Embrace Your Inner Twinkie

us. We need to keep up with, preferably in front of, technological and social change.

What should we do when faced with the latest challenge (starting with mobile devices) to our professional world? We should embrace change. Each time we are faced with the latest "interference" with our teaching mission, we have an opportunity to think outside the box. Look for a way to use the technology to your advantage. Each time someone presents us with a truly off the wall innovation we have an opportunity to show mental flexibility and to experiment with something new and interesting. Who knows what we might learn? Thus it is with smart phones in the classroom. Annoying and disruptive as they are, they provide us an opportunity to stay on the leading edge of our field. We must poke, prod, dissect and peer closely at them. We must give a fair shake to all innovative ideas about addressing the challenge of mobile devices in the classroom. We have to be willing to try the weird. Often it takes guts.

Each time someone presents us with a truly off the wall innovation we have an opportunity to show mental flexibility and to experiment with something new and interesting.

About that deep fried Twinkie? It was delicious. **lr**

Lisa C. Kaczmarczyk
Consultant
San Diego, California 92130 USA
lisak@acm.org

DOI: 10.1145/2189835.2189843

COMMUNITY COLLEGE CORNER

Elizabeth K. Hawthorne

Infusing Software Assurance in Computing Curricula

ACCORDING TO THE COMPUTER EMERGENCY RESPONSE TEAM (CERT) at the Software Engineering Institute (SEI) at Carnegie Mellon University, "nearly every facet of modern society depends heavily on highly complex software systems. The business, energy, transportation, education, communication, government, and defense communities rely on software to function, and software is an intrinsic part of our personal lives. Software assurance is an important discipline to ensure that software systems and services function dependably and are secure". Toward assured and dependably software in modern society, the ACM Committee for Computing Education in Community Colleges (CCECC) partnered with the SEI to produce *Software Assurance Curriculum Project Volume IV: Community College Education* [1]. This 2011 Technical Report (CMU/SEI-2011-TR-017), sponsored by the U.S. Department of Homeland Security (DHS) National Security Division (NCSD), includes a review of related curricula suitable to community colleges, outcomes and a body of knowledge, expected academic backgrounds of target audiences, and outlines of six undergraduate courses.

According to the American Association for Community Colleges [2], the mission of the community college sector is diverse: preparing students for transfer into four-year institutions, helping working adults prepare for new careers, as well as offering noncredit programs that offer a range of knowledge and skills. The target audience for the six courses outlined in this curriculum is two-fold: 1) students planning to transfer into some type of baccalaureate degree program in computing, and 2) students with prior undergraduate technical degrees who wish to become more specialized in software assurance.

The training of professionals requires a detailed knowledge of how to get things to work right AND how to prevent (reduce the likelihood) of 'never events'

The CCECC provided the content of the introductory Computer Science I-II-III course sequence [3] derived from the ACM/IEEE-CS CS2008 Interim Report [4] and augmented with software assurance topics. Assessment rubrics for these three

introductory computer science courses were developed by the CCECC [3] and are also included in the report. Volume IV was also informed by three years (2009, 2010 and 2011) of working group reports from the annual ACM conference on Innovation and Technology in Computer Science Education (ITiCSE) [5, 6, and 7].

In developing the first volume [8] in this four-part curricular series, the term software assurance (SwA) was 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". The emphasis here is on the concomitant importance of both technologies and process in software assurance. This emphasis was carried through all the way to Volume IV, Community College Education [1].

An appropriate selection of courses for a specialty or concentration in Software Assurance (SwA) is recommended in this report as Computer Science I, II, and III and more specialized courses such as Introduction to Computer Security, Secure Coding, and Introduction to Assured Software Engineering. These more specialized courses are not intended to be an exhaustive list of possibilities, but rather a set of courses that could reasonably be taken by students wishing to pursue further education in software assurance. Community colleges could easily offer a certificate program in SwA complimentary to typical associate degrees in computer science, information systems and information technology. This certificate option is part of the CCECC's curriculum, assessment and pedagogy online repository – www.capspace.org. Brief descriptions of the six community college courses outlined in Volume IV are as follows.

Computer Science I: This course is the first in a three-course sequence that provides students with a foundation in computer science. Using a high level programming language, students develop fundamental programming skills, including secure coding awareness, human-computer interactions, and social responsibility.

Computer Science II: This course is the second in a three-course sequence that provides students with a foundation in computer science. Using a high level programming language, students develop intermediate programming skills with an emphasis on algorithms, software development, and secure coding techniques, as well as gain an appreciation for ethical conduct.

Computer Science III: This course is the third in a three-course sequence that provides students with a foundation in computer science. Using a high level programming language, students continue to develop programming skills focusing on data structures, algorithmic analysis, software engineering and software assurance principles, as well as giving emphasis to professionalism.

Introduction to Computer Security: 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 secu-

rity 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.

Secure Coding: This course covers security vulnerabilities of programming in weakly typed languages like C and in more modern languages like Java. Common weaknesses exploited by attackers are discussed, as well as mitigation strategies to prevent those weaknesses. Students practice programming and analysis of software systems through testing and static analysis. Topics covered include methods for preventing unauthorized access or manipulation of data, input validation and user authentication, memory management issues related to overflow and corruption, misuse of strings and pointers, and inter-process communication vulnerabilities.

