is the code for the outcome that requires a student be able to make ethical professional decisions and practice ethical professional behavior.</td>
</tr>
<tr>
<td>GPA</td>
<td>Grade Point Average</td>
</tr>
<tr>
<td>GSwE</td>
<td>Graduate Software Engineering</td>
</tr>
<tr>
<td>GSwERC</td>
<td>Graduate Software Engineering Reference Curriculum, the name of earlier drafts of GSwE2009</td>
</tr>
<tr>
<td>GSwE2009</td>
<td>Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>INCOSE</td>
<td>International Council on Systems Engineering</td>
</tr>
</tbody>
</table>
iSSEc Integrated Software and Systems Engineering Curriculum project

IT Information Technology

KA Knowledge Area

LEARN LEARN 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 PERSPECTIVE 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 RECONCILE 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 Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering,
SwE       Software Engineering

SWEBOK   Software Engineering Body of Knowledge

SYS ENG  SYS ENG 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     TEAM 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     TECH 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
6.2 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 standalone) 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.
This page intentionally left blank.
Appendix A. Comparison Collaboration Process

GSwE2009 & University representatives Collaboration for Comparison Process

**Purpose:** This document spells out the process by which a member of the GSwE team collaborates with a University representative to conduct a comparison of their SwE program with the GSwE2009 guidelines.

**Related Documents:** Participating faculty will receive (1) a copy of GSwE2009, (2) the process instructions, and (3) an Excel template for data collection. The Excel template has 5 worksheets: Demographics, Students, Core, Electives, and Outcomes.

**Entry Criteria:**
- University representative has agreed to conduct the comparison.
- University representative has reviewed the GSwE guidelines.
- University representative has reviewed the GSwE comparison process.
- University representative has completed the basic demographic information table defined in step 5 of the comparison process, and sent it to GSwE representative.
- University representative has identified a single course that is going to be discussed during the introductory meeting.
- GSwE representative has obtained a description of the university’s software engineering program of study.
- GSwE representative has received a copy of the course syllabus for the course to be discussed during the introductory meeting.

**Exit Criteria:**
- Comparison is completed to the satisfaction of both representatives.
- Comparison is submitted to the GSwE group.

**Table 6. Activities requires for collaboration between participating faculty and the Curriculum Author Team (CAT).**

<table>
<thead>
<tr>
<th>Activities</th>
<th>Description</th>
</tr>
</thead>
</table>
| Pre-Meeting        | • University representative completes the Student and Core worksheets of the Excel template to the best of his/her ability for the individual course (not the combined totals).  
                      • GSwE representative reviews the submitted Excel file. |
| Introductory meeting | • **Purpose:** for the two representatives to get acquainted,  
                        • University representative briefly introduces their software engineering program.  
                        • GSwE representative briefly overviews the activities related to the comparison.  
                        • The two representatives walk through the draft Excel file submitted.  
                        • The two representatives walk through the process of comparison, by |
<table>
<thead>
<tr>
<th>Activities</th>
<th>Description</th>
</tr>
</thead>
</table>
| comparing and completing the table of outcome achievements for the identified individual course.  
- The two representatives briefly discuss the activities in Comparison Process Steps 5-9 for any clarification.  
- Next meeting time is identified.  
- Activities prior to the next meeting will be planned. | **NOTE: This initial activity should take less than two hours** |
| Conducting the comparison (at least the initial draft) | • **Purpose:** for the university representative to actually conduct the comparison, by following the steps defined in the GSwe comparison process.  
- In case there is a need, the University representative contacts the GSwe representative for assistance/guidance.  
- The completed comparison is submitted to the GSwe representative for review.  
- The GSwe representative evaluates the comparison, for completeness and accuracy.  
- Issues are identified for further discussion within the follow up teleconference.  
- Issues are communicated to the university representative (to be discussed during the next step). |
| Completing comparison and quality control | • **Purpose:** to review and complete the submitted comparison.  
- The two representatives walk through the initial comparison, and discuss issues of concern.  
- Through discussions, addition/modification/clarification to the initial comparison is conducted  
- Depending on how “good” the initial comparison was there may be a need for another iteration of the previous 2 activities. In this case the university representative will be asked to submit a second comparison. |
Appendix B. Comparison Process Instructions

This appendix contains information on the process provided to universities to create comparisons, which includes the rubric used to help standardize the comparisons.

GSwE2009

Process for Comparing Your Software Engineering Program with the GSwE2009 Model

The GSwE2009 model graduate curriculum for a terminal, professional master’s level program in software engineering is intended to assist faculty desiring to implement new software engineering programs at the graduate level or to revise existing programs. A first step in using the model curriculum is to compare one’s existing program (or, perhaps, a hypothetical program one is planning) with the GSwE2009 model. This document outlines a recommended procedure for performing such a comparison. The results of this procedure, if provided to the GSwE2009 team, are used (anonymously) to help improve its understanding of the range of existing programs and the similarities and differences between existing programs and GSwE2009.

1.0 Terminology

Certain terminology is used in this procedure to assure consistency in the comparisons. If a given situation does not appear to be covered by the terminology used, the GSwE2009 team should be consulted for assistance.

University – an academic institution that offers one or more graduate level programs in Software Engineering.

Academic Unit – an organization within a university that is responsible for the program in software engineering. For example, consider a college of engineering or a school of computing.

Academic Department – the organization within an academic unit that has specific responsibility for the program in software engineering. For example, consider a department of computer science and software engineering.
Program – a program of study leading to a master’s level degree in software engineering. For example, a software engineering master’s program or a software engineering track within a computer science master’s program.

Entry Requirements – a set of criteria that students must satisfy to be admitted to a program. For example, consider an undergraduate degree in computer science and one year of industrial experience. Note that in some cases, students may be conditionally admitted to a program provided they take certain courses or take other actions to satisfy the equivalent of the entry requirements.

Faculty – individuals who provide instruction in the software engineering program.

1.1 Measures

Several measured quantities are used in the comparison process. For the sake of consistency, we define some of these measures here.

- **Number of Students** in a program. This is measured by the number of program graduates per year. This measure was chosen so as to provide a consistent quantification of student body size regardless of whether students are full-time or part-time. For programs whose number of graduates fluctuates, reasonable averages should be used.

- **Number of Graduate SW Engineering Courses Taught per year.** This is another measure of the size of the program. This includes only courses for which credit is given toward the Software Engineering program and does not include leveling or articulation courses. If multiple sections of a course are taught each year, count each offering as a separate course.

- **Program Term Structure.** Please explain how your program terms are structured. The most common structures are semester and quarter systems. However, some schools use trimesters, or are on a semester system with additional summer sessions. Please provide information on how the academic calendar for your institution is ordered.

