Proceedings of the 9th Colloquium for Information Systems Security Education
Georgia Institute of Technology
Atlanta, Georgia, 6-9, June 2005

Secure Code: The Capstone Class in an Information Assurance Track

B. Endicott-Popovsky, V. Popovsky and D. Frincke, *IEEE Member*

At the 7th and 8th Annual CISSE conferences, case studies were presented describing a process for adding a three-course track in information assurance to the curriculum of a small, private university in the Pacific Northwest, with only a moderate budget and without hiring additional permanent faculty. [1, 2] In this paper, we finish describing the evolution of that curriculum, by discussing the third, and final, course in the series—Secure Code. [3]

This course was designed to lead a primarily professional, mature student audience to a learning epiphany that would change their behaviors as developers. To achieve this end, the authors developed a pedagogical model for designing IA curriculum that draws on sources from both East and West. Indeed, several months later, after having completed the course, students indicated they still were using the secure coding techniques they were taught.

The original stimulus to create this particular course came from an NSA Center of Excellence (University of Idaho) whose director encouraged Seattle University to pursue work being done to identify and build instruction in best-coding practices. The lessons learned from this effort could prove useful to other universities contemplating similar attempts to develop an IA track or implement a course in Secure Coding Practices.

I. Introduction

In January 15, 2002, just months before the development of the Seattle University (SU) curriculum described in this paper, the Chairman of Microsoft, Bill Gates, wrote his famous 'Trustworthy Computing' memo. [4] In it he outlined an initiative to focus resources on developing a 'new breed' of computing system that would make systems more 'reliable and available.' [4]

The memo sent ripples through the Pacific Northwest development community by focusing a spotlight on software security. This regional emphasis on software security requirements served as a driver for expansion of the Master's of Science in Software Engineering (MSSE) program at Seattle University (SU)[1].

Feedback from MSSE students seeking employment locally indicated interviewers were concerned about their knowledge of IA and good coding practices. Applicants were challenged to think about doing their jobs differently, with security in mind, and needed a stronger academic background to draw upon.

Meeting the demands of local industry and satisfying student needs for new, marketable skills became the major motivation behind SU's commitment to develop an IA track within the MSSE program. [3] This new track was launched with three courses. The first introduced basic IA concepts, mapping to CNSS standard 4011. The second covered computer and network forensics, exploring the consequences of poorly written software. The third explored methods and approaches for avoiding bad code. Each course builds on the next and must be taken in order.

Course developers placed this course at the end of the series by design. Since many students in the program are seasoned adult professionals who have firmly formed coding habits developed during the Northwest dot.com boom, it was believed they needed to be convinced of security's importance first, before they would be open to learning new ways to code.

The authors expect that at some time in the future a separate course in secure coding may not be necessary. SU's software engineering curriculum is undergoing redesign currently with one of the objectives to integrate secure coding methods and practices throughout the program. In the meantime, the department has made a pragmatic decision to teach secure coding methods as a separate class.

Taken as a series, the curriculum has led to 'aha' experiences for students that have convinced the majority of the need to 'bake security in' to their development

B. Endicott-Popovsky, Lecturer, Seattle University;
V..Popovsky, Affiliate Professor, School of Education, Department of HPERD, University of Idaho;
D. Frincke, Chief Scientist Cybersecurity, Pacific Northwest National Laboratory and Associate Professor (on leave), Computer Science Department, University of Idaho

[1] Local Fortune 500 companies are primary beneficiaries of SU's software engineering program graduates, all together more than 350 in number since the beginning of the program. Microsoft, alone, employs over 60.

ISBN 1-933510-99-4/05/$15.00 © 2005 IRI/CISSE

activities. Students report a permanent change in their approach to software development. This was the desired end-state for this program in the minds of the developers when they started this initiative.

This was a particularly satisfying result considering that, as mentioned earlier, the typical MSSE student is a mature professional with years of experience and set coding habits that are hard to change. The authors attribute the result to the application of a pedagogical model that integrates sources from both East and West.

The balance of this paper deals with how the course in question, Secure Code, was conceived and developed using this pedagogical model and resources from a variety of available sources to deliver a realistic and meaningful learning experience. Curriculum artifacts are available in the Microsoft Corporation's curriculum database. [5]

II. Pedagogical Model for IA Curriculum Development

Because of the nature of the students and goals set for the curriculum–changing the work habits and mindset of professionals, rather than shaping the incremental progress of relatively receptive new students–it was determined that a traditional pedagogical model would be ineffective. Thus, a hybrid pedagogical model for designing IA curriculum was designed, drawing on sources from both East and West. This section discusses that model.

A. Theoretical Basis

We begin by drawing on Kuzmina [6] and Bespalko [7]'s findings, paraphrasing them here. Pedagogy can be delivered effectively as an art or as a science. As an art, pedagogy is an intuitive process. As a science, it is practiced through pedagogical systems, that are based on sound pedagogical theory, and that allow production of repeatable results. An effective approach, involves treating learning as a whole system where content, student, teacher and teaching methods interact in relation to one another as seen through the lens of the identified goals or learning objectives desired.

Employing a theoretical analysis of East and West pedagogical literature [6, 7, 8, 9, 10, 11], the authors developed a pedagogical model for IA curriculum development (Figure 1) that was applied and refined throughout the development of the Seattle University IA track.

The theoretical base for the model derives from three primary sources:

1) Theory of Pedagogical Systems—a systems approach [6, 7]
2) Psycho-Pedagogical Theory of Activity—combining views from East and West [8, 9, 10, 11]
3) Theory and Methodologies of Information Assurance, Computer Science and Trustworthy Computing Development [12, 13, 14]

B. The Pedagogical Model

The pedagogical system (Figure 1) resides in a dynamic, social/professional context impacted by **trends** (technological, economic, etc.) and the **job market**. These two elements interact. For example, advances in software and hardware are a trend that has influenced the skills demanded in the job market, which, in turn, influences curriculum taught to some degree. (For example, C++ and Java are now the basic languages taught in computer science programs, not C.)

The hiring climate induced by the Gate's Trustworthy Computing memo is a