Software Engineering Institute

Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines

Nancy R. Mead
Thomas J. Hilburn
Richard C. Linger

August 2010

TECHNICAL REPORT
CMU/SEI-2010-TR-019
ESC-TR-2010-019

CERT® Program

http://www.cert.org

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library).

Table of Contents

Acknowledgments

The authors thank the following individuals for their contributions to this report. We greatly appreciate their insights and efforts.

- Our sponsor Joe Jarzombek, U.S. Department of Homeland Security (DHS) National Cyber Security Division (NCSD), had the insight to recognize the need for such a curriculum and support its development.
- The DHS NCSD Workforce Education & Training Working Group provided valuable review comments on the draft document.

These individuals provided critical insights in their review of this document.

- Dick Fairley
- Dan Shoemaker, University of Detroit Mercy
- Carol Sledge, Software Engineering Institute

We also acknowledge the Department of Homeland Security's work on the Software Assurance Curriculum Body of Knowledge (SwACBK) and associated principles [DHS 2010b]. The SwACBK is foundational material for these course outlines.

In addition, we thank the following individuals from the Software Engineering Institute for their support: Jennifer Kent and Tracey Tamules.

Abstract

Modern society depends on software systems of ever-increasing scope and complexity. Virtually every sphere of human activity is impacted by these systems, from social interaction in our personal lives to business, energy, transportation, education, communication, government, and defense. Because the consequences of failure can be severe, dependable functionality and security are essential. As a result, software assurance is emerging as an important discipline for the development, acquisition, and operation of software systems and services that provide requisite levels of dependability and security. This report is the second volume in the Software Assurance Curriculum Project sponsored by the Department of Homeland Security. The first volume, the *Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005), presented a body of knowledge from which to create a Master of Software Assurance degree program, as both a standalone offering and as a track within existing software engineering and computer science master's degree programs. This report focuses on an undergraduate curriculum specialization for software assurance. The seven courses in this specialization are intended to provide students with fundamental skills for either entering the field directly or continuing with graduate-level education.

1 An Undergraduate Curriculum Focus on Software Assurance

Modern society depends on software systems of ever-increasing scope and complexity. Virtually every sphere of human activity is impacted by these systems, from personal lives and social interaction to business, energy, transportation, education, communication, government, and defense. Because the consequences of failure can be severe, dependable functionality and security are essential. As a result, software assurance is emerging as an important discipline for the development, acquisition, and operation of software systems and services that provide requisite levels of dependability and security. Software assurance (SwA) is defined as 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 2010].*

The U.S. Department of Homeland Security (DHS) has recognized the importance of this discipline for protecting national infrastructures and systems, and has pointed out the growing need for skilled practitioners in this area. At the direction of the DHS, the Software Engineering Institute (SEI) at Carnegie Mellon University developed the Software Assurance Curriculum Project. Volume I is the *Master of Software Assurance Reference Curriculum* (MSwA2010) [Mead 2010]. This report is Volume II, which focuses on an undergraduate curriculum specialization for software assurance. The foundation upon which this work rests includes the SEI's work on the DHS Build Security In website [DHS 2010a] and work by DHS on the Software Assurance Curriculum Body of Knowledge (SwACBK) [DHS 2010b].

The courses in this specialization are intended to provide students with fundamental skills for either entering the field directly or continuing with graduate-level education. Authors of the Software Assurance Curriculum Project include faculty and researchers from Carnegie Mellon University, Embry-Riddle Aeronautical University, Monmouth University, and Stevens Institute of Technology. In addition to basing the project on the body of knowledge for software assurance education, these course outlines are based on the authors' discussion and professional experience.

These course outlines support the recent Association for Computing Machinery (ACM)/IEEE Computer Society (IEEE-CS) *Computer Science Curriculum 2008 (CS2008)* guidelines [ACM 2008]. CS2008 includes the following comments related to software security:

- "The Review Task Force has received urgent requests from industrialists requesting that substantial attention to security matters be regarded as compulsory for all computing graduates."
- "The emergence of security as a major area of concern" is a recent trend.
- security curriculum concerns
  - "seen to be of increasing importance, to the extent that all graduates should have a general awareness of security issues"
  - "should embrace the issues associated with access, encryption, networking, etc."

- "importantly this should also include a general awareness about how to write safe and secure software—the latter was deemed to be very important"

MSwA2010 Curriculum Objectives

It is illuminating to parse the definition of software assurance into its constituent components as a guide to curriculum development. The specific MSwA2010 curriculum objectives are

- focus on both *software systems and services*

  Software capabilities can originate from many sources, including new system development; legacy system evolution; system acquisition through a variety of means, including supply chains, open source, and commercial, off-the-shelf (COTS); and service acquisition through methods including service-oriented architecture (SOA), cloud computing, and virtualization. Systems often aggregate combinations of these sources, all of which require a level of assurance with respect to correct functionality and security. In some cases, such as service acquisition, the software itself may not be available for analysis, and assurance must be achieved through other means. *Thus, the MSwA2010 curriculum must focus on both software systems and services in meeting assurance objectives.*

- software systems and services *function in the intended manner*

  Software systems and services must exhibit levels of quality and correct functionality commensurate with the consequences of their failure. Developing quality software requires rigorous software engineering capabilities and best practices in technologies and processes. Effective development, testing, and management skills are always required. Software assurance adds key perspectives and capabilities to development and acquisition processes to further improve quality. *Thus, the MSwA2010 curriculum must include technologies and processes to achieve correct functionality and reduce errors in software development and evolution, as well as in software and service acquisition.*

- software systems and services *are free from accidental or intentional vulnerabilities*

  In operational use, both legitimate users and intruders seeking to disrupt operations or obtain access to information use software systems. Intruders seek vulnerabilities in software they can use to gain access and control. Avoiding vulnerabilities (where possible) and eliminating vulnerabilities (where necessary) require thoroughly analyzing software and applying rigorous security requirements engineering, architecture and design, coding, and testing techniques. *Thus, the MSwA2010 curriculum must focus on the development of robust software systems and the acquisition of software services that do not provide a means to achieve unauthorized access and exploitation of vulnerabilities.*

- software systems and services *provide security capabilities appropriate to the threat environment*

  Software systems operate in threat environments whose virulence can vary with the value of the functions and information the systems provide. High-value systems will be subjected to sophisticated attacks at all levels and must incorporate security capabilities to ensure that intrusion is as difficult and costly as possible to the intruder. Virtually all systems must

implement security capabilities, such as authentication, authorization, availability, confidentiality, integrity, non-repudiation, and privacy. *Thus, the MSwA2010 curriculum must include threat environment analysis and security assurance technologies and methods at application, system, and network levels. The curriculum must also include methods for assuring security in the acquisition of software and services and for monitoring security in system operations.*