- **Number of Full-Time Software Engineering Faculty.** This is the number of full-time faculty whom the academic department associates primarily (over half time) with the graduate software engineering program. In cases where faculty are associated with multiple programs, you may use a pro-rata value. For example, suppose a department has 10 full-time faculty, of whom three are devoted primarily to the software engineering graduate program, two are devoted half time to the program and five are devoted full-time to a Computer Science program. Then the number of full-time software engineering faculty would be four.

- **Define Full-Time.** For your program, please provide the definition of “full-time” faculty. For example, a program may define full-time as teaching 8 courses per year. If your school has no official definition for ‘full time’, please assume six courses meeting for at least 150 minutes each week for 12-14 weeks and state your assumptions.

- **Number of Part-Time/Adjunct Program Faculty.** This is the number of part-time faculty who teach within the program, regardless of how many courses they teach. A Part time/adjunct faculty member typically only teaches classes, and either they work full time at another organization, or they have retired from their full time positions.
High Level Summary of Comparison Procedure

Using the accompanying spreadsheet, complete the following tasks:

1. Provide basic demographic information about the university, academic unit, academic department, students, faculty and program.
2. Identify expected background of entering students.
3. Identify the “core” courses which all students are required to take, regardless of background or educational goals.
4. Identify major paths through the program, if there are any options or “tracks”.
5. Characterize one or more hypothetical students representing the range of backgrounds and program paths. Should a student require additional courses (e.g., leveling courses) in order to meet the standard entry requirements, please also provide that information. (The nominal target for each program is to characterize 3 such students. However, programs with only one “track” and few electives might have only one.)
6. For each hypothetical student, estimate the level to which the student would have achieved the 10 GSwE2009 outcomes upon entry to the program (based on the expected background of that student upon admission to the program and completion of ‘leveling’ courses, if any). Use the rubric provided in Appendix C.
7. Estimate the level to which the student would have achieved the 10 GSwE2009 outcomes upon completion of the required core coursework. Note: this should be identical for each student. If, because of significant differences in student background or other factors, you do not judge that each student would have attained the same level upon completing the required core coursework, include a note to that effect so we can better understand the issue.
8. Identify the non-core courses that each hypothetical student would take in order to complete the program. (Some of these may be required for a student’s chosen “track” and others may be electives.)
9. For each student and non-core course, estimate the level to which the given student taking the given course would be expected to achieve the 10 GSwE2009 outcomes upon completion of the course (using the rubric provided in Appendix C). Note: it is likely that most individual courses will help a student attain only some of the outcomes and will build upon attainments achieved in earlier courses. Although you may need to consider how individual courses contribute to outcomes, please only record the total expected achievement from each of the non-core courses, assuming they have completed all core and prerequisite courses.
10. Combine the information from steps 6, 7, and 9 to determine, for each student, the level to which the total program, plus entrance criteria, would enable attainment of the 10 GSwE2009 outcomes.

The specific details of this process are found below. The most time consuming portion of this process is steps 6, 7 and 9, which often require consultation with other faculty to properly evaluate the full range of courses. However, since most students have courses in common,
the first student is usually the most difficult. A certain amount of judgment is required and, thus, the process is not highly precise. However, we have found that if carried out as described the process yields an effective assessment of how a program compares with the model.
Detailed Procedure

Use the accompanying spreadsheet to provide the information described below.

1. **Provide basic demographic information about the program.**

   The attached spreadsheet, first tab, requests demographic information about the university and the software engineering program. This and later tabs are used to document information from certain of the other steps in this procedure.

2. **Identify expected background of entering students.**

   One of the most important characteristics of graduate software engineering programs is the expectations or prerequisites imposed on entering students. Some programs, for example, may require an academic degree in a specific discipline or type of discipline (for example, undergraduate degree in a technical field). Some may require specific experience, such as a number of years with industrial software engineering experience. Other programs may require no specific entry criteria. The GSwE2009 model curriculum has specific entry criteria, so it is helpful to document how they compare with those of the curriculum being evaluated.

   Note that many programs have ways to assist students who lack the desired entry criteria. Sometimes these criteria may be waived, in which case the criteria for a waiver should be documented. In other cases, there are “leveling” or “articulation” courses available to help entering students attain knowledge or experience that they lack. Although such courses typically do not earn credit toward the software engineering degree, such is not always the case. Whatever a program’s specific approach to students who do not meet entry criteria, it is valuable to document that approach to facilitate the comparison.

3. **Identify the courses required for all students.**

   Most programs have several “core” courses which all students must take, regardless of background or educational goals. These “core” courses provide a common baseline for a program. Please provide a description of the courses which are required for all students, including the official course numbers and titles.

4. **Identify major paths through the program, if there are any options or “tracks”.**

   Some programs offer no choices or electives for software engineering students, while others offer several electives, and some may offer a significant number of choices, perhaps organized into distinct “tracks” or “options”. Comparisons with the GSwE2009 model are best done at this level – in other words, if a program offers several distinct options then it is desirable to compare each of them separately. It is not necessary to identify every conceivable option – the nominal goal is to have about three per program. However, it is acceptable to have only one or to have four or five. The idea is to see whether different paths through the program, if offered, result in significantly different comparison results.
5. Characterize one or more hypothetical students representing the range of backgrounds and program paths. (The nominal target for each program is to characterize 3 such students. However, programs with only one “track” and few electives might have only one.)

We have found that, as a rule, students with significantly different backgrounds and/or significantly different “tracks” or electives tend to show different overall results when compared with the GSwe2009 model. Thus this step of the process is to identify several hypothetical students who would differ significantly in terms of the background they would bring to the program and/or the courses they would take during the program. Each hypothetical student should be characterized in terms of their background, their academic and professional goals, and the courses they will take during the program. For example: Student A has an industrial background and takes the data communications track; student B has a purely academic background and takes the system software track; student C has an industrial background and takes the management track.

In addition, some programs allow students to be admitted who do not completely fulfill the entrance requirements. If such students are included in your hypothetical cases, please explain what courses the student would need to take in order to fully participate in the program. For some programs, this may include requirements such as an introductory course in software engineering and an internship or work study program to gain real-world experience.

6. For each hypothetical student, estimate the level to which the student would have achieved the 10 GSwe2009 outcomes upon entry to the program (based on the expected background of that student upon admission to the program). Use the rubric provided in Appendix C.

To illustrate, a student with an undergraduate degree in software engineering might be expected to have attained at least a moderate level of mastery of the core software engineering material (outcome 1 of GSwe2009). A student with 2 years of industrial software engineering experience might be expected to have attained some level of mastery in teamwork (outcome 6).

The abbreviated definitions of GSwe2009 Outcomes may be found in Appendix D of this document. The outcomes are fully defined in GSwe2009, as is the Core Body of Knowledge (Core BOK or CBOK). Should you require a copy of GSwe2009, please contact your GSwe2009 point of contact.

