EMBRY-RIDDLE
Aeronautical University
SCHOLARLY COMMONS

Electrical, Computer, Software and Systems Engineering - Daytona Beach

College of Engineering

2013

Development of a Master of Software Assurance Reference Curriculum

Andrew J. Kornecki
Embry-Riddle Aeronautical University, kornecka@erau.edu

James McDonald

Julia H. Allen

Mark Ardis

Nancy Mead

See next page for additional authors

Scholarly Commons Citation

Kornecki, A. J., McDonald, J., Allen, J. H., Ardis, M., Mead, N., Linger, R., & Hilburn, T. B. (2013). Development of a Master of Software Assurance Reference Curriculum. , *2*(1). https://doi.org/10.3390/electronics2010041

Authors

Andrew J. Kornecki, James McDonald, Julia H. Allen, Mark Ardis, Nancy Mead, Richard Linger, and Thomas B. Hilburn

Software Engineering Institute | Carnegie Mellon University

Development of a Master of Software Assurance Reference Curriculum

Andrew J. Kornecki
James McDonald
Julia H. Allen
Mark Ardis
Nancy Mead
Richard Linger
Thomas B. Hilburn

February 2011

ABSTRACT: Modern society is deeply and irreversibly dependent on software systems of remarkable scope and complexity in areas that are essential for preserving our way of life. The security and correct functioning of these systems are vital. Recognizing these realities, the U. S. Department of Homeland Security (DHS) National Cyber Security Division (NCSD) enlisted the resources of the Software Engineering Institute at Carnegie Mellon University to develop a curriculum for a Master of Software Assurance degree program and define transition strategies for implementation. In this article, we present an overview of the Master of Software Assurance curriculum project, including its history, student prerequisites and outcomes, a core body of knowledge, and a curriculum architecture from which to create such a degree program. We also provide suggestions for implementing a Master of Software Assurance program.

DEVELOPMENT OF A MASTER OF SOFTWARE ASSURANCE REFERENCE CURRICULUM

Software has become the core component of modern products and services. It has enabled functionality, business operations, and control systems critical to our way of life. However, software's race to ubiquity has outpaced security advances commensurate with software's vital role in our society. Consequently, as our dependence on software and software-intensive systems grows, we find ourselves exposed to an increasing number of risks.

The complexity of software and software-intensive systems, for instance, poses inherent risk. It obscures the essential intent of the software, masks potentially harmful uses, precludes exhaustive testing, and introduces problems in the operation and maintenance of the software. This complexity, combined with the interdependence of the systems we rely on, also creates a weakest link syndrome: attackers need only take down the most vulnerable component to have far-reaching and damaging effects on the larger system. What's more, anywhere-to-anywhere interconnectivity makes the proliferation of malware easy and the identification of its source hard.

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

The rising number of vulnerabilities compounds risk and—gives attackers even more targets of opportunity—as shown by the rising number of incidents targeting software vulnerabilities [Bosworth, 2002].

In this environment, the threats are large and diverse, ranging from independent, unsophisticated, opportunistic hackers to the very technically competent intruders backed by organized crime [Anderson, 2008]. Malicious actors are increasingly acquiring information technology skills that allow them to launch attacks designed to steal information for financial gain, and to disrupt, deny access to, degrade, or destroy critical information and infrastructure systems. Technical sophistication is no longer a necessary requirement: increasingly sophisticated attack methods, thanks to the growing underground trade in productized attack tools, no longer require great technical savvy to execute.

Recognizing these realities, the U. S. Department of Homeland Security (DHS) National Cyber Security Division (NCSD) enlisted the resources of the Software Engineering Institute (SEI) at Carnegie Mellon University to develop a curriculum for a Master of Software Assurance degree program and define transition strategies for future implementation. For the purposes of this curriculum, the discipline of software assurance is targeted specifically to the security and correct functioning of software systems, whatever their origins, application domain, or operational environments.

As noted in our curriculum report, the need for a master's level program in this discipline has been growing for years [Mead, 2010a].

- At the Knowledge Transfer Network Workshop in Paris in March 2009, cybersecurity education was recognized as part of the information security, privacy, and assurance roadmap vision. Cybersecurity education was also identified as one of the workshop's lines of development [LSEC, 2009].
- A study by the nonpartisan Partnership for Public Service points out that "[President Obama's] success in combating these threats [to cybersecurity] and the safety of the nation will depend on implementing a comprehensive and coordinated strategy—a goal that must include building a vibrant, highly trained and dedicated cybersecurity workforce in this country." The report found that "The pipeline of new talent [with the skills to ensure the security of software systems] is inadequate. . . . only 40 percent of CIOs [chief information officers], CISOs [chief information security officers] and IT [information technology] hiring managers are satisfied or very satisfied with the quality of applicants applying for federal cybersecurity jobs, and only 30 percent are satisfied or very satisfied with the number of qualified candidates who are applying [PPS, 2009].