- software systems and services *recover from intrusions and failures*

  No amount of security and discipline can guarantee that systems will not be exploited and compromised. Operational continuity and survival must be assured even in adverse circumstances. *Thus, the MSwA2010 curriculum must include methods to define and assure that capabilities exist to recover from intrusions, failures, and accidents.*

These objectives are to be achieved through the following means:

- *application of technologies and processes*

  Assurance technologies include analytical areas such as verifying software functionality; analyzing software vulnerabilities, threat environments, and security capabilities; and reverse engineering software to determine as-built functionality and security properties. Assurance processes define methods for achieving required levels of confidence that can be integrated into traditional software development and acquisition process models. *Thus, in addition to a technology focus, the MSwA2010 curriculum must include a process-oriented view of assurance activities, including organizational goals, objectives, and constraints; risk analysis and reduction; and integration of assurance processes into organizational processes, methods, and procedures.*

- Achieve a *required level of confidence* that assurance goals are met

  A key responsibility of software assurance is to create auditable evidence that supports achievement of assurance goals. Assurance requirements can vary with business objectives, threat environments, system capabilities, risk analysis, legal and compliance requirements, and internal and external standards. *Thus, the MSwA2010 curriculum must provide methods for cost-effective and auditable assurance that satisfy organizational and technical objectives, requirements, and constraints.*

Course Descriptions

The course outlines in this report are part of the Software Assurance (SwA) Education work for the DHS. Although our primary focus in 2010 is on developing a reference curriculum for a Master of Software Assurance degree program, we recognize that software assurance education is relevant at all levels.

Our objective is not to suggest modifications to standard computing curricula guidance documents (for example, “The Computer Science Curriculum 2008” [ACM 2008]), although we believe emphasizing assurance topics in computing programs is a worthy goal. Computing curricula guidance has been a major activity of the professional societies, and we realize that major modification of current computing curricula would require an extended, labor-intensive effort.

Because such modification might take years, our objective in this report is to describe changes that universities can make in the near term to enhance software assurance education in their current curricula. Some programs might start with changing existing courses and adding a few of the recommended courses, and then over several years evolve the curriculum so that it includes all of the recommended courses. Our future work may include more comprehensive recommendations for software assurance concentrations in specific degree programs, such as computer engineering, computer science, software engineering, and information systems.

This report includes outlines for undergraduate courses in software assurance that could be offered in conjunction with a variety of programs. Some of the courses can be offered as standalone elective courses or as part of the curriculum core, whereas others might better be offered as part of an area of concentration in degree programs in computing (for example, computer engineering, computer science, software engineering, or information systems). The group of courses also includes two introductory courses in computer science, Computer Science I (CSI) and Computer Science II (CSII), which can provide an emphasis on software assurance in the first year of a curriculum. Our intent is not to offer a completely restructured CSI and CSII, but rather to propose a traditional CSI and CSII (based on *Computing Curricula 2001* [IEEE-CS 2001]) with modest changes to emphasize certain SwA (for example, secure coding) fundamentals. We believe these courses are appropriate for any CS major, because even if students do not pursue a curriculum with a SwA focus, they (and the community) will benefit from the changes we recommend to CSI and CSII. Some universities may have already modified CSI and CSII with a different distribution of assurance topics. We do not intend to be prescriptive; rather, we want to provide an example of how software assurance can be incorporated into a traditional program.

We provide descriptions of generic courses in this report. These descriptions include Bloom's taxonomy levels and general guidance on assessment (see Appendix A). We also provide syllabi of some specific existing courses in Appendix B.

We do not claim that this is an exhaustive list of courses, and there is no doubt that our outlines would need to be tailored to fit a particular university's degree programs. We envision that the courses outlined here will promote more attention to software assurance offerings at the undergraduate level.

- Introduction to Computer Security

  Provides an introduction to techniques for defending against hostile adversaries in modern computer systems and computer networks. We provide a course description that draws upon several existing course offerings.

- Software Security Engineering

  Covers a broad range of topics that are relevant and tailored to software security engineering, including properties of secure software, requirements engineering, architecture and design, coding and testing, system assembly, and governance and management. The course will include discussions of security properties, common vulnerabilities, threat environments, and dangerous practices.

- Secure Programming

  Teaches students how to avoid introducing common vulnerabilities into software. This is a popular topic that is taught at a number of universities at both the undergraduate and graduate levels, and also by training organizations. The courses tend to focus on specific languages and/or operating systems, and are often unique to the institution. We provide in Appendix B the syllabus, along with links to materials, for the Secure Programming course taught at Carnegie Mellon University.

- Special Topics in Information Assurance and Security

  Case studies can be effective for learning about security. A course that is designed around case studies can fit into a variety of curricula and would be an interesting elective. A half-semester course based on case studies is taught at Carnegie Mellon University. The syllabus for this course can be found in Appendix B.

- Software Quality Assurance

  Introduces quality processes and technologies for software development to assure that new software provides sufficient security for the threat environment and functions in the intended manner.

- Software Assurance Analytics

  Introduces analysis processes and technologies for assuring that legacy or acquired software (which is typically available) and services provide sufficient security for the threat environment and function in the intended manner.

- Software Assurance Capstone Project

  Encompasses development or modification of a significant software system, employing software assurance knowledge gained from courses throughout the program.

An Undergraduate Specialization

We provide one example of an undergraduate specialization in software assurance composed of five one-semester courses. Although we realize that it may be difficult to introduce five new courses into an existing curriculum, the prerequisites are part of most computing programs (programming fundamentals, computer architecture, operating systems, networks, and introduction to software engineering). The courses could either be used to replace upper-level required courses or become part of a set of specified electives. CS2008 proposes designing a curriculum around an organizing principle. One such organizing principle is to provide a focus on security: "Essentially security is a systems matter; for the system itself to be regarded as secure all major components typically need to be secure. . . .So this theme can be used as a unifying concept for the curriculum" [ACM 2008].

With programs that require an area of concentration, software assurance could become one of the specialization areas. The specialization is characterized by more thorough coverage than is customary in traditional software engineering programs of software quality, security throughout the life cycle, and operational security. The first two courses, Introduction to Computer Security and Secure Programming, focus on security threats and defenses. The next two courses, Software Quality Assurance and Software Assurance Analytics, focus on software quality and methods for

software reverse engineering and analysis. The Software Assurance Capstone Project course concentrates on software assurance processes and methodologies, and requires that students apply those processes and methodologies in a team project.