Introduction to Assured Software Engineering: This course covers the basic principles and concepts of assured software engineering; system requirements; secure programming in the large; modeling and testing; object-oriented analysis and design using the unified modeling language (UML); design patterns; frameworks and application programming interfaces (APIs); client-server architecture; user interface technology; and the analysis, design and programming of extensible software systems.

Below are two possible sequencing options for the six courses given the suggested pre-requisite structure. Other options are also possible to meet local program needs.

It is important to note that the Computer Science I–II–III course sequence, typical at community colleges as well as smaller four-year colleges, is the equivalent of the Computer Science I–II course sequence at other four-year colleges and universities. The depth of coverage of the topics in each course varies, as do the associated level of Bloom's Taxonomy. In many areas, students need to be able to discuss and describe the topics, but in other areas they must be able to apply the techniques learned in the course to actual software projects.

As part of the outreach efforts, a well-attended Birds-of-a-Feather roundtable discussion was held at the 2012 SIGCSE conference. Volume IV along with previous volumes of this curricular project - *Volume I: Master of Software Assurance Reference Curriculum* [8], *Volume II: Undergraduate Course Outlines* [9], and *Volume III: Software Assurance Course Syllabi* [10] - were disseminated at SIGCSE 2012. **Ir**

Option 1 for typical course sequencing

Term 1	Term 2	Term 3	Term 4
CS I	CS II	CS III	Secure Coding
Discrete Structures	Calculus I	Assured Software Engineering	
	Introduction to Computer Security		

Option 2 for typical course sequencing

Term 1	Term 2	Term 3	Term 4
CS I	CS II	CS III	Secure Coding
Discrete Structures	Calculus I	Introduction to Computer Security	Assured Software Engineering

References

[1] Mead, Nancy R., Hawthorne, Elizabeth K. and Ardis, Mark. (2011) *Software Assurance Curriculum Project Volume IV: Community College Education* (CMU/SEI-2011-TR-017). Software Engineering Institute, Carnegie Mellon University. available from www.cert.org/mswa/.

[2] American Association of Community Colleges (AACC). (2011) *Students at Community Colleges.* available from www.aacc.nche.edu/AboutCC/Trends/Pages/studentsatcommunitycolleges.aspx

[3] Association for Computing Machinery Committee for *Computing Education in Community Colleges (ACM CCECC). (2009) Computing Curricula 2009: Guidelines for Associate-Degree Transfer Curriculum in Computer Science.* ACM and IEEE Computer Society. available from www.acmccecc.org/2009ComputerScienceTransferGuidelines.pdf.

[4] Association for Computing Machinery & IEEE Computer Society. (2008) *Computer Science Curriculum 2008: An Interim Revision of CS2001.* available from www.acm.org/education/curricula/ComputerScience2008.pdf.

[5] Cooper, Stephen, Nickell, Christine, Piotrowski, Victor, Oldfield, Brenda, Abdallah, Ali, Bishop, Matt, Caelli, Bill, Dark, Melissa, Hawthorne, Elizabeth K., Hoffman, Lance, Pérez, Lance C., Pfleeger, Charles, Raines, Richard, Schou, Corey and Brynielsson, Joel. (2009) *An exploration of the current state of information assurance education.* ACM SIGCSE Bulletin, Volume 41 Issue 4, pp. 109 -125. doi: 10.1145/1709424.1709457.

[6] Cooper, Stephen, Nickell, Christine, Pérez, Lance C., Oldfield, Brenda, Brynielsson, Joel, Gencer Gökce, Asım, Hawthorne, Elizabeth K., Karl J. Klee, Lawrence, Andrea and Wetzel, Susanne. (2010) *Towards information assurance (IA) curricular guidelines.* ACM ITiCSE-WGR '10: Proceedings of the 15th annual conference reports on Innovation and technology in computer science education. doi: 10.1145/1971681.1971686.

[7] Pérez, Lance C., Cooper, Stephen, Hawthorne, Elizabeth K., Wetzel, Susanne, Brynielsson, Joel, Gencer Gökce, Asim, Impagliazzo, John, Khmelevsky, Youry, Klee, Karl J., Leary, Margaret, Philips, Amelia, Pohlmann, Norbert, Taylor, Blair and Upadhyaya, Shambhu. (2011) *Information assurance education in two- and four-year institutions.* ACM ITiCSE-WGR '11: Proceedings of the 16th annual conference reports on Innovation and technology in computer science education. doi: 10.1145/2078856.2078860.

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

[9] Mead, Nancy R.; Hilburn, Thomas B.; & Linger, Rick. (2010) *Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines* (CMU/SEI-2010-TR-019). Software Engineering Institute, Carnegie Mellon University. available from www.sei.cmu.edu/library/abstracts/reports/10tr019.cfm.

[10] Mead, Nancy R.; Allen, Julia H.; Ardis, Mark; Hilburn, Thomas B.; Kornecki, Andrew J.; & Linger, Rick. (2011) *Software Assurance Curriculum Project Volume III: Master of Software Assurance Course Syllabi* (CMU/SEI-2011-TR-013). Software Engineering Institute, Carnegie Mellon University. available from www.sei.cmu.edu/library/abstracts/reports/11tr013.cfm

Elizabeth K. Hawthorne
Computer Science Department
Union County College
Cranford, New Jersey 07016 USA
Hawthorne@ucc.edu

DOI: 10.1145/2189835.2189844