- The need for cybersecurity education was emphasized by The New York Times in quoting Dr. Nasir Memon, a professor at the Polytechnic Institute on New York University: “There is a huge demand, and a lot more schools have created programs, but to be honest, we’re still not producing enough students” [Drew, 2009].
- Carnegie Mellon University and CERT have been active in this area for years, particularly in the Survivability and Information Assurance (SIA) Curriculum and the Scholarship for Service program [CERT, 2007]. The SIA Curriculum has been provided to thousands of faculty members and other interested parties. The Federal Cyber Service Scholarship for Service program offers scholarships to applicants who attend an approved institution of higher learning and agree to work for several years in the cybersecurity area at U.S. government organizations after graduation. The popularity and growth of this program is an indicator of the pressing need for cybersecurity expertise [U.S. Office of Personnel Management, 2010].
- In discussions with industry and government representatives, we have found that the need for more capacity in cybersecurity continues to grow. Anecdotal feedback from the authors’ own students indicates that even a single course with a cybersecurity focus enhances their positioning in the job market. They felt they were made job offers they would not have received otherwise.
- Another aspect of the need for cybersecurity education occurs in educational institutions. Based on our collective experience in software engineering education, we know that it can be very difficult to start a new program or track from scratch, and we want to assist those organizations and faculty members that wish to undertake such an endeavor. Our objective is to support their needs, while recognizing that there are a variety of implementation strategies.

In this article, we will present an overview of the Master of Software Assurance curriculum project [MSwA2010], and highlight the Master of Software Assurance Reference Curriculum report and its history [Mead, et al., 2010a]. We define student prerequisites and outcomes, a core body of knowledge, and a curriculum architecture from which to create such a degree program, either as a standalone offering or as a track within existing software engineering and computer science master’s degree programs. We also provide suggestions for implementing a Master of Software Assurance program.

BACKGROUND

As is typical in a project of this nature, a good bit of time is spent deciding how to tackle the project. The team members all had expertise in software engineering. In addition, some had experience in curriculum design, software assurance, or both. However, many decisions had to be made at the outset to get the project off the ground. One of our challenges was to decide how we would operate as a team with members in geographically dispersed locations. Not all of the team members had worked together before, but we quickly coalesced into an effective unit. For the most part, we held weekly telecoms, and occasional face-to-face work sessions when we needed a concentrated block of time. This worked remarkably well.

At the outset, we needed to define software assurance, examine recent curriculum and body of knowledge efforts to see which ones would apply, identify the audience for our work, and highlight ways in which our work was unique.

One of our first tasks was to examine existing definitions of software assurance, select a candidate definition from the literature, and assess whether it met our needs. Initially we selected the definition of the Committee on National Security Systems, as this definition was in wide use and used by our Department of Homeland Security sponsor:

> Software assurance (SwA) is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner. [Committee on National Security Systems, 2009]

As we got further into the project, we found that the definition needed to be extended slightly for our purposes:

> Software assurance (SwA) is the application of technologies and processes to achieve a required level of confidence that software systems and services function in the intended manner, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures [Mead et al., 2010a].

The extended definition emphasizes the importance of both technologies and processes in software assurance, notes that computing capabilities may be acquired through services as well as new development, acknowledges the need for correct functionality, recognizes that security capabilities must be appropriate to the threat environment, and identifies recovery from intrusions and failures as an important capability for organizational continuity and survival.

After examining the earlier Master of Software Engineering curriculum documents [Ardis & Ford, 1989; Ford, 1991], we concluded that the Graduate Software Engineering 2009 [GSwE2009] Curriculum Guidelines for Graduate Degree Programs in Software Engineering [Pyster, 2009] was the most relevant recent curriculum work to build on. We also drew on work done by Carnegie Mellon University's Software Engineering Institute in support of the U.S. Department of Homeland Security Build Security In website [DHS, 2010a]. We found that both the Software Assurance Curriculum Body of Knowledge (SwACBK) [DHS, 2010b] and the SWEBOK [IEEE-CS, 2004] were relevant as well.

We then considered the audience, and quickly concluded that the primary audience for the MSwA2010 curriculum is faculty who are responsible for designing, developing, and maintaining graduate programs that have a focus on software assurance knowledge and practices. However, we expect that the document will be read by other educators and trainers with an interest in this area, as well as industry and government executives and practitioners.

Finally, we identified what was different about this curriculum compared to traditional software engineering and computer science programs. Areas of special emphasis and unique properties that distinguish this curriculum (shown in italics) from others are the following:

- software and services
- development and acquisition
- security and correct functionality
- software analytics
- system operations
- auditable evidence
- organizational continuity

We developed the curriculum intending that it would be for practitioners, not for researchers. We also documented some initial thoughts on undergraduate coursework in software assurance in a separate document [Mead et al., 2010b].

We envision that the MSwA2010 curriculum can be offered as an independent master's degree program in software assurance or as a track in a Master of Software Engineering (MSE) or a Master of Computer Science degree program. This article describes how it can be incorporated as a track in an MSE degree program if the software engineering program is based on the GSwE2009 recommendations. The independent master's degree program in software assurance assumes a student enters the program with an undergraduate degree in computer science [ACM & IEEE-CS, 2008], computer engineering [IEEE-CS & ACM, 2004], or software engineering [IEEE-CS & ACM, 2004] and supplements the content of those degrees with appropriate prerequisite materials. For students with other backgrounds, the program incorporates the necessary preparation in computer science and software engineering to allow them to study software assurance.