The course outlines in this report respond to requirements for knowledge and skills that are emerging from real-world organizations. These organizations are coping with the complexities of large-scale systems that carry high consequences of failure and compromise, and operate in ever-changing threat environments. Our task has been to aggregate and communicate these requirements in a program that, while sharing many foundations with mainstream computer science, adds and emphasizes subject matter that focuses specifically on software assurance. We are seeking to coalesce requirements for the emerging discipline of software assurance, while recognizing that these requirements can be met in a variety of ways.

Even if institutions are unable to offer the entire specialization of five courses, there are many opportunities to embed software assurance topics in existing courses. This approach permits introduction of the software assurance discipline and its subject matter in a stepwise manner to ease the cost and risk of start-up. For example, the software analytics subject matter overlaps with some topics traditionally covered in software maintenance courses, and many computing programs include a capstone course that could provide a software assurance focus. Another approach is to provide the Computer Science I and II courses described in this report, a subset of the courses outlined in this report, depending on faculty availability, and the capstone course with a software assurance focus.

2 Computer Science I

Course Description

This course introduces the fundamental concepts of procedural programming. Topics include data types, control structures, functions, arrays, and files. There is special emphasis on software assurance, covering topics in defensive programming, reviews, and unit testing. The course also offers an introduction to the historical and social context of computing and an overview of the computing disciplines.

Prerequisites

Sufficient facility with high school mathematics to solve simple linear equations and to appreciate the use of mathematical notation and formalism

Syllabus

Topic	Bloom's Level[1]
Computing domains: information systems, embedded systems; distributed and net-centric computing; computational computing; simulation and gaming	K
Fundamental programming constructs: syntax and semantics of a higher-level language; variables, types, expressions, and assignment; simple I/O; conditional and iterative control structures; functions and parameter passing; structured decomposition	AP
Algorithms and problem solving: problem-solving strategies; the role of algorithms in the problem-solving process; implementation strategies for algorithms; debugging strategies; the concept and properties of algorithms	C/AP
Fundamental data structures: primitive types; arrays; records; strings and string processing	C/AP
Machine-level representation of data: bits, bytes, and words; numeric data representation and number bases; representation of character data	AP
Introduction to language translation: comparison of interpreters and compilers; language translation phases; machine-dependent and machine-independent aspects of translation	C
Software assurance: foundations of information security; design concepts and principles; design by contract; exception handling; secure programming; coding standards; algorithm and code review; unit test design; penetration testing; program metrics; and quality assessment	C/AP
Overview of operating systems: the role and purpose of operating systems; simple file management	K
Tools and environments: integrated development environments (IDEs), libraries, testing and debugging tools	C/AP
Human-computer interaction: introduction to design issues	C
Social context of computing: history of computing and computers; professionalism, codes of ethics and responsible conduct; copyrights, intellectual property, and software privacy and piracy	K

[1] See Appendix A for a description of Bloom's levels.

Sources

IEEE Computer Society (IEEE-CS) & Association of Computing Machinery (ACM). *Computing Curricula 2001: Computer Science, Final Report* [IEEE-CS 2001].

ACM & IEEE-CS. "Computer Science Curriculum 2008: An Interim Revision of CS 2001." *Computing Curriculum Series* [ACM 2008].

Additional Items

Course Delivery Features

Besides conventional lecture/discussion methods, a number of hands-on individual and team projects are appropriate for course delivery; for example

- individual programming projects (including design documentation, coding using a coding standard, preparation of a unit test plan, test results)
- individual or team review/inspections—design, code, unit test plans
- team ethics case study exercises
- research papers and presentations of general computing topics

Course Assessment Features

Many of the course topics are listed at the AP (Application) Bloom's level, which means that students must be able to use information, methods, concepts, and theories to solve problems that require the skills or knowledge taught in the course. For topics labeled C (Comprehension), students must be able to discuss, describe, and interpret the topics. Assessment of the results of the activities and exercises discussed in the "Course Delivery Features" section is a good way of judging achievement at the specified Bloom's level.

3 Computer Science II

Course Description

This course introduces the concepts of object-oriented programming to students with a background in the procedural paradigm. The object-oriented programming paradigm in this course focuses on the definition and use of classes along with the fundamentals of object-oriented design. Other topics include a continuation of software assurance from Computer Science I and an overview of programming language principles, analysis of simple algorithms, and basic searching and sorting techniques.

Prerequisites

Computer Science I

Syllabus

Topic	Bloom's Level
Review of control structures, functions, and primitive data types	AP
Object-oriented programming: object-oriented design; encapsulation and information-hiding; separation of behavior and implementation; classes, subclasses, and inheritance; polymorphism; class hierarchies	C/AP
Software assurance: quality attributes; software processes; inspections and reviews; assurance coding standards and practices; measurement and metrics; quality assessment	C/AP
Fundamental computing algorithms: simple searching and sorting algorithms (linear and binary search, selection and insertion sort)	AP
Fundamentals of event-driven programming	C/AP
Introduction to computer graphics: using a simple graphics application program interface (API)	C
Overview of programming languages: history of programming languages; brief survey of programming paradigms	K
Virtual machines: the concept of a virtual machine; hierarchy of virtual machines; intermediate languages	C
Introduction to database systems: history and motivation for database systems; use of a database query language	C

Sources

ACM & IEEE-CS. "Computer Science Curriculum 2008: An Interim Revision of CS 2001." *Computing Curriculum Series* [ACM 2008].

IEEE-CS & ACM. *Computing Curricula 2001: Computer Science, Final Report* [IEEE-CS 2001].

Additional Items

Course Delivery Features

In addition to conventional lecture and discussion methods, a number of hands-on individual and team projects are appropriate for course delivery; for example

- individual/team programming projects (including detail and class design documentation, coding using a coding standard, preparation of a unit test plan, and test results)

- individual or team review/inspections—design, code, unit test plans
- team ethics case study exercises (reinforcing ethics topic in CSI)
- research papers and presentations of general computing topics

Course Assessment Features

Many of the course topics are listed at the AP (Application) Bloom's level, which means that students must be able to use information, methods, concepts, and theories to solve problems that require the skills or knowledge taught in the course. For topics labeled C (Comprehension), students must be able to discuss, describe, and interpret the topics. Assessment of the results of the activities and exercises discussed in the "Course Delivery Features" section is a good way of judging achievement at the specified Bloom's level.

4 Introduction to Computer Security

Course Description

This course provides an overview of the fundamentals of computer security. Topics include security standards, policies, and best practices; principles, mechanisms, and implementation of computer security and data protection; security policy, encryption, and authentication; access control and integrity models and mechanisms; network security; secure systems; programming and vulnerabilities analysis; principles of ethical and professional behavior; regulatory compliance and legal issues; information assurance; risk management and threat assessment; business continuity and disaster recovery planning; and security across the life cycle (requirements, architecture and design, construction, testing, operation, maintenance, acquisition, and services).