The result of this step might look as follows:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

52
7. Estimate the level to which the student would have achieved the 10 GSwE2009 outcomes upon completion of the core courses.

Estimate, using the rubric in Appendix C, the extent to which the core courses support achievement of the 10 outcomes, assuming all “leveling” courses have been completed. This level of achievement will normally be the same for each student. If, because of significant differences in student background or other factors, you do not judge that each student would have attained the same level upon completing the require core coursework, include a note to that effect so we can better understand the issue. Although you may need to consider how individual courses contribute to outcomes, please only record the total expected achievement upon completion of each of the core courses, assuming they have completed all leveling and prerequisite courses.

8. Identify the courses that each hypothetical student would take in order to complete the program.

Each course should be described by a paragraph as well as by a title. Note that there may be several courses that more than one of the students will take. Here is an example of how courses would be described in the spreadsheet:

<table>
<thead>
<tr>
<th>#</th>
<th>Course Name</th>
<th>Brief Course Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SWE 7001 Software Requirements</td>
<td>Defining and specifying software requirements that can be used as the basis for designing and testing software. Includes use-cases for describing system behavior, formal methods, specifying functional vs. nonfunctional requirements and the relationship of requirements to software testing.</td>
</tr>
<tr>
<td>2</td>
<td>SWE 7002 Software Architecture and Design</td>
<td>Explores the role of design in the software lifecycle including different approaches to design, design tradeoffs and the use of design patterns in modeling object-oriented solutions. Focuses on important aspects of a system’s architecture including the division of functions among system modules, synchronization, asynchronous and synchronous messaging, interfaces and the representation of shared information.</td>
</tr>
<tr>
<td>3</td>
<td>SWE 7005 Software Testing and Quality Assurance</td>
<td>The relationship of software testing to quality with an emphasis on testing techniques and the role of testing in the validation of system requirements. Includes module and unit testing, integration, code inspection, peer reviews, verification and validation, statistical testing methods, preventing and detecting errors, selecting and implementing project metrics and defining test plans and strategies that map to system requirements.</td>
</tr>
<tr>
<td>4</td>
<td>SWE 7009 Software Project Planning and Management</td>
<td>The issues associated with the successful management of a software development project. Includes planning, scheduling, tracing, cost and size, estimating, risk management, configuration, management quality, engineering and process improvement.</td>
</tr>
</tbody>
</table>

9. For each student, estimate the level to which such a student would be expected to attain the 10 GSwE2009 outcomes (using the rubric provided in Appendix C).

This should be done separately for each course. Also, if there is a core group of courses that all students must take, they should be grouped together. (Normally, if two or more students take the same course, the attainment expectations would be the same for each student. However, if other factors, such as student background, would result in differing attainment levels, this should be documented.)

Note: although you may need to consider how individual courses contribute to outcomes, please only record the total expected achievement from each of the non-core courses, assuming they have completed all core and prerequisite courses.

The result of this step might look something like this (for an individual student):
10. Combine the information from steps 4, 6, and 9 to determine, for each student, the level to which the total program, plus entrance criteria, would enable attainment of the 10 GSwE2009 outcomes.

This is typically determined by using the highest attainment level from steps 6, 7, and 9, although sometimes the cumulative effect of several courses may be higher than any individual course attains.

The result of this step might look like this (for an individual student):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>154 701 Software Engineering</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>154 702 Software Architecture and Engineering</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>154 705 Software Testing and Quality Assurance</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>154 709 IE Project Management</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>

| 154 701 Software Engineering | Medium | High | Low | Low | Medium | Medium | Low | Medium | Medium | Low |
| 154 702 Software Architecture and Engineering | Medium | High | Low | Low | Medium | Medium | Low | Low | Medium | High |
| 154 705 Software Testing and Quality Assurance | Medium | High | Low | Low | Medium | Medium | Medium | Medium | Medium | Medium |
| 154 706 IE Advanced Java | Medium | High | Low | Low | Low | Medium | Medium | Medium | Medium | Medium |
| 154 707 IE Distributed Java | Medium | High | Low | Low | Low | Low | Low | Medium | High | Medium |
| 154 708 IE XML | Medium | High | Low | Low | Low | Low | Low | Medium | High | Medium |
| 154 709 IE Advanced Software Engineering | Medium | High | Low | Low | Low | Low | Low | Medium | Medium | Medium |
| 154 710 IE Mobile and Web Design | Medium | High | Low | Low | Medium | Medium | Medium | Medium | Medium | Medium |
| Total Contributions to Outcomes | Low | Low | Medium | Medium | High | High | Medium | High | Medium | Medium | High |
Appendix C. Rubric for Comparison of Courses/Background with Outcomes

Additional information on the outcomes may be found in Appendix D. For detailed explanation, please see GSwE2009.

<table>
<thead>
<tr>
<th>[1] CBOK</th>
<th>Low</th>
<th>Low+/Medium-</th>
<th>Medium</th>
<th>Medium+/High-</th>
<th>High</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Students show mastery of a few CBOK knowledge areas but are missing many of the knowledge areas</td>
<td>Students show mastery of some CBOK areas but are missing knowledge in other areas</td>
<td>Students show good mastery of some CBOK knowledge areas but a lesser mastery of other areas</td>
<td>Students show good mastery of most CBOK knowledge areas</td>
<td>Students show good mastery of all CBOK knowledge areas</td>
<td>&quot;show&quot; is based on the program’s self-defined assessment of student learning</td>
</tr>
</tbody>
</table>

| [2] Application domain | Little mastery or little understanding of any application domain | Partial mastery or some understanding of one application domain | Elementary mastery and some understanding of one application domain | Mastery and understanding of at least one application domain | Very good mastery and understanding of at least one application domain | Typical application areas include physics, chemistry, finance, military, biology, etc. |

| [3] Synthesis level | Little ability to do synthesis in any CBOK area or sub-area | Some ability to do synthesis in one CBOK sub-area | Some ability to do synthesis in one CBOK area or several sub-areas | Good depth in one CBOK sub-area | Very good depth in one CBOK area or several sub-areas | "synthesis level" implies that the student can take existing ideas and combine those in new ways |