PROCESS USED TO DEVELOP MSWA2010 CURRICULUM CONTENT

We started out with a schedule for the MSwA curriculum work and a set of activities to be performed to arrive at the curriculum. Once we decided that the GSwE2009 document would be a primary source, we reviewed it to see what elements could be carried over or modified for the MSwA2010 curriculum, and where we would have to tackle unique aspects. As we proceeded with the work, we realized that we had touched on many different areas that were seemingly unrelated. We therefore decided that it was worthwhile to document the process we had used not only for our own benefit, but also for our readers and others undertaking a similar activity.

We used the following seven-step process to develop the software assurance curriculum topics, practices, knowledge units, outcomes, and core body of knowledge, with course descriptions as an eighth activity (see Table 1).

Table 1. Software assurance curriculum development

1. Develop project guidelines	We modified guidelines from the GSwE2009 report for the MSwA2010 curriculum, which significantly influenced the development of outcomes (step 6).
2. Identify and review sources	We reviewed 29 credible and reputable sources of software security practices in industry, government, and academia (at the graduate and undergraduate levels).
3. Define topics	We used the guide in [Allen, 2008] as the organizing structure for our review of sources in Step 2 and supplemented it with our experience. This activity resulted in nine topics.
4. Define SDLC practices and categories	We evaluated sources for the topics listed above to identify practices, which we grouped into four high level categories.

5. Solicit external feedback	We sought input, through a 3-page questionnaire, from representatives of our target audience—managers, practitioners, and educators.
6. Develop outcomes and core Body of Knowledge (BoK)	We identified curriculum outcomes, influenced by GSwE2009 and questionnaire responses. Each outcome is a knowledge area in the BoK.
7. Compare knowledge areas to practices	We performed a cursory gap analysis by comparing the BoK knowledge areas to the SDLC practices and categories
8. Develop course descriptions	We developed course descriptions for the 9 core courses in an MSwA program and the 7 courses that would be added to a GSwE degree program for a software assurance specialization.

PROPOSED OUTCOMES WHEN A STUDENT GRADUATES

We needed to focus on the proposed outcomes in order to drive the program content. The outcomes specify the knowledge, skills, and capabilities that graduates of an MSwA program can expect to have; correspondingly, they represent the minimum capabilities that should be expected of a software assurance professional when they complete a master's degree program. Our process was not sequential; rather, we iterated on the outcomes, knowledge areas, and lifecycle practices over the course of the project.

When we solicited external feedback (step 5 in our process), we found that the MSwA2010 curriculum was not necessarily a match for all software assurance positions. Some organizations were more concerned with the qualifications of entry-level programmers who had not completed a master's degree program. Others were concerned with hands-on systems administrators. This curriculum is not a panacea, but it should help to grow the pool of leadership talent in software assurance, in much the same way that graduates of a master of software engineering program can be expected to become leaders in software engineering.

The primary audience for the MSwA2010 project, graduate faculty, should be prepared to teach courses that achieve these outcomes, listed below. Software development and acquisition employers responsible for staffing technical leadership positions in software assurance and developing increased software assurance capabilities of their current employees should expect graduates of an MSwA program to be proficient in capabilities described in these outcomes. The seven outcomes are grouped into two main areas—assurance process and management and assurance product and technology. Their brief descriptions follow [Mead et al., 2010a].

Assurance Process and Management

Assurance across life cycles:

Graduates will have the ability to incorporate assurance technologies and methods into life-cycle processes and development models for new or evolutionary system development, and for system or service acquisition.

Risk management:

Graduates will have the ability to perform risk analysis, trade-off assessment, and prioritization of security measures.

Assurance assessment:

Graduates will have the ability to analyze and validate the effectiveness of assurance operations and create auditable evidence of security measures.

Assurance management:

Graduates will have the ability to make a business case for software assurance, lead assurance efforts, understand standards, comply with regulations, plan for business continuity, and keep current in security technologies.

Assurance Product and Technology

System security assurance:

Graduates will have the ability to incorporate effective security technologies and methods into new and existing systems.

System functionality assurance:

Graduates will have the ability to verify new and existing system functionality for conformance to requirements and absence of malicious content.

System operational assurance:

Graduates will have the ability to monitor and assess system operational security and respond to new threats.

CORE BODY OF KNOWLEDGE

The MSwA2010 core body of knowledge (BoK) is characterized by the set of software assurance practices that are required to support the MSwA2010 outcomes. All software assurance professionals must know these practices to perform their jobs effectively. The MSwA2010 BoK is structured into seven knowledge areas (corresponding to the seven outcomes), with each knowledge area subdivided into a set of knowledge units, as shown in Table 2. The information in the table is expanded on in the MSwA2010 document [Mead, et al., 2010a].

The knowledge areas are defined in terms of the Bloom cognitive levels [Bloom, 1956]. This taxonomy is often used by educators to set the level of educational and learning objectives required for students engaged in an education unit, course, or program. Bloom's levels used are

- Knowledge (K)
- Comprehension (C)
- Application (AP)
- Analysis (AN)
- Synthesis (S)

Since we were developing a curriculum for a master's degree program, the Bloom's levels ranged from C through AN.