Prerequisites

Computer Science II or an equivalent course

Syllabus

Topic	Bloom's Level
Security goals and fundamentals: confidentiality, integrity, availability, reliability, etc.	K
Secure systems: types, models, design, changes to non-secure systems; comparative analysis	C
Access controls: controlling access to resources, access matrix model, access control lists and capability lists; mandatory controls, originator controls	C
Networks and security: internet security architecture, analysis of internet protocols, design and implementation considerations; firewalls	C
Integrity: cryptographic checksums, malicious logic, viruses, Trojan horses; defenses, prevention	C
Cryptography fundamentals: classical, public key; implementation, problems	C
Authentication, passwords, introduction to protocols	C
Attacks: software attacks (malicious code, buffer overflows, social engineering, injection attacks, and related defense tools); network attacks (denial of service, flooding, sniffing and traffic redirection, defense tools and strategies); website attacks (cross-site scripting)	C
Management: planning for security; introduction to risk assessment and management; business cases; regulatory compliance and legal issues; Federal Information Security Management Act; and business continuity/disaster planning	K
Security standards in government and industry: Common Criteria/Orange Book, sample corporate and institutional security policies	K
Security issues in requirements, architecture, design, implementation, testing, operation, maintenance, acquisition, and services	C
Ethics and professionalism as related to computer security	K

Sources

ACM & IEEE-CS. "Computer Science Curriculum 2008: An Interim Revision of CS 2001." *Computing Curriculum Series* [ACM 2008].

Allen, Julia H.; Barnum, Sean; Ellison, Robert J.; McGraw, Gary; & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers* [Allen 2008].

Bishop, Matt. *Computer Security: Art and Science* [Bishop 2002].

Redwine, Samuel T., Jr. *Secure Software Engineering Education.* This is a description of the first three years of the master's program in secure software engineering at James Madison University, including a somewhat detailed description of a one-semester course in secure software engineering [Redwine 2004].

Stallings, W. *Network Security Essentials* [Stallings 2007].

Wright, Marie & Kakalik, John. *Information Security: Contemporary Cases* [Wright 2007].

Note: To develop this course outline, we analyzed a number of existing and proposed courses that are designed to introduce security issues into a computing curriculum, including

- CS2008—Introduction to Computer Security outline [ACM 2008]
- Carnegie Mellon University outlines [CMU 2010a, CMU 2010b]
- James Madison University outlines [Redwine 2004]
- University of California, Davis outline [UC Davis 2010]
- Purdue University outline [Purdue 2010]

Additional Items

Course Delivery Features

In addition to conventional lecture and discussion methods, the following techniques are appropriate for course delivery:

- This course provides an excellent opportunity to use case studies. The Wright and Kakalik source *Information Security: Contemporary Cases* provides a source for reading, study, and case-study exercises.
- A number of hands-on individual and team projects could be assigned; for example
  - a requirements or architecture exercise involving quality attributes related to security
  - an exercise involving the review of program code to identify security problems
  - analysis of the security shortcomings of an existing software artifact or a computing system (standalone application, network, operating system, website)
  - research paper or presentation on current security technology or issue

Course Assessment Features

Most of the course topics are listed at the C (Comprehensive) Bloom's level, which means that students must understand course material beyond a simple recall level. They must be able to discuss, describe, and interpret course security topics at a level that shows insight and appreciation of computer security issues. Assessment of the results of the activities and exercises discussed in the "Course Delivery Features" section is a good way of judging achievement at the specified Bloom's level.

5 Software Security Engineering

Course Description

This course covers a range of topics that are relevant and tailored to software security engineering, including properties of secure software, requirements engineering, architecture and design, construction and testing, system integration/assembly, and governance and management. A summary of key practices and guidance on how to get started is provided. These are largely based on and inspired by material from the DHS Build Security In website [DHS 2010a].

Prerequisites

Software engineering undergraduate course

Syllabus

Topic	Bloom's Level
Why is security a software issue?: understanding the problem (threats, sources, assurance versus security), detecting software defects early, introduction to key practices	K
What makes software secure?: properties of secure software, defender and attacker perspectives, attack patterns, introduction to assurance evidence	K
Security of web applications: consideration of network-level attacks, cross-site scripting, SQL injection	C
Requirements engineering for secure software: importance of requirements engineering, quality requirements, security requirements engineering, Security Quality Requirements Engineering (SQUARE) introduction,[2] SQUARE case studies, SQUARE extensions, technology transition	AP
Secure software architecture and design: architectural risk analysis activities (including application of security principles and guidelines)[3]	AP
Considerations for secure coding and testing: introduction to practices (code analysis, code review, coding), software versus software security testing, security testing methods/techniques, testing throughout the software development life cycle (SDLC)	AP
Security and complexity—system development challenges: security failures, perspectives for security analysis, complexity	K
Governing and managing for more secure software: definitions and characteristics, risk management framework, project management—security in the SDLC	C
Getting started: determining where and how to begin, summary of key practices	K

Source

Allen, Julia H.; Barnum, Sean; Ellison, Robert J.; McGraw, Gary; & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers* [Allen 2008].

Additional References

McGraw, Gary. *Software Security: Building Security In* [McGraw 2006].

[2] See the CERT website for more information about SQUARE [CERT 2010a].

[3] Examples include the Architectural Tradeoff Analysis Method (ATAM) and Quality Attribute Workshop (QAW). Learn more at http://www.sei.cmu.edu/library/abstracts/reports/03tr016.cfm?WT.DCSext.abstractsource=RelatedLinks.

Howard, Michael & Lipner, Steve. *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software* [Howard 2006].

Wysopal, Chris; Nelson, Lucas; Zovi, Dina Dai; & Dustin, Elfriede. *The Art of Software Security Testing*: *Identifying Software Security Flaws* [Wysopal 2006].

McGraw, Gary; Chess, Brian; & Migues, Sammy. *Building Security In Maturity Model BSIMM v1.0* [McGraw 2008].

Mead, Nancy R.; Allen, Julia H.; Conklin, W. Arthur; Drommi, Antonio; Harrison, John; Ingalsbe, Jeff; Rainey, James; & Shoemaker, Dan. *Making the Business Case for Software Assurance* [Mead 2009].

Additional Items

Course Delivery Features

In addition to conventional lecture and discussion methods, a number of hands-on individual and team projects are appropriate for course delivery; for example

- individual homework exercises (including straightforward applications of security requirements engineering; architecture and design; coding and testing)
- larger team development projects (including aspects of requirements engineering, architecture and design; coding; and testing)
- research papers and presentations on software security engineering topics such as secure development life cycles, risk analysis, and security of network applications

Course Assessment Features

Assessment should include a combination of homework exercises, class presentations, team projects, and exams. Projects and exams should be weighted heavily. Depending on the degree program, this course could be slanted to emphasize management aspects or technical aspects.