55
<table>
<thead>
<tr>
<th>[4] Ethics</th>
<th>Low</th>
<th>Low+/Medium-</th>
<th>Medium</th>
<th>Medium+/High-</th>
<th>High</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students behave ethically most of the time, but do not understand how to make ethical professional decisions</td>
<td>Students behave ethically, but sometimes do not demonstrate good decision making abilities for ethical decisions</td>
<td>Students behave ethically, but often have to guess at what is a good ethical professional decision. Students study accepted ethics standards such as the ACM/IEEE sw engineering code of ethics.</td>
<td>Students always behave ethically and usually make good ethical professional decisions. Students study accepted ethics standards such as the ACM/IEEE sw engineering code of ethics.</td>
<td>for &quot;demonstrating how to make good ethical professional decisions&quot; the student should be able to not only make the correct choice, but be able to justify that choice when faced with an ethical dilemma</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[5] SwE and SE</th>
<th>Low</th>
<th>Low+/Medium-</th>
<th>Medium</th>
<th>Medium+/High-</th>
<th>High</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students have no awareness of the relationship between system engineering and software engineering and have no idea how to apply the system engineering principles and practices to software engineering</td>
<td>Students have little or some awareness of the relationship between system engineering and software engineering but don't know how to apply the system engineering principles and practices to software engineering</td>
<td>Students have a good understanding of the relationship between system engineering and software engineering and can sometimes apply those system engineering principles and practices to software engineering</td>
<td>Students have a good understanding of the relationship between system engineering and software engineering and can apply those system engineering principles and practices to software engineering</td>
<td>Students need to learn to define and solve problems and participate in interdisciplinary work</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Low+/Medium-</td>
<td>Medium</td>
<td>Medium+/High-</td>
<td>High</td>
<td>Comments</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[6] Team</td>
<td>Students have never worked in a team that has any diversity</td>
<td>Students have worked with a team that has some diversity but have never shown any leadership</td>
<td>Students work well in diverse team settings and occasionally take a lead role</td>
<td>Students work well in diverse team settings and are comfortable leading in at least one area</td>
<td>Students do very well in diverse team settings and can easily lead in one or more areas</td>
<td></td>
</tr>
<tr>
<td>[7] Reconciliation</td>
<td>Rarely can students deal with conflicting project objectives.</td>
<td>Occasionally students can find a partial solution for conflicting project objectives but those solutions only take into account a few of the limitations</td>
<td>Students can sometimes find a solution for conflicting project objectives that takes into account some of the limitations</td>
<td>Students can often find a solution for conflicting project objectives that takes into account almost all of the limitations</td>
<td>Students are very good at finding a means of reconciling conflicting project objectives that take into account all of the limitations</td>
<td></td>
</tr>
<tr>
<td>[8] Negotiation</td>
<td>Students don't really know these items nor appreciate them, can identify some stakeholders but may not communicate well with them</td>
<td>Students know and appreciate some of the items, can identify some of the stakeholders but may not communicate well with them</td>
<td>Students know what all of the items are, but may not fully appreciate the importance of them; can identify stakeholders and know how to communicate with them</td>
<td>Students understand and appreciate all of the items; can identify key stakeholders and know how to communicate with them</td>
<td>Students are very good at all of the items and have excellent communication skills</td>
<td></td>
</tr>
<tr>
<td>[9] Professional development</td>
<td>Low</td>
<td>Low+/Medium-</td>
<td>Medium</td>
<td>Medium+/High-</td>
<td>High</td>
<td>Comments</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
<td>--------------</td>
<td>--------</td>
<td>--------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>Students have no interest in emerging developments in either field but may have some idea of where that could be found</td>
<td>Students have little interest in emerging developments in software engineering and no interest in systems engineering developments but have some idea of where the software development information may be found</td>
<td>Students are somewhat interested in emerging developments in software engineering but not systems engineering and may not know where to find information on each</td>
<td>Students are interested in emerging developments in the software and systems engineering fields but may not know where to find such information</td>
<td>Students are interested in emerging developments in the software and systems engineering fields and know where to find such information.</td>
<td>Students could be encouraged to join technical/professional societies and/or to take the CSDP or other professional certification exam.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[10] Analysis techniques</th>
<th>Low</th>
<th>Low+/Medium-</th>
<th>Medium</th>
<th>Medium+/High-</th>
<th>High</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students cannot conduct any analysis of a significant software product or problem, nor can they articulate an assessment. They cannot envision any improvements or extensions to the product.</td>
<td>Students cannot conduct a correct analysis of a significant software product or problem, but to some extent can articulate what they did assess. They cannot envision improvements or extensions to the product.</td>
<td>Students can conduct an analysis of a significant software product or problem, but may not be able to clearly articulate their assessment or may not be able to envision improvements or extensions to the product.</td>
<td>Students can conduct an analysis of a significant software product or problem and articulate their assessment and envision improvements or extensions to the product.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix D. Brief Description of GSwE2009 Outcomes

[1] Master the Core Body of Knowledge
The CBOK specifies a Bloom Taxonomy level for each included knowledge area, subarea, topic, and subtopic. Mastering the CBOK requires learning principles exemplified through practice. A graduating student will have demonstrated he or she can perform at the specified Bloom level, which ranges from knowledge (the lowest level) up through analysis (the fourth level). Such performance is the definition of mastery used herein.

The CBOK Knowledge Areas are:

A. Ethics and Professional Conduct
B. System Engineering
C. Requirements Engineering
D. Software Design
E. Software Construction
F. Testing
G. Software Maintenance
H. Configuration Management (CM)
I. Software Engineering Management
J. Software Engineering Process
K. Software Quality

For additional information on the CBOK, please see Appendix F.

[2] Be able to apply software engineering in one application domain, such as finance, medical, transportation, or telecommunications, and in one application type, such as real-time, embedded, safety-critical, or highly distributed systems. That ability to apply software engineering includes understanding how differences in domain and type manifest themselves in both the software itself and in their engineering, and includes understanding how to learn a new application domain or type.

[3] Have mastered at least one knowledge area or sub-area from the CBOK to at least the Bloom Synthesis level.

[4] Be able to make ethical professional decisions and practice ethical professional behavior.

[5] Understand the relationship between software engineering and systems engineering and be able to apply systems engineering principles and practices in the engineering of software.

[6] Be able to work effectively as part of a team, including teams that may be multinational and geographically distributed, to effectively communicate both orally and in writing, and to lead in one area of project development, such as project management, requirements analysis, architecture, construction, or quality assurance.

[7] Be able to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, risk, existing systems, and organizations.
[8] Understand and appreciate the importance of feasibility analysis, negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.

[9] Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.

[10] 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.
Appendix E. GSwE2009 Outcomes

GSwE2009 defines 10 outcomes which it recommends students should be able to achieve upon completion of a master’s degree in software engineering. A brief description of these outcomes can be found in Table 8 below. For more detail, please see *Graduate Software Engineering 2009: Curriculum Guidelines for Graduate Degree Programs in Software Engineering*.

Table 7. Abbreviated description of GSwE2009 outcomes.