Table 2. MSwA2010 core body of knowledge

Knowledge Area		Bloom Level
1. Assurance Across Life Cycles	1.1. Software Life-Cycle Processes	
	1.1.1. New development	C
	1.1.2. Integration, assembly, and deployment	C
	1.1.3. Operation and evolution	C
	1.1.4. Acquisition, supply, and service	C
	1.2. Software Assurance Processes and Practices	
	1.2.1. Process and practice assessment	AP

	1.2.2. Software assurance integration into SDLC phases	AP
2. Risk Management	2.1. Risk Management Concepts	
	2.1.1. Types and classification	C
	2.1.2. Probability, impact, severity	C
	2.1.3. Models, processes, metrics	C
	2.2. Risk Management Process	
	2.2.1. Identification	AP
	2.2.2. Analysis	AP
	2.2.3. Planning	AP
	2.2.4. Monitoring and management	AP
	2.3. Software Assurance Risk Management	
	2.3.1. Vulnerability and threat identification	AP
	2.3.2. Analysis of software assurance risks	AP
	2.3.3. Software assurance risk mitigation	AP
	2.3.4. Assessment of Software Assurance Processes and Practices	AP
3. Assurance Assessment	3.1. Assurance Assessment Concepts	
	3.1.1. Baseline level of assurance; allowable tolerances, if quantitative	AP
	3.1.2. Assessment methods	C
	3.2. Measurement for Assessing Assurance	
	3.2.1. Product and process measures by life-cycle phase	AP

	3.2.2. Other performance indicators that test for the baseline, by life-cycle phase	AP
	3.2.3. Measurement processes and frameworks	C
	3.2.4. Business survivability and operational continuity	AP
	3.3. Assurance Assessment Process (collect and report measures that demonstrate the baseline)	
	3.3.1. Comparison of selected measurements to the established baseline	AP
	3.3.2. Identification of out-of-tolerance variances	AP
4. Assurance Management	4.1. Making the Business Case for Assurance	
	4.1.1. Valuation and cost/benefit models, cost and loss avoidance, return on investment	AP
	4.1.2. Risk analysis	C
	4.1.3. Compliance justification	C
	4.1.4. Business impact/needs analysis	C
	4.2. Managing Assurance	
	4.2.1. Project management across the life cycle	C
	4.2.2. Integration of other knowledge units	AN
	4.3. Compliance Considerations for Assurance	
	4.3.1. Laws and regulations	C
	4.3.2. Standards	C
	4.3.3. Policies	C
5. System Security Assurance	5.1. For Newly Developed and Acquired Software for Diverse Applications	
	5.1.1. Security and safety aspect of computer-intensive critical infrastructure	K

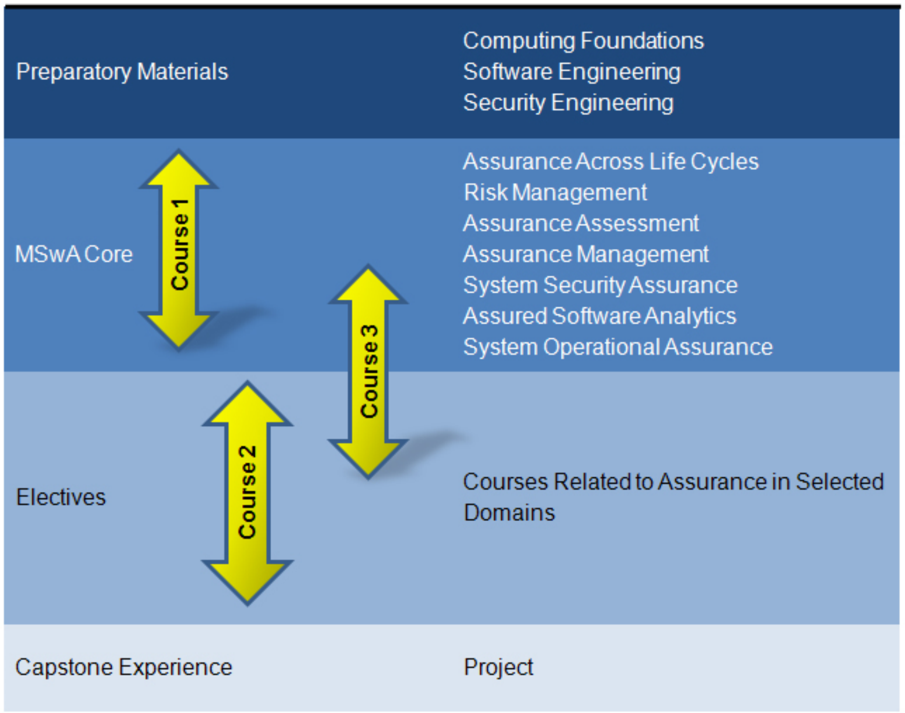
	5.1.2. Potential attack methods	C
	5.1.3. Analysis of threats to software	AP
	5.1.4. Methods of defense	AP
	5.2. For Diverse Operational (Existing) Systems	
	5.2.1. Historic and potential operational attack methods	C
	5.2.2. Analysis of threats to operational environments	AN
	5.2.3. Designing of and plan for access control, privileges, and authentication	AP
	5.2.4. Security methods for physical and personnel environments	AP
	5.3. Ethics and Integrity in Creation, Acquisition, and Operation of Software Systems	
	5.3.1. Overview of ethics, code of ethics, and legal constraints	C
	5.3.2. Computer attack case studies	C
6. System Functionality Assurance	6.1. Assurance Technology	
	6.1.1. Technology evaluation	AN
	6.1.2. Technology improvement	AP
	6.2. Assured Software Development	
	6.2.1. Development methods	AP
	6.2.2. Quality attributes	C
	6.2.3. Maintenance methods	AP
	6.3. Assured Software Analytics	
	6.3.1. Systems analysis	AP