6 Software Quality Assurance

Course Description

This course introduces quality processes and technologies for software development to assure that new software provides sufficient security for the threat environment and functions in the intended manner. Topics include quality and security requirements and specifications; quality in architecture, design, and construction; correctness verification, inspection, and testing techniques; process and product assurance; statistical quality control; and quality management.

Prerequisites

Basic knowledge of software engineering principles, methods, and practices

Syllabus

Topics	Bloom's Levels
Introduction to software quality and security assurance	C
Definition of quality and security requirements and specifications	C/AP
Quality methods in architecture,[4] design, and construction, including secure coding	C/AP
Correctness verification technologies and methods	C/AP
Inspection and review processes	C/AP
Test planning and assessment	C
Testing methods including black box, white box, control flow, and data flow	C
Statistical, usage-based testing for software certification	C
Capabilities and limitations of tools for software assurance	C/AP
Statistical analysis of inspection and test results	C
Software process definition and evaluation for achieving quality objectives	C
Quality assurance tradeoffs and management	C
Assurance standards and regulatory compliance for new software development	K

Source

Partially based on

IEEE-CS & ACM. "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering." *Computing Curriculum Series* [IEEE-CS 2004].

Additional Items

Course Delivery Features

In addition to conventional lecture and discussion methods, a number of hands-on individual and team projects are appropriate for course delivery; for example

- individual analysis projects (including correctness verification of simple programs)

[4] Examples include the Architectural Tradeoff Analysis Method (ATAM) and Quality Attribute Workshop (QAW). Learn more at http://www.sei.cmu.edu/library/abstracts/reports/03tr016.cfm?WT.DCSext.abstractsource=RelatedLinks.

- team analysis projects (including architecture, design, construction, and quality assurance of new programs; inspection and review of existing programs for correct function and security properties; and studies of comparative capabilities of software tools to assist with quality assurance tasks)
- research papers and presentations on quality assurance topics

Course Assessment Features

For topics listed as C/AP (Comprehension/Application), students must be able to use information, methods, and concepts presented in the course to solve representative problems in quality assurance. The individual and team assignments listed above are ideal for evaluating achievement at the AP level. For topics listed as C, in-class case studies and discussions on particular topic areas followed by individual and team evaluation through quizzes can help gauge comprehension.

7 Software Assurance Analytics

Course Description

This course introduces analysis processes and technologies for assuring that legacy or acquired software and services provide sufficient security for the threat environment and function in the intended manner. Topics include software documentation assessment, code analysis and structuring, reverse engineering, supply chain and vendor analysis, requirements and acceptance for acquired software, service assessment, and service level agreements.

Prerequisites

Basic knowledge of software engineering principles, methods, and practices

Syllabus

Topic	Bloom's Level
Evaluating software requirements, specifications, architectures, and designs	C
Evaluating open source and COTS software functionality and security	C
Code analysis and structuring	C/AP
Reverse engineering of code for functionality and security properties	C/AP
Evolution of legacy system functionality and security properties	C
Capabilities/limitations of tools for software functionality and security analysis	C/AP
Analyzing supply chain and vendor processes and capabilities	C
Defining functional and security requirements for acquired software	C
Acceptance testing and evaluation of acquired software	C
Defining software service functional and security requirements	C
Evaluating service provider processes and capabilities	K
Defining supply chain, vendor, and service provider agreements	K
Assurance standards/regulatory compliance for acquired software and services	K

Source

SwA curriculum development team's discussions and professional experience

Additional Items

Course Delivery Features

In addition to conventional lecture and discussion methods, a number of hands-on individual and team projects are appropriate for course delivery; for example

- individual analysis projects (including manual code structuring, reading, and documentation to reverse engineer code)
- team analysis projects (including reverse engineering of code through team review and documentation of function and security properties; analysis of example supply chain threat

environments, vulnerabilities, and improvements; and analysis and improvement of example service provider agreements)

- research papers and presentations on software assurance topics
- course assessment features

For topics listed as C/AP (Comprehension/Application), students must be able to use information, methods, and concepts presented in the course to solve representative problems in software assurance. The individual and team assignments listed above are ideal for evaluating achievement at the AP level. For topics listed as C, in-class case studies and discussions on particular topic areas followed by individual and team evaluation through quizzes can help gauge comprehension.

8 Software Assurance Capstone Project

Course Description

This course focuses on development or modification of a significant software system, employing software assurance knowledge gained from courses throughout the program. For example, an evolutionary or maintenance project would be a good choice for this course. The course includes development or modification of requirements; architecture and design; construction; and testing of the system. In conjunction with this, teams may select COTS software. The project should incorporate use of quality assurance and software assurance analytics.

Students must use a selection of software assurance methods discussed in the earlier courses in this software assurance area of concentration and must manage the project themselves, following all appropriate project management techniques. Success of the project is determined in large part by whether students have adequately applied software assurance methods over the course of the project.

Prerequisites

- Completion of all courses in the software assurance area of concentration and most of the courses in an undergraduate computing program.
- Senior-level status would be an expected requirement.

Syllabus

This course will not necessarily have standard lectures, although regular meetings with the instructor and mini-lectures will likely be needed.

Topic	Bloom's Level
The software development life cycle and the role of software assurance in life-cycle phases	AP
Project team process, organization, communication, and assessment	AP
Project management (planning, risk management, configuration management) and software assurance plans	AP
Quality assurance, software assurance metrics, and software assurance analytics	AP
Requirements; architecture and design; and construction	AP
Testing, inspections, and review, including independent assurance testing	AP
Creation and maintenance of auditable evidence for software assurance	AP
Evolution and operation issues	C/AP
Reuse, COTS selection, and acquisition	C/AP
Project reports and presentations	AP

Sources

IEEE-CS & ACM. "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering." *Computing Curriculum Series*. Capstone course description [IEEE-CS 2004].

Embry Riddle Aeronautical University. *Master of Science in Software Engineering Degree Program*. [ERAU 2010].

Carnegie Mellon University. *95-750 Security Architecture & Analysis* course descriptions [CMU 2010a].

Additional Items

Course Delivery Features

- This is a project-oriented course based on the knowledge and experience gained in other undergraduate courses. No, or very little, new material will be presented in lectures and/or class discussions. Any new knowledge or capability needed to complete the project (for example, domain knowledge or knowledge about the use of a new tool or method) will require research by the students on the project team.
- The instructor typically acts as a team coach or mentor or in some cases as a customer.
- There is much debate about the source of project work. It should be a real-world project with a real customer or a made-up, but realistic, project. There are advantages and disadvantages to both.
- The project should involve significant software security elements, and software assurance methods and activities should be used to address these elements.