<table>
<thead>
<tr>
<th>Code</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBOK</td>
<td>Master the CBOK.</td>
</tr>
<tr>
<td>Domain</td>
<td>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.</td>
</tr>
<tr>
<td>Depth</td>
<td>Master at least one KA or sub-area from the CBOK to at least the Bloom Synthesis level.</td>
</tr>
<tr>
<td>Ethics</td>
<td>Be able to make ethical professional decisions and practice ethical professional behavior.</td>
</tr>
<tr>
<td>Sys Eng</td>
<td>Understand and the relationship between SwE and SE and be able to apply SE principles and practices in the engineering of software.</td>
</tr>
<tr>
<td>Team</td>
<td>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.</td>
</tr>
<tr>
<td>Reconcile</td>
<td>Be able to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.</td>
</tr>
<tr>
<td>Perspective</td>
<td>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.</td>
</tr>
<tr>
<td>Learn</td>
<td>Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.</td>
</tr>
<tr>
<td>Tech</td>
<td>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.</td>
</tr>
</tbody>
</table>
Appendix F. GSwE2009 CBOK

Table 9 below provides a brief overview of the knowledge areas (KAs) which appear in the CBOK as well as the general subjects which are included under that knowledge area. The CBOK is defined in greater detail in GSwE2009.

Table 8. Abbreviated description of the GSwE2009 CBOK.

<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>Bloom Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A. Ethics and Professional Conduct</strong></td>
<td></td>
</tr>
<tr>
<td>1. Social, legal, and historical issues</td>
<td>C</td>
</tr>
<tr>
<td>2. Codes of ethics and professional conduct</td>
<td>C/AP</td>
</tr>
<tr>
<td>3. The nature and role of software engineering standards</td>
<td>C</td>
</tr>
<tr>
<td><strong>B. System Engineering</strong></td>
<td></td>
</tr>
<tr>
<td>1. Systems Engineering Concepts</td>
<td>C</td>
</tr>
<tr>
<td>2. System Engineering Life Cycle Management</td>
<td>C</td>
</tr>
<tr>
<td>3. Requirements</td>
<td>C/AP</td>
</tr>
<tr>
<td>4. System Design</td>
<td>C/AP</td>
</tr>
<tr>
<td>5. Integration and Verification</td>
<td>C</td>
</tr>
<tr>
<td>6. Transition and Validation</td>
<td>C</td>
</tr>
<tr>
<td>7. Operation, Maintenance and Support</td>
<td>C</td>
</tr>
<tr>
<td><strong>C. Requirements Engineering</strong></td>
<td></td>
</tr>
<tr>
<td>1. Fundamentals of Requirements Engineering</td>
<td>C/AP</td>
</tr>
<tr>
<td>2. Requirements Engineering Process</td>
<td>C</td>
</tr>
<tr>
<td>3. Initiation and Scope Definition</td>
<td>AP</td>
</tr>
<tr>
<td>4. Requirements Elicitation</td>
<td>AP</td>
</tr>
<tr>
<td>5. Requirements Analysis</td>
<td>AN</td>
</tr>
<tr>
<td>6. Requirements Specification</td>
<td>AP</td>
</tr>
<tr>
<td>7. Requirements Validation</td>
<td>AP</td>
</tr>
<tr>
<td>8. Practical Considerations</td>
<td>C/AP</td>
</tr>
<tr>
<td><strong>D. Software Design</strong></td>
<td></td>
</tr>
<tr>
<td>1. Software Design Fundamentals</td>
<td>C/AP</td>
</tr>
<tr>
<td>2. Key Issues in Software Design</td>
<td>AP</td>
</tr>
<tr>
<td>3. Software Structure and Architecture</td>
<td>AP/AN</td>
</tr>
<tr>
<td>4. Software Design Quality Analysis and Evaluation</td>
<td>AP</td>
</tr>
<tr>
<td>5. Software Design Notations</td>
<td>AP</td>
</tr>
<tr>
<td>6. Software Design Strategies and Methods</td>
<td>AP/AN</td>
</tr>
<tr>
<td>Knowledge Area</td>
<td>Bloom Level</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>E. Software Construction</strong></td>
<td></td>
</tr>
<tr>
<td>1. Software Construction Fundamentals</td>
<td>AP</td>
</tr>
<tr>
<td>2. Managing Construction</td>
<td>AP</td>
</tr>
<tr>
<td>3. Practical Considerations</td>
<td>AP</td>
</tr>
<tr>
<td><strong>F. Testing</strong></td>
<td></td>
</tr>
<tr>
<td>1. Testing Fundamentals</td>
<td>AP</td>
</tr>
<tr>
<td>System testing and software testing</td>
<td></td>
</tr>
<tr>
<td>Testing-related terminology</td>
<td></td>
</tr>
<tr>
<td>Key issues</td>
<td></td>
</tr>
<tr>
<td>Relationships of testing to other activities</td>
<td></td>
</tr>
<tr>
<td>2. Test Levels</td>
<td>AP</td>
</tr>
<tr>
<td>The target of the tests</td>
<td></td>
</tr>
<tr>
<td>Objectives of testing</td>
<td></td>
</tr>
<tr>
<td>Component testing</td>
<td></td>
</tr>
<tr>
<td>Integration testing</td>
<td></td>
</tr>
<tr>
<td>System testing</td>
<td></td>
</tr>
<tr>
<td>Acceptance testing</td>
<td></td>
</tr>
<tr>
<td>3. Testing Techniques</td>
<td>AP</td>
</tr>
<tr>
<td>Based on tester’s intuition and experience</td>
<td></td>
</tr>
<tr>
<td>Specification-based</td>
<td></td>
</tr>
<tr>
<td>Code-based</td>
<td></td>
</tr>
<tr>
<td>Fault-based</td>
<td></td>
</tr>
<tr>
<td>Usage-based</td>
<td></td>
</tr>
<tr>
<td>Based on nature of application</td>
<td></td>
</tr>
<tr>
<td>Selecting and combining techniques</td>
<td></td>
</tr>
<tr>
<td>4. Test-Related Measures</td>
<td>AP/AN</td>
</tr>
<tr>
<td>Evaluation of the program or system under test</td>
<td></td>
</tr>
<tr>
<td>Evaluation of the tests performed</td>
<td></td>
</tr>
<tr>
<td>5. Test process</td>
<td>C/AP</td>
</tr>
<tr>
<td>Management concerns</td>
<td></td>
</tr>
<tr>
<td>Test activities</td>
<td></td>
</tr>
<tr>
<td><strong>G. Software Maintenance</strong></td>
<td></td>
</tr>
<tr>
<td>1. Software Maintenance Fundamentals</td>
<td>C</td>
</tr>
<tr>
<td>Definitions and terminology</td>
<td></td>
</tr>
<tr>
<td>Nature of maintenance</td>
<td></td>
</tr>
<tr>
<td>Need for maintenance</td>
<td></td>
</tr>
<tr>
<td>Majority of maintenance costs</td>
<td></td>
</tr>
<tr>
<td>Evolution of software</td>
<td></td>
</tr>
<tr>
<td>Knowledge Area</td>
<td>Bloom Level</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Categories of maintenance</td>
<td></td>
</tr>
<tr>
<td>2. Key Issues in Software Maintenance</td>
<td>AP</td>
</tr>
<tr>
<td>Technical</td>
<td></td>
</tr>
<tr>
<td>- Limited understanding</td>
<td></td>
</tr>
<tr>
<td>3. Maintenance Process</td>
<td>AP</td>
</tr>
<tr>
<td>4. Techniques for Maintenance</td>
<td>AP</td>
</tr>
<tr>
<td>H. Configuration Management (CM)</td>
<td></td>
</tr>
<tr>
<td>1. Management of the CM Process</td>
<td>C/AP</td>
</tr>
<tr>
<td>2. Configuration Identification</td>
<td>AP</td>
</tr>
<tr>
<td>3. Configuration Control</td>
<td>AP</td>
</tr>
<tr>
<td>4. Configuration Status Accounting</td>
<td></td>
</tr>
<tr>
<td>5. Software Release Management and Delivery</td>
<td>AP</td>
</tr>
<tr>
<td>I. Software Engineering Management</td>
<td></td>
</tr>
<tr>
<td>1. Software Project Planning</td>
<td>AP</td>
</tr>
<tr>
<td>Project goals and objectives</td>
<td></td>
</tr>
<tr>
<td>Project policies and standards</td>
<td></td>
</tr>
<tr>
<td>Process planning</td>
<td></td>
</tr>
<tr>
<td>Project assumptions and forecasts</td>
<td></td>
</tr>
<tr>
<td>Project deliverables</td>
<td></td>
</tr>
<tr>
<td>Project staffing</td>
<td></td>
</tr>
<tr>
<td>Effort, schedule, and cost estimation</td>
<td></td>
</tr>
<tr>
<td>Resource allocation</td>
<td></td>
</tr>
<tr>
<td>Quality management</td>
<td></td>
</tr>
<tr>
<td>Project plan/budget development and management</td>
<td></td>
</tr>
<tr>
<td>2. Risk Management</td>
<td>AP</td>
</tr>
<tr>
<td>Risk management concepts</td>
<td></td>
</tr>
<tr>
<td>- Probability, impact</td>
<td></td>
</tr>
<tr>
<td>- Timeframe</td>
<td></td>
</tr>
<tr>
<td>Risk management process</td>
<td></td>
</tr>
<tr>
<td>- Frameworks, standards, and guidelines</td>
<td></td>
</tr>
<tr>
<td>- Risk identification, analysis and risk prioritization techniques</td>
<td></td>
</tr>
<tr>
<td>- Risk mitigation strategies</td>
<td></td>
</tr>
<tr>
<td>Risk management tools</td>
<td></td>
</tr>
<tr>
<td>- Earned value tracking</td>
<td></td>
</tr>
<tr>
<td>- Technical performance measurement</td>
<td></td>
</tr>
<tr>
<td>- Defect tracking and reporting</td>
<td></td>
</tr>
<tr>
<td>3. Software Project Organization and Enactment</td>
<td>AP</td>
</tr>
<tr>
<td>4. Review and Evaluation</td>
<td>C</td>
</tr>
<tr>
<td>Knowledge Area</td>
<td>Bloom Level</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>5. Closure</td>
<td>C</td>
</tr>
<tr>
<td>7. Engineering Economics</td>
<td>C</td>
</tr>
<tr>
<td>J. Software Engineering Process</td>
<td></td>
</tr>
<tr>
<td>1. Process Implementation and Change</td>
<td>C/AP</td>
</tr>
<tr>
<td>2. Process Definition</td>
<td>C</td>
</tr>
<tr>
<td>3. Process Assessment</td>
<td>AP</td>
</tr>
<tr>
<td>4. Product and Process Measurement</td>
<td>AP</td>
</tr>
<tr>
<td>K. Software Quality</td>
<td></td>
</tr>
<tr>
<td>1. Software Quality Fundamentals</td>
<td>AP</td>
</tr>
<tr>
<td>2. Software Quality Management Processes</td>
<td>AP</td>
</tr>
<tr>
<td>3. Verification and Validation (V&amp;V)</td>
<td>AP</td>
</tr>
</tbody>
</table>
Appendix G. Participating School Profiles (Anonymous)