	6.3.2. Structural analysis	AP
	6.3.3. Functional analysis	AP
	6.3.4. Analysis of methods and tools	C
	6.3.5. Testing for assurance	AN
	6.3.6. Assurance evidence	AP
	6.4. Assurance in Acquisition	
	6.4.1. Assurance of acquired software	AP
	6.4.2. Assurance of software services	AP
7. System Operational Assurance	7.1. Operational Procedures	
	7.1.1. Business objectives	C
	7.1.2. Assurance procedures	AP
	7.1.3. Assurance training	C
	7.2. Operational Monitoring	
	7.2.1. Monitoring technology	C
	7.2.2. Operational evaluation	AP
	7.2.3. Operational maintenance	AP
	7.2.4. Malware analysis	AP
	7.3. System Control	
	7.3.1. Responses to adverse events	AN
	7.3.2. Business survivability	AP

MSWA2010 CURRICULUM ARCHITECTURE

The MSwA2010 specifies an architectural description that provides a framework for organizing and structuring master's programs that focus on software assurance. The curriculum architecture, which was influenced by GSwE2009, contains the following components: preparatory material, core materials, elective materials, and a capstone experience.

Figure 1 depicts the architecture for an MSwA curriculum. The preparatory materials represent the material which students should master before entering the program. Individual programs will determine how to prepare students whose background falls short. The MSwA2010 outcomes and BoK identify the fundamental skills and knowledge that all graduates of a master's program in software assurance must possess. This is captured in the Figure 1 row labeled MSwA Core. Where appropriate, the core curriculum will emphasize the guidelines used to define the MSwA2010 BoK, including its dependencies on related disciplines such as software engineering, testing, and project management. Courses that cover core content should be part of all programs.



Preparatory Materials	Computing Foundations Software Engineering Security Engineering
MSwA Core	Assurance Across Life Cycles Risk Management Assurance Assessment Assurance Management System Security Assurance Assured Software Analytics System Operational Assurance
Electives	Courses Related to Assurance in Selected Domains
Capstone Experience	Project

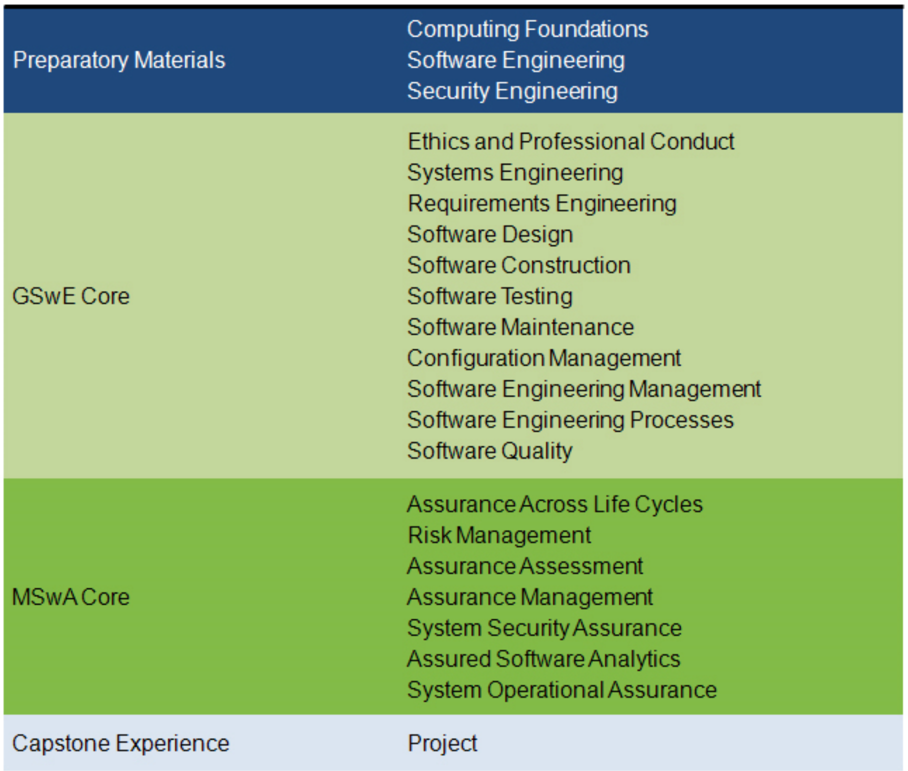
Figure 1

Electives accommodate individual students' interests and may cover unique requirements of a program or institution. Students may take electives to gain more depth in a core area (e.g., assurance assessment) or to extend and broaden their knowledge in a particular application domain (e.g., financial systems).

We recommend that students demonstrate their accumulated skills and knowledge in a capstone experience, which engages students in a realistic team project emphasizing software assurance concepts and practices. A capstone project is ideally a practical software assurance undertaking with a real customer, possessing actual software assurance objectives, and using best software assurance practices and tools. Students completing the curriculum must be able to understand and appreciate the skills needed to produce assured software in a typical software development environment. These topics should be integrated into the core materials and perhaps could be reinforced in the elective materials. However, the presence of a capstone project is important, as it offers students the opportunity to tackle a major project that is likely to be more comprehensive in realistic software assurance experience than their prior course projects.