Course Assessment Features

All of the course topics are listed at the AP (Application) Bloom's level, which means that students must be able to use information, methods, concepts, and theories to solve problems requiring the skills or knowledge thus far acquired.

- The chief source for assessing how well students achieve the AP level for this course is project artifacts such as process and plan documents; risk and configuration management plans and reports; reports on software assurance audits; requirements and design documents; source code; test plans and reports; inspection and review reports; and team assessments of processes and products.
- In team projects, individual assessment is a special problem. Here are some ideas:
  - Base a student's grade on a combination of a team assessment and an individual assessment (for example, 70% for the team and 30% for the individual).
  - Base the team assessment on the quality of the team artifact (for example, some sort of weighted sum).
  - Base an individual assessment on self-assessment and peer assessment, and, if possible, on the quality of an artifact for which the individual had primary responsibility. An instructor might also use individual observations (for example, how well does an individual participate in a team design review) and/or an interview and discussion with the individual student.

Appendix A: Bloom's Taxonomy and the GSwE2009

Bloom's Taxonomy is a classification system devised in 1956 by a group of educators lead by Benjamin Bloom [Bloom 1956]. The taxonomy can be 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 Taxonomy divides educational objectives into three domains: affective, psychomotor, and cognitive. In this report, the focus is on the cognitive domain, which is concerned with what we know and how we know it [Huitt 2006]. Conventional education systems tend to stress outcomes in the cognitive domain, particularly the lower-level objectives.

Bloom's taxonomy is hierarchical; that is, learning at a higher level is dependent on attaining prerequisite knowledge and skills at the lower levels. Table 1 provides a description of the Bloom's Levels for the Cognitive Domain.

Note: This table was adapted from an appendix in the GSwE2009 [iSSec 2009].

Table 1: Bloom's Taxonomy

Level	Competency	Objective Descriptors
Knowledge (K)	(Lowest level) Remembering previously learned material. Test observation and recall of information, i.e., "bring to mind the appropriate information" (e.g., dates, events, places, knowledge of major ideas, mastery of subject matter).	list, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name (who, when, where, etc.)
Comprehension (C)	Understanding information and ability to grasp meaning of material presented. For example, translate knowledge into new context, interpret facts, compare, contrast, order, group, infer causes, predict consequences, etc.	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
Application (AP)	Ability to use learned material in new and concrete situations. For example, use information, methods, concepts, and theories to solve problems requiring the skills or knowledge presented.	apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover
Analysis (AN)	Ability to decompose learned material into constituent parts in order to understand structure of the whole. This includes seeing patterns, organization of parts, recognition of hidden meanings, and identification of parts.	analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer
Synthesis (S)	Ability to put parts together to form a new whole. This involves using existing ideas to create new ones, generalizing from facts, relating knowledge from several areas, and predicting and drawing conclusions. It may also involve adapting general solution principles to the embodiment of a specific problem.	combine, integrate, modify, rearrange, substitute, plan, create, design, invent, what if?, compose, formulate, prepare, generalize, rewrite
Evaluation (E)	(Highest level) Ability to pass judgment on value of material within a given context or purpose. This involves making comparisons and discriminating between ideas, assessing value of theories, making choices based on reasoned arguments, verifying value of evidence, and recognizing subjectivity.	assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize

Appendix B: Course Syllabus Examples

Content in the following course examples comes from the source listed and is unedited except for adding citations.

Secure Programming (Carnegie Mellon University)

Course Description

This course provides a detailed explanation of common programming errors in C and C++ and describes how these errors can lead to software systems that are vulnerable to exploitation. The course concentrates on security issues intrinsic to the C and C++ programming languages and associated libraries. It does not emphasize security issues involving interactions with external systems such as databases and web servers, as these are rich topics on their own. Topics to be covered include the secure and insecure use of integers, arrays, strings, dynamic memory, formatted input/output functions, and file I/O.

Prerequisites

System Programming

Text

The primary text for the course is *Secure Coding in C and C++* (SSCC) [Seacord 2005].

This secondary text is *The CERT C Secure Coding Standard* [Seacord 2008]. Alternatively, you can refer to the wiki text on the *CERT Secure Coding Standards* website [CERT 2010b].

Syllabus

Secure Programming Concepts
Readings:

- SCCC Chapter 1
- Lions, J. L. ARIANE 5 Flight 501 Failure Report. Paris, France: European Space Agency (ESA) & National Center for Space Study (CNES) Inquiry Board, July 1996.

Strings
Readings:

- SCCC Chapter 2
- Smashing The Stack For Fun And Profit

Integers

- Reading: SCCC Chapter 5

Instrumented Fuzz Testing using AIR Integers

Dangerous Optimizations and a loss of Causality
Readings:

- #5 memset_s() to clear memory, without fear of removal
- Analyzability

Structured code review
Readings:

- Danny Kalev. Static Assertions
- Robert C. Seacord Validating C and C++ For Safety and Security
- Tom Plum, David Keaton. Eliminating Buffer Overflows, Using the Compiler or a Standalone Tool

Pointer Subterfuge

- Reading: SCCC Chapter 3

Dynamic Memory Management

- Reading: SCCC Chapter 4

Formatted output
Readings:

- SCCC Chapter 6
- Hal Burch and Robert C. Seacord. Programming Language Format String Vulnerabilities. Dr. Dobbs Journal. February 02, 2007.
- Robert C. Seacord. Variadic Functions: How they contribute to security vulnerabilities and how to fix them

File IO Basics

Privileges, Permissions, & File I/O
Readings:

- SCCC Chapter 7
- Secure Programming for Linux and Unix HOWTO Chapter 7. Structure Program Internals and Approach

File System Vulnerabilities

Race Conditions
Readings;

- SCCC Chapter 7
- This doesn't have much to do with the lecture but is interesting Fixing Unix/Linux/POSIX Filenames

Rose Overview and C/C++ for Rose

Writing Checkers with Rose

Signals

Environment Variables

Error Handling

Recommended Practices

- Reading: SCCC Chapter 8 “Recommended Practices”

Grading

Examinations

- Midterm examination: 20%
- Final examination: 30%

Assignments

- Assignment #1: Signal Handler Service: 10%
- Assignment #2: Integral Security: 10%
- Assignment #3: Secure Coding Guidelines: 10%
- Assignment #4: Source Code Analysis: 10%
- Assignment #5: Exploit: 10%

Source

Carnegie Mellon *CS 15-392 Secure Programming* undergraduate course [CMU 2010d]

Special Topics: Information Assurance & Security, 6 units (Carnegie Mellon University)

Required Text

Information Security: Contemporary Cases by Marie Wright and John Kakalik. [Wright 2007]

Course Overview

This course is an overview of an increasingly important area of information assurance and security. As more and more functionality and dynamic decision-making are pushed down and out into the organization (power to the edge), assurance and security concerns, with their organizational and human dimensions, impact the fidelity of the data and the very survival of the organization. Class sessions will be centered on case studies and discussion. Topics include overview and definitions, protecting employee data, disaster and contingency planning, compliance with federal information security requirements, tracking a computer intruder, and implementing an information security awareness program. Students will leave the course with an understanding of the various concepts and their impacts on the organization.