This appendix contains a short profile of the universities and programs that submitted comparisons. As mentioned in the introduction, school names are not associated with the information submitted in order to preserve anonymity.

Each abbreviated profile includes:

1. a brief introduction, describing any unique characteristics of the school and program;
2. a brief overview of entrance requirements;
3. an overview of the program’s architecture; and
4. a brief description of the core curriculum.
G.1 University A Profile

University A is a private international university that serves the aerospace related disciplines in engineering, science, and business. The Masters of Software Engineering (MSE) program is a professional program that focuses on the development of highly reliable software subsystems that are embedded in real-time, safety-critical systems. University A emphasizes the role of defined software processes in the effective and efficient development and maintenance of software systems. The program’s core courses all involve team project work, where students work on the development of industrial strength software artifacts, such as project plans, system and software requirements specifications, architectural and module design specification, documented code, test plans, and software process definitions.

G.1.1 Entrance Requirements

University A requires the following for entrance into its MSE program:

- Bachelor’s degree in a computing or an engineering or scientific field with a minor in computing with a 3.0 of better grade point average (GPA)
- Prerequisite knowledge in discrete mathematics, data structures and algorithms, computing systems, and programming in at least one high-level language.
- Experience developing at least one modest-sized real software product or substantial modification to an existing product.

G.1.2 Program Architecture

University A’s program consists of 5 3-credit core courses (15 credits total), a required capstone experience applying theories, practices, and technologies studied in core courses (3 credits), and at least 3 3-credit electives (9 credits total), bringing the program requirement to 27 credit hours.

G.1.3 Program Core

The MSE program includes 5 core courses: Software Engineering Discipline, Software Project Management, Requirements Engineering, Object-Oriented Software Construction, and Software Systems Architecture and Design.
G.2 University B Profile

University B has offered an M.S. in SwE since 1986. They have multiple program tracks, which allow for students who have previously received a bachelor’s degree in software engineering, for a general MSSE degree, and for specialties in embedded software telecommunications, information systems, software organization management, and the management of software technology. Because University B has a high percentage of part-time students who also work in industry, classes are scheduled one day per week, in the early or later afternoon. Several of the courses require students to do small team projects in preparation for the capstone project.

G.2.1 Entrance Requirements

University B requires the following for entrance into its MSSE program:

- Coursework in: overview of SwE, object-oriented design, and software process.
- There is no specific bachelor’s requirement, though most students have either Computer Science or Computer Engineering degrees.
- Students who have not had software development experience must participate in a 2-semester team project course.

G.2.2 Program Architecture

University B’s MSSE program requires 5 3-credit core courses (15 credit hours total) for all students who do not already possess an undergraduate degree in SwE. A capstone is required only for students without software development experience. The remainder of the curriculum is filled with electives. A student will generally take 10 to 17 3-credit courses (30 to 51 credits).

G.2.3 Program Core

University B requires the following courses: Mathematical Foundations of Software Engineering; Software Systems Requirements; Software System Design; Software Verification, Validation, and Maintenance; and The Process of Engineering Software.
G.3 University C Profile

The master’s program in software engineering at University C is the only such program offered in its region. It includes delivery of courses to many countries, especially those in the Far East. This program is offered in a computer science department and students are allowed to switch degrees from software engineering to computer science even after starting classes. Prospective students who do not have CS or SwE backgrounds are required to take a set of electives. Graduation at University C is conferred on completing 10 3-credit semester courses.

G.3.1 Entrance Requirements

University C requires the following for entrance into its MSSE program:

- A bachelor’s degree with an overall GPA of 3.0 of higher.
- All students must take the Graduate Record Examinations (GRE).
- Prerequisite knowledge in: mathematics (calculus and calculus-based statistics); basic and advanced computer science (programming, machine organization, and PC and UNIX-based program development environment proficiency; data structures, software engineering, computing theory, computer architecture, operating system principles)

University C does not generally provide leveling courses for students who do not meet their acceptance criteria.

G.3.2 Program Architecture

University C has 9 required courses for all students and also requires a capstone experience (either a master’s thesis or a master’s project). Each student will generally take 2 electives in addition to the required courses. Students are encouraged to take electives which will better enable their capstone.

G.3.3 Program Core

Program C requires the following subject areas of all students: software system engineering, software engineering metrics, software requirements analysis and design, software engineering project management, software verification and validation, software architecture, secure software engineering, human-computer interface design, and research methodology.
G.4 University D Profile

University D offers a large number of SwE graduate courses, most of them available on-line as well as in a traditional classroom setting. The program is also offered in an “executive” format at local industry sites. Almost all students are part-time professionals working in various nearby industries and typically have an undergraduate degree in a computing discipline. Because of this most classroom offerings of the courses are taught in 3-hour sessions at night or on weekends. More than half of the SwE faculty members are adjuncts who carry many years of industrial experience. The graduation requirement at University D is 10 3-credit semester courses.

G.4.1 Entrance Requirements

University D requires the following for entrance into its MSSE program:

- A bachelor’s degree in a quantitative science, computer science, mathematics, or an engineering discipline.
- At least one year of professional experience in software development and/or maintenance.
- Prerequisite knowledge in: discrete computational structures, programming languages, data structures, computer architecture and networks, and object-oriented programming.

Students without an undergraduate degree in computer science receive conditional admission and must complete a leveling course with a “B” or better in order to continue into the program. This course does not count toward the degree. Generally, entrance criteria are not waived unless the student has extensive experience and background in software engineering.

G.4.2 Program Architecture

University D’s program consists of a mixture of four required and six elective courses. Three of the electives must be taken from a list of about a dozen SwE courses, whereas the other three may include courses in computer science or other related fields. There is no capstone project or thesis required for this program because such a high percentage of the students take the courses via distance education.

Average Number of Graduates (per year): 40
Average Number of Faculty:
  Full-time: 3
  Part-Time/Adjunct: 6
Type of Degree Offered: Master of Science in Software Engineering
G.4.3 Program Core

University D requires that all students take courses on the following subjects: software requirements generation and analysis; software architecture and design; software testing and quality assurance; and software project planning and management.
G.5 University E Profile

At University E, the SwE program is focused on quantitative analysis, simplification and trustworthiness of software systems, and integration with Systems Engineering. Most students work for defense industries or the U.S. Department of Defense with a background in science or engineering. Most of the graduate courses in SwE are offered in four formats: on campus semester, at company trimesters, web-based on-line, and on-site modular (week long lectures followed by a 10 week project and on-line exercises.). A student needs to take 10 courses of 3 credit units to finish the degree.

G.5.1 Entrance Requirements

University E requires the following for entrance into its MSSE program:

- A bachelor’s degree in any engineering or science field.
- Prerequisite knowledge in: computer science foundations, computer engineering foundations.

G.5.2 Program Architecture

University E’s program consists of five required courses, four elective courses and a capstone project.

G.5.3 Program Core

University E’s program requires that students take courses in fundamentals of software engineering; software cost and estimation metrics; requirements analysis and engineering; software architecture and design; and software testing, quality assurance and maintenance.
G.6 University F Profile

The SwE program at University F is comprehensive and applied. Most of the courses offered are foundational and focus on security. All faculty members are full-time; most of the students are also full-time, with an undergraduate degree in some computing discipline. Students with industrial experience do not need to take certain prerequisite courses. A student will normally take 10 to 12 3-unit semester courses to satisfy graduation requirements.

G.6.1 Entrance Requirements

University F requires the following for entrance into its MSSE program:

- A bachelor’s degree in computer science, software engineering, or electrical or computer engineering.
- Prerequisite knowledge: foundations of computer science and foundations of computer engineering.

For students who have no undergraduate degree in computing or engineering, leveling courses in computer science and computer engineering are required. The only exception is for students with two years or more of software field experience.

G.6.2 Program Architecture

University F’s program consists of a mixture of three required and five elective courses. There is no capstone project or thesis required for this program.

G.6.3 Program Core

University F’s program requires that all students take courses in software project management, software modeling and architecture, and software verification and validation.
G.7 University G Profile

The Master of Science in Software engineering at University G is a professional program that focuses on software engineering, organizational, and business skills needed to work at companies that are product-focused, which measure development time in weeks instead of years, and often have an entrepreneurial culture. The program has vibrant full-time and part-time programs with distributed teams of students from around the world. Most of the students have a computer science background or extensive industry experience. Every core course is team-based, project-based leading to a capstone project with an external client at the end of the degree. The graduation requirement is 12 3-credit semester courses.

G.7.1 Entrance Requirements

University G requires the following for entrance into its MSSE program:

- A bachelor’s degree
- Prerequisite knowledge includes principled software development.

University G does not waive any of its entrance criteria. Students not meeting these criteria prior to application are not selected for admission to the program. Previously, University G required two years of relevant industry experience, but removed the requirement after a few exceptions were allowed and those students performed well.

G.7.2 Program Architecture

Each student comes to recognize individual strengths and weaknesses and establishes targets for improvement during the rest of the program. The student also becomes familiar with University G’s learn-by-doing pedagogy and philosophy of effective team-based work, as well as a selection of foundational elements from computer science and software processes.

University G’s program is somewhat unique in that it offers two different tracks for students, and the track dictates not only the electives a student will take, but also the core courses. Students in one of the tracks will take a practicum as part of their required curriculum, while students in the other track instead have two additional required courses.

G.7.3 Program Core

All students in University G’s program are required to take courses in the foundations of software engineering, requirements engineering, and software architecture and design.
Students in the more technically focused track will also take courses on avoiding software project failures and metrics. This is the track that also requires students to complete a practicum, which will be a real-world business problem where the student works in a team and coordinates with an actual client.

Students in the more managerially focused track will take courses in developing metrics for software organizations and in managing projects and processes.
G.8 University H Profile

The Masters of Computer Science with a specialization in Software Engineering at University H is a program that has a computer science core followed by courses that provide a strong foundation in the software engineering discipline. Most of the students are non-traditional, working professionals, attending school part-time and at their company’s expense. The courses are offered in several formats: face-to-face, hybrid and online. All face-to-face courses meet in the evening or on the weekend. Courses include team projects, individual projects and an optional capstone experience. The graduation requirement consists of 11 4-credit (quarter hour) courses.

G.8.1 Entrance Requirements

University H requires the following for entrance into its MSCS-SE program:

- A bachelor’s degree.

- Prerequisite knowledge includes: foundational-level knowledge in mathematics and computer science; intermediate ability to program in an object-oriented programming language.

Students who have the required background but need to refresh their mathematics or computer science skills are highly encouraged to take refresher courses. Students who do not have the required background in mathematics, computer science, and the concentration or current object oriented programming skills will design an undergraduate foundation program in preparation for this degree. These preparatory courses do not count toward the student’s degree.

A unique attribute to University H’s program is that the school is open enrollment—students are advised to follow guidelines, but there is no strict enforcement.

G.8.2 Program Architecture

The curriculum for University H’s program is comprised of roughly 50% required courses and 50% elective courses. Although no capstone is technically required, the majority of students elect to take a capstone experience course.

G.8.3 Program Core

All students in University H’s program are required to take courses in computer networking, the management of software projects, the security foundations of computer systems, and database
systems. The University also stands out because all students are required to take a course in leadership and ethical decision-making.
G.9 University I Profile

The Master’s of Software engineering at University I is a hybrid program in which students complete a thesis that involves completion of a significant capstone-like project. A majority of the students complete the program through distance learning. Most of the students enter the program with an engineering background and all students have at least two years of work experience in an engineering discipline. The graduation requirement at University I is a minimum of twelve courses (at least 42 quarter-hour credits).

G.9.1 Entrance Requirements

University I requires the following for entrance into its MSSE program:

- A bachelor’s degree in computer science, computer engineering, or a related field.
- Prerequisite knowledge includes: calculus.
- Two years of software development or maintenance experience.

Admission may be granted to an applicant without a CS or related engineering bachelor's degree if the applicant can demonstrate the basic knowledge of computation and software through work experiences or other extramural studies.

G.9.2 Program Architecture

University I’s program consists of six required courses and seven elective courses. Included in the required courses is a thesis project. The first semester subjects for each student consist of underpinning subjects intended for students with little prior knowledge of information technology or software engineering.

G.9.3 Program Core

All students are required to take courses in software methodology, software engineering and management, software engineering formal methods, requirements engineering, and risk assessments. Thesis research is also required.
G.10 University J Profile

This is a non-US university with a software engineering specialty within an information technology degree, rather than a software engineering degree by title. This program is designed to enable students to achieve a comprehensive and greater understanding of information technology in specialized technical or management areas. The wide range of specializations allows students to tailor the program to satisfy their career development needs. It is essential to keep IT knowledge and skills up-to-date. This program provides students with an enhanced understanding of the business context and technical developments shaping contemporary information and communications technology and equips them to meet the challenges of working in the IT industry.

G.10.1 Entrance Requirements

University J requires the following for entrance into its MSSE program:

- A bachelor’s degree.

G.10.2 Program Architecture

University J requires that all students in the information technology program take eight pre-defined courses and a series of electives. Students typically take between four and eight electives, depending on their focus. No capstone project or thesis is required.

G.10.3 Program Core

University J’s program requires students to take courses in object-oriented systems development methodologies; LAN hardware and physical layer standards; information technology management in enterprises; database design and implementation; research preparation and information technology research methods; professional standards and ethical behavior; and project management.
G.11 University K Profile

This is a non-US university with a software engineering specialty within a master of informatics degree rather than a software engineering degree by title. This is an academic program that focuses on the research aspects of IS modeling, development and use in business support. Most of the students are part time, coming from a computer science or information systems background. The core courses involve research and development and a dissertation is required towards the end of the degree. The graduation requirement is 34 credits within 4 semesters.

G.11.1 Entrance Requirements

University K requires the following for entrance into the MI program:

- A bachelor’s degree in computer science or a related area such as engineering or IT.

G.11.2 Program Architecture

University K’s program requires that all students take eight semester courses in software engineering fundamentals and four to five elective courses in additional concepts. No capstone project or thesis is required for these courses. Some courses are lecture based while others are delivered in a laboratory setting.

G.11.3 Program Core

University K’s program requires students to take courses in: scientific methods; algorithm analysis, complexity theory, and non-deterministic algorithms; database concepts and modeling; basic concepts of software engineering, development, and quality assessment; information system modeling languages and techniques; data network concepts and services; advanced programming; and formal logic concepts.
G.12 University L Profile

This is a non-US university with a software engineering specialty within a systems engineering degree rather than simply a software engineering degree by title. University L offers a set of SwE graduate disciplines, which can be combined to compose the SwE graduate program. All disciplines are available in a traditional classroom setting. Most of the students are full-time; some are professionals working in various nearby industries. Typical educational backgrounds for University L students include engineering (in general), computer science, or information systems. All faculty members are full time, and most have strong collaboration with industry. The graduation requirement at University L is 8 3-credit semester courses and dissertation.

G.12.1 Entrance Requirements

University L requires the following for entrance into the MSECS program:

- A bachelor’s degree in a related area such as engineering, computer science, or informatics.

The selection process for University L includes curriculum vitae evaluation, written exams, including a specific exam for SE, and formal interviews.

G.12.2 Program Architecture

The architecture of University L’s program includes:

- Three standard required courses that all students take;
- A series of four courses of which all students must take two;
- Three to four elective courses.

A capstone experience is not required within University L’s program.

G.12.3 Program Core

All students in University L’s program are required to take courses in: software process and product quality; general software reuse concepts; and experimental software engineering, which includes planning, execution, and analysis.

All students must take at least two of the following options: basic concepts in data arrays and algorithms; general concepts of set theory; artificial intelligence and data mining; or database organization and information retrieval.