This architecture is not intended to specify course titles, course content, or course sequencing, but rather to indicate the overall content in aggregate. Individual programs may choose the arrangement of courses, topics, and learning activities that best suit the needs and capabilities of their institutions.

Figure 2 illustrates a Master of Software Engineering program with a specialization in Software Assurance. As indicated in the figure, the core BoK includes knowledge areas from both the GSwE core and the MSwA core. Since there is overlap between the two BoKs (e.g., Software Engineering Management and Assurance Management), the required core content would be somewhat less than the sum of the two; however, the program would still be tight and would leave little or no room for electives.



Preparatory Materials	Computing Foundations Software Engineering Security Engineering
GSwE Core	Ethics and Professional Conduct Systems Engineering Requirements Engineering Software Design Software Construction Software Testing Software Maintenance Configuration Management Software Engineering Management Software Engineering Processes Software Quality
MSwA Core	Assurance Across Life Cycles Risk Management Assurance Assessment Assurance Management System Security Assurance Assured Software Analytics System Operational Assurance
Capstone Experience	Project

Figure 2

COURSE DESCRIPTIONS

Once we had the curriculum architecture and the body of knowledge, we were able to develop a sample set of course descriptions for MSwA as a standalone program, as well as courses that could be added to an MSwE program for a software assurance specialization. The knowledge units that each course should cover appear in parentheses by the course name.

MSwA Standalone Program (nine courses)

Assurance Management (2.1, 2.2, 2.3, 4.1, 4.2, 4.3)

Assurance Assessment (3.1, 3.2, 3.3, 6.4)[1]

System Operational Assurance (7.1, 7.2, 7.3)

[1] This course is not present in the MSwA Courses Added to the MSwE program.

System Security Assurance (5.1, 5.2, 5.3)

Assured Software Analytics (6.3)

Assured Software Development 1 (1.1, 1.2, 6.1, 6.2 [requirements])[2]

Assured Software Development 2 (6.1, 6.2 [specification, design])

Assured Software Development 3 (6.2 [code, test, verification, validation])

Software Assurance Capstone Experience

MSwA Courses Added to MSwE Program (seven courses)

Assurance Management (1.2, 2.1, 2.2, 2.3, 4.1, 4.2, 4.3)

System Operational Assurance (3.1, 3.2, 3.3, 6.4, 7.1, 7.2, 7.3)[3]

System Security Assurance (5.1, 5.2, 5.3)

Assured Software Analytics (6.3)

Assured Software Development 1 (1.1, 6.1, 6.2 [requirements, specification, design])

Assured Software Development 2 (6.2 [code, test, verification, validation])[4]

Software Assurance Capstone Experience

It is necessary but not sufficient to have a defined set of student prerequisites, established outcomes, a core body of knowledge, a curriculum architecture, and course descriptions. Often the most challenging part of putting a new program or a new track in place is implementation. The next section provides several guidelines and recommendations for faculty members to consider when contemplating such a program.

IMPLEMENTATION GUIDELINES

[2] The 1.2 knowledge unit, italicized, is different in Assured Development 1 in the standalone program and Assurance Management in the MSwA Courses Added to MSwE program.

[3] The bolded knowledge units are not covered at the same Bloom's level as in the standalone program.

[4] Condensed versions of Assured Software Development 1, 2, and 3 from the standalone program are in the MSwE program.

We realized that schools choosing to adopt our suggested curriculum would face several challenges besides deciding which topics to teach. In particular, schools would need to address

- planning and launching a new program
- recruiting and preparing students
- finding and training faculty
- acquiring resources
- capstone courses

For each of these issues, we offer some discussion of the problems and some advice for addressing those problems, drawing on our experience in starting similar programs. In addition, we found several good suggestions in the Frequently Asked Questions report published with GSwE2009 [Ardis, Lasfer, & Michael, 2009].

Planning and Launching a New Program

A prerequisite for starting any successful program is a champion who will lead the effort. This might be a faculty member, a department head, a dean, or another member of the academic community dedicated to starting the program. In addition, it helps to have other champions from industry and government who will support the program, perhaps by voicing support to others, hiring graduates, providing resources, and offering projects for the capstone experience. If possible, it is advisable to form an industry advisory board (IAB) early on to help support and shape the program.

The academic champion needs to make a convincing case for the program by preparing a business plan, including a market study. The plan should be used to convince university colleagues and administrators that there will be sufficient interest in the program, and that graduates will be successful in their career plans. Competing programs should be identified, some of which may be on the same campus.

New programs need to be sold at several levels of campus administration, and even at regional levels in some cases. For example, some states require extensive proposals for new academic programs, including details about courses, faculty, and dedicated resources. It is often much easier to get approval to create a new track within an existing program than it is to create a new program.

There are U.S. federal government assistance programs, such as the Scholarship for Service program that may help [U.S. Office of Personnel Management, 2010]. These programs provide some financial assistance to students and help justify the need for new academic programs. There are also U.S. federal agencies (for example, the National Science Foundation and the Department of Education) that provide start-up funds for innovative educational programs.