Prerequisites

Upper-class standing and at least one programming course (for example 15-110 Introduction to Programming)

This class is a combination of lecture, readings (case studies), & in-class discussion/presentation of case studies.

Course Expectations and Course Grades

To complete this course successfully, students will be expected to complete the following activities. Weights indicate the contribution to the final course grade.

- [Best] 4 of 6 Quiz grades 40%
- [Best] 5 of 6 Case Study Question Narrative grades 25%
- [Best] 5 of 6 Case Study Question Presentation grades 25%
- In-class discussion 10%

See also the "Case Study Requirements" at the end of this section.

From week 2 through week 7 there will be a timed, in-class quiz (open book, open (hardcopy) notebook, open (hardcopy) notes; closed laptops, closed pdas, closed phones, etc. – see also lecture one) based on information/reading from prior week/class **and** reading for the current week/class.

Other factors, such as amount of class participation and punctual, regular attendance may be used, at the instructor's option, to make adjustments to final grades.

Case Study Requirements

The week before a particular case study is scheduled for discussion, case study questions (located at the end of the case study) will be assigned to individual students or small teams of students (depending on the number of students enrolled).

For each case study, student(s) are required to

1. Submit written answers to each question assigned to them. [Students should also bring a hardcopy for themselves to reference during the class discussion.]
    a. The first page must include the following:
        i. Student's full name (first and last name, for all of the students assigned, if more than one)
        ii. Case study title
        iii. The question (in its entirety) and question number
        iv. Page number 1
    b. Each subsequent page must include the following:
        i. student(s) name(s)
        ii. case study title
        iii. question number
        iv. page number of the hardcopy answer (i.e. page 2, etc.)

Grading of the narrative will be based on content, spelling, punctuation, sentence structure, grammar, and adherence to instructions/format. Answers are expected to be in a narrative format,

not PowerPoint. Each student's best 5 of 6 case study narrative answers will be used as part of the computation of the final course grade [worth 25% of final course grade].

Additionally, for each case study question, student(s) are required to

1. Prepare a presentation of their answers, which takes place the following week (when the case will be discussed.)
    a. Presentation should be professional, and the use of visuals, such as PowerPoint, is expected.
    b. The PowerPoint slides should not just be a "cut and pasting" of the narrative answer submitted; it should be able to stand alone – i.e. make sense if one did not have the narrative at hand
    c. Each student must clearly identify themselves at the start of their part of the presentation
    d. If more than one student has been assigned the particular question, each student must participate in the presentation.
    e. One hard copy of the presentation (1-up or 2-up format; NOT 3-up or 6-up format) is due to the instructor at the start of the presentation.
    f. Students are expected to use their laptop computers to do the presentation (i.e. the instructor's laptop is NOT available for student presentations)

Grading of the presentation will be based on content, the level of professionalism displayed in the presentation and adherence to instructions/format. For example, the presentation should not consist of the student reading from the monitor/screen, or from a piece of paper held in front of the student's face. Each student's best 5 of 6 case study presentations will be used as part of the computation of the final course grade [worth 25% of final course grade].

Note: Syllabus may evolve over the course of the term.

Course Outline

Week	Coverage, Readings, Quizzes, etc.
Week 1	**Overview and Definitions** **No quiz; *Readings for this class will be done in class*** Read 1) 'Titles Associated with Executive Compensation' - available at http://www.compensationresources.com/titles-associated-with-executive-compensation.php 2) "The Chief Information Security Officer: An Analysis of the Skills Required for Success" by Dwayne Whitten
Week 2	**CASE: Kraft Foods, Inc Protecting Employee Data** **Required Reading:** Pages 1-24 in the required text Course Syllabus (on 67-309 BlackBoard site) – read in its entirety **Required Viewing:** View 12 minute movie (73MB) "Warriors of the Net": http://www.warriorsofthe.net/ ***Quiz #1***
Week 3	**CASE:** Advo, Inc.: Integrating IT and Physical Security **Required Reading:** Pages 25-60 in the required text ***Quiz #2***

Week	Coverage, Readings, Quizzes, etc.
Week 4	**CASE: Yale New Haven Center for Emergency Preparedness and Disaster Response: Contingency Planning** **Required Reading:** Pages 61-88 in the required text ***Quiz #3***
Week 5	**CASE: SRA International, Inc.: Automating Compliance with Federal Information Security Requirements** **Required Reading:** Pages 115-160 in the required text ***Quiz #4***
Week 6	**CASE: Aetna: Developing and Implementing a Successful Information Security Awareness Program** **Required Reading:** Pages 184-207 in the required text ***Quiz #5***
Week 7	**CASE: FBI New Haven Field Office – Computer Analysis and Response Team: Tracking a Computer Intruder** **Required Reading:** Pages 161-184 in the required text ***Quiz #6***

Source

Carnegie Mellon *67-309 Special Topics: Information Assurance & Security* undergraduate course [CMU 2010c]

Bibliography

URLs are valid as of the publication date of this document.

[ACM 2008]
Association for Computing Machinery (ACM) & IEEE Computer Society (IEEE-CS). "Computer Science Curriculum 2008: An Interim Revision of CS 2001." *Computing Curriculum Series*. http://www.acm.org//education/curricula/ComputerScience2008.pdf (2008).

[Allen 2008]
Allen, Julia H.; Barnum, Sean; Ellison, Robert J.; McGraw, Gary; & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers.* Addison-Wesley Professional, 2008. http://www.sei.cmu.edu/library/abstracts/books/032150917X.cfm

[Bishop 2002]
Bishop, Matt. *Computer Security: Art and Science.* Addison-Wesley Professional, 2002.

[Bloom 1956]
Bloom, B. S., ed. *Taxonomy of educational objectives: The classification of Educational goals: Handbook I, Cognitive Domain*. Longmans, 1956.

[CERT 2010a]
CERT. *SQUARE*. Software Engineering Institute, Carnegie Mellon University. http://www.cert.org/sse/square/ (2010).