Recruiting and Preparing Students

If you build it, they may not come. Recruitment of students needs to be a continual process. A good market study should identify the likely areas from which to draw students. An IAB can help keep the study up to date and provide some additional help in recruiting.

Since some of the potential students are already in the workforce, it is helpful to establish relationships with the human resources (HR) departments of likely employers, including those that regularly recruit students from your institution. HR departments administer benefits, such as reimbursement for tuition, and often provide information to employees about educational opportunities. It may be possible to give in-house presentations to local companies, arranged through their HR departments or a member of your IAB.

Local professional organizations may provide opportunities for student recruitment. Trade organizations provide networking for local professionals and many of them have social events sponsored by local companies. There often are opportunities to give a short presentation or set up a booth at some of these meetings.

Most universities have professionals who help recruit students, but these individuals need to be informed about any new program and the types of students who best fit. Developing brochures and a web presence help to inform both internal staff and prospective students.

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

Finding and Training Faculty

There are two sources of faculty to teach in new programs of this type: (1) faculty from related areas who have knowledge and interest in teaching software assurance and (2) experienced practitioners from industry who are interested in teaching. The former often work in computer science academic units, but they may be found in almost any discipline that uses computing. Although they may have good teaching skills, they may need some help adjusting to the professional nature of the program. Some of their students will already have considerable experience and expect to learn about the latest methods and tools. It is important for faculty to stay current in the field. Consulting is one good way to do this.

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

It is prudent to ramp up faculty at a pace consistent with the growth of the program. This means that some part-time faculty will be needed early on before there is enough demand to justify hiring full-time faculty. Adjunct faculty from industry are often used for this, but also consider faculty from other academic units at your institution.

Acquiring Resources

Hardware and software may be provided by local companies or members of the IAB. In addition, some vendors have academic alliance programs that provide hardware or software at deep discounts. However, there should be an annual budget allocated to acquiring and maintaining computing systems. A small program should be able to share support staff with other programs.

Capstone Courses

Capstone courses in software assurance provide their own challenges. Fortunately, there are several models from which to choose. One issue to resolve early on is whether the capstone course(s) will be integrated with other courses in the curriculum. Integrated capstones provide connections to several other courses in the curriculum, offering opportunities for students to practice skills they learn in those courses. Standalone capstones are easier to implement because they do not have to be synchronized with the content of other courses.

To provide a realistic setting for a capstone course, it is helpful to have real clients. Finding clients is another recruiting activity to plan and implement each year. Another alternative is to pursue open source projects. The community of open source developers can play the role of clients, but they usually do not have the same level of commitment as a real client.

For more information about implementation considerations, consult the GSwE2009 FAQ Discussion Forum.[5] The Implementation/Execution forum[6] specifically addresses important issues for faculty members and institutions involved in implementing and executing a graduate program in software engineering. Many of these issues are the same for implementing an MSwA2010 degree program.

Ways in Which Industry Can Support Software Assurance Education

There are many ways that industry can provide support, from monetary assistance to participation in capstone projects. We describe several of these ideas as suggestions to readers from industry, and as advice to help new programs begin to make connections with potential industry sponsors.

For degree programs targeted toward professionals, such as the MSwA2010, industry support is essential. In addition to participating in industry advisory boards, making donations, or providing discounts on equipment and software, there are a number of other ways in which industry can contribute towards advancing this new discipline. These include

- encouraging employees to work with universities as adjunct faculty or guest lecturers—This can enrich both the industry organization and the university program.
- sponsoring and speaking at faculty development workshops—It's important to provide faculty development workshops for those who wish to teach a new discipline. However, the cost of such workshops can be significant. Industry could assist with the cost, help to shape the material, and provide guest speakers.
- providing grants to help develop new degree programs—Implementing new degree programs is very expensive, and assistance with some of the development costs could help get a new program off the ground.
- providing scholarships and summer internships to students in these programs—This is a good way to ensure that graduates can hit the ground running once they complete their degree program.

[5] http://www.gswe2009.org/faq/#cat5

[6] http://www.gswe2009.org/faq/?tx_mmforum_pi1[action]=list_topic&tx_mmforum_pi1[fid]=8

- providing support for realistic capstone projects—Industry could provide valuable support by proposing capstone problems, acting as a client, reviewing deliverables, and/or furnishing advice about project management, development methods, and technology.
- modifying and updating employee position descriptions to raise the bar—Many industry position descriptions focus on low-level skills, such as ability to code in C or Java and do not highlight the more advanced skills needed to produce assured software, such as background in risk analysis, attack patterns, threat modeling, and secure programming and testing.
- creating an endowed chair position in software assurance—An endowed position would ensure longevity for the program.

CONCLUSION AND FUTURE PLANS

The work described in this article can serve as a solid foundation for developing a master's degree program in software assurance. But developing the curriculum is just the first step in the set of activities needed to support Master of Software Assurance degree programs and tracks. To be successful, the curriculum model must be available, understood by the targeted academic and industrial communities, viewed as a key reference for software assurance curriculum development, and used to develop and modify software-assurance-focused curricula.

The process that we used worked well, in part because many of us had worked together in previous professional activities. However, there are certainly improvements that could be made. We did not plan as well as we could have for external review of the work. At times we had multiple authors making updates to the material, and coordination was sometimes a challenge. We spent a good bit of time on scope issues because we had not foreseen the need to clearly define the scope at the outset. On the plus side, we found that the diverse backgrounds among the authors allowed us to see different perspectives. We also found that there was benefit to starting outreach activities prior to the publication of the curriculum and presented it at several conferences and workshops.

During the coming year we will be involved in outreach activities. We plan to conduct faculty workshops and work with universities that may wish to adopt aspects of this curriculum. We will also extend our work to include considerations of software assurance specializations within other master's degree programs, such as Information Systems, and will further consider software assurance education needs at undergraduate levels and also in community colleges. We hope that this curriculum will be another step along the path of improving software assurance education and ultimately result in improvements in software systems assurance.

REFERENCES

[ACM & IEEE-CS, 2008]

ACM & IEEE-CS. (2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001. *Computing Curriculum Series.* Retrieved August 30, 2010 from http://www.acm.org//education/curricula/ComputerScience2008.pdf

[Allen, 2008]

Allen, J. H., et al. (2008). *Software security engineering: A guide for project managers*. Upper Saddle, NJ: Addison-Wesley Professional.

[Anderson, 2008]

Anderson, R. J. (2008).*Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition*. New York, N.Y. : John Wiley,

[Ardis & Ford, 1989]

Ardis, M., Ford, G. (1989). *1989 SEI Report on Graduate Software Engineering Education* (CMU/SEI-89-TR-21). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[Ardis, Lasfer, & Michael, 2009]

Ardis, M., Lasfer, K. & Michael, B. (Eds.). (2009). *Frequently asked questions on implementing GSwE2009.* Stevens Institute of Technology. Hoboken, N.J. Retrieved August 30, 2010 from http://www.gswe2009.org/faq/

[Bloom, 1956]

Bloom, B. S. (Ed.). (1956). *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. New York, N.Y. : Longman.

[Bosworth, 2002]

Bosworth, S. & Kabay, M.E. (Eds.). (2002). *Computer Security Handbook.* New York, N.Y.: John Wiley.

[CERT, 2007]

CERT. (2007). Survivability and Information Assurance Curriculum. *Software Engineering Institute, Carnegie Mellon University*, Retrieved October 4, 2007, from http://www.cert.org/sia/

[Committee on National Security Systems, 2009]

Committee on National Security Systems. (2009). *Instruction No. 4009, National Information Assurance Glossary*. Revised June 2009.

[DHS, 2010a]

Department of Homeland Security (DHS) (2010a). *Build Security In*. Retrieved August 30, 2010 from https://buildsecurityin.us-cert.gov/bsi/home.html

[DHS, 2010b]

Department of Homeland Security (DHS) (2010b). Software Assurance (SwA)Workforce Education and Training Working Group. *Software assurance CBK/principles organization.* Retrieved August 30, 2010 from https://buildsecurityin.us-cert.gov/swa/wetwg.html

[Drew, 2009]

Drew, C. Wanted: 'Cyber Ninjas.' *New York Times*. Retrieved December 29, 2009 from http://www.nytimes.com/2010/01/03/education/edlife/03cybersecurity.html?emc=eta1

[Ford, 1991]

Ford, G. (1991). *1991 SEI Report on Graduate Software Engineering Education* (CMU/SEI-91-TR-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[IEEE-CS, 2004]

IEEE-CS. (2004). IEEE Computer Society. *Software Engineering Body of Knowledge (SWEBOK).* Retrieved August 30, 2010 from http://www.computer.org/portal/web/swebok

[IEEE-CS & ACM, 2004]

IEEE-CS & ACM. (2004). Software engineering 2004: Curriculum guidelines for undergraduate degree programs in software engineering. *Computing curriculum series*. Retrieved August 30, 2010 from http://sites.computer.org/ccse/SE2004Volume.pdf

[LSEC, 2009]

Leaders in Security. (2009, March). *Building In ... Information Security, Privacy And Assurance*. Paper presented at the Knowledge Transfer Network Paris Information Security Workshop, Paris, France.

[Mead, et al., 2010a]

Mead, N. R., et al. (2010a). *Master of software assurance reference curriculum* (CMU/SEI-2010-TR-005/ESD-TR-2010-005). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[Mead, et al., 2010b]

Mead, N. R., et al. (2010b). *Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines* (CMU/SEI-2010-TR-019, ESC-TR-2010-019). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[Partnership for Public Service, 2009]

Partnership for Public Service & Booz Allen Hamilton. (2009). Cyber IN-Security: Strengthening the Federal Cybersecurity Workforce. *Partnership for Public Service.* Retrieved July, 2009, from http://ourpublicservice.org/OPS/publications/viewcontentdetails.php?id=135

[Pyster, 2009]

Pyster, A. (Ed.). (2009). *Graduate software engineering 2009 (GSwE2009) curriculum guidelines for graduate degree programs in software engineering, version 1.0.* Hoboken, NJ: Stevens Institute of Technology.

[U.S. Office of Personnel Management, 2010]

U.S. Office of Personnel Management. (2010). *Federal Cyber Service: Scholarship For Service*. Retrieved February 17, 2011, from https://www.sfs.opm.gov/

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Department of Homeland Security or the United States Department of Defense.

DM-0001120