[CERT 2010b]
CERT. *Secure Coding Standards*. Software Engineering Institute, Carnegie Mellon University. http://www.cert.org/secure-coding/scstandards.html (2010).

[CMU 2010a]
Carnegie Mellon University (CMU), Master of Information Systems (MISM) degree program. *95-750 Security Architecture & Analysis* course descriptions. http://www.andrew.cmu.edu/course/95-750 (May 2010).

[CMU 2010b]
Carnegie Mellon University (CMU), Electrical & Computer Engineering (ECE) degree program. *Course 18-730: Introduction to Computer Security.* http://www.ece.cmu.edu/courses/18730 (May 2010).

[CMU 2010c]
Carnegie Mellon University (CMU), College of Humanities and Social Sciences Information Systems (IS) degree program. *Course 67-309: Special Topics: Information Assurance and Security* http://www.cmu.edu/information-systems/electives/ (July 2010).

[CMU 2010d]
Carnegie Mellon University (CMU), School of Computer Science. *Course 15-392: Special Topic—Secure Programming*. http://www.cs.cmu.edu/afs/cs/usr/cathyf/www/ugcoursedescriptions.htm (August 2010).

[DHS 2010a]
Department of Homeland Security (DHS) Software Assurance (SwA). *Build Security In*. https://buildsecurityin.us-cert.gov/daisy/adm-bsi/home.html (2010).

[DHS 2010b]
Department of Homeland Security (DHS) Software Assurance (SwA) Workforce Education and Training Working Group. *Software Assurance CBK/Principles Organization.* https://buildsecurityin.us-cert.gov/bsi/dhs/927-BSI.html (2010).

[ERAU 2010]
Embry Riddle Aeronautical University (ERAU). *Master of Science in Software Engineering Degree Program*. http://daytonabeach.erau.edu/coe/degrees/graduate-degrees/software-engineering/index.html (Accessed May 2010).

[Howard 2006]
Howard, Michael & Lipner, Steve. *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software.* Microsoft Press, 2006.

[Huitt 2006]
Huitt, W. "The cognitive system." *Educational Psychology Interactive*. Valdosta State University, http://www.edpsycinteractive.org/topics/cogsys/cogsys.html (2006).

[IEEE-CS 2001]
IEEE Computer Society (IEEE-CS) & Association for Computing Machinery (ACM). *Computing Curricula 2001: Computer Science, Final Report*. http://www.acm.org/education/curric_vols/cc2001.pdf (2001).

[IEEE-CS 2004]
IEEE Computer Society (IEEE-CS) & Association for Computing Machinery (ACM). "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering." *Computing Curriculum Series*. http://sites.computer.org/ccse/SE2004Volume.pdf (2004).

[iSSEc 2009]
Integrated Software & Systems Engineering Curriculum (iSSEc) Project. *Graduate Software Engineering 2009 (GSwE2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering, Version 1.0*. Stevens Institute of Technology, 2009.

[McGraw 2006]
McGraw, Gary. *Software Security: Building Security In.* Addison-Wesley Professional, 2006.

[McGraw 2008]
McGraw, Gary; Chess, Brian; & Migues, Sammy. *Building Security In Maturity Model BSIMM v1.0*. http://www.bsi-mm.com/ (March 2008).

[Mead 2005]
Mead, N. R.; Hough, E.; & Stehney, T. *Security Quality Requirements Engineering (SQUARE) Methodology,* (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/library/abstracts/reports/05tr009.cfm

[Mead 2009]
Mead, Nancy R.; Allen, Julia H.; Conklin, W. Arthur; Drommi, Antonio; Harrison, John; Ingalsbe, Jeff; Rainey, James; & Shoemaker, Dan. *Making the Business Case for Software Assurance* (CMU/SEI-2009-SR-001). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09sr001.cfm

[Mead 2010]
Mead, Nancy R.; Allen, Julia H.; Ardis, Mark; Hilburn, Thomas B.; Kornecki, Andrew J.; Linger, Rick; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tr005.cfm

[Purdue 2010]
Purdue University. *CS42600 Computer Security*, upper division undergraduate course. https://esa-oas-prod-wl.itap.purdue.edu/prod/bwckctlg.p_disp_course_detail?cat_term_in=201110&subj_code_in=CS&crse_numb_in=42600 (May 2010).

[Redwine 2004]
Redwine, Samuel T., Jr. *Secure Software Engineering Education*. https://buildsecurityin.us-cert.gov/swa/downloads/JMU_SSE.pdf (May 2010).

[Seacord 2005]
Seacord, Robert C. *Secure Coding in C and C++*. Addison-Wesley, 2005. http://www.sei.cmu.edu/library/abstracts/books/0321335724.cfm

[Seacord 2008]
Seacord, Robert C. *The CERT C Secure Coding Standard*. Addison-Wesley, 2008. http://www.sei.cmu.edu/library/abstracts/books/0321563212.cfm

[Stallings 2007]
Stallings, W. *Network Security Essentials*, 3rd Ed. Prentice Hall, 2007.

[UC Davis 2010]
University of California Davis. *ECS 153 Computer Security*, upper division undergraduate course. http://www.cs.ucdavis.edu/courses/exp_course_desc/153.html (May 2010).

[Wright 2007]
Wright, Marie A. & Kakalik, John S. *Information Security: Contemporary Cases*. Jones and Bartlett Publishers, Inc., 2007.

[Wysopal 2006]
Wysopal, Chris; Nelson, Lucas; Zovi, Dina Dai; & Dustin, Elfriede. *The Art of Software Security Testing*: *Identifying Software Security Flaws.* Addison-Wesley Professional, 2006.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	August 2010	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines	FA8721-05-C-0003

6. AUTHOR(S)

Nancy R. Mead, Thomas J. Hilburn, Richard C. Linger

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2010-TR-019

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2010-019

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Modern society depends on software systems of ever-increasing scope and complexity. Virtually every sphere of human activity is impacted by these systems, from social interaction in our personal lives to business, energy, transportation, education, communication, government, and defense. Because the consequences of failure can be severe, dependable functionality and security are essential. As a result, software assurance is emerging as an important discipline for the development, acquisition, and operation of software systems and services that provide requisite levels of dependability and security. This report is the second volume in the Software Assurance Curriculum Project sponsored by the Department of Homeland Security. The first volume, the *Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005), presented a body of knowledge from which to create a Master of Software Assurance degree program, as both a standalone offering and as a track within existing software engineering and computer science master's degree programs. This report focuses on an undergraduate curriculum specialization for software assurance. The seven courses in this specialization are intended to provide students with fundamental skills for either entering the field directly or continuing with graduate-level education.

14. SUBJECT TERMS	15. NUMBER OF PAGES
software assurance, software assurance education, software engineering education, computer science education	41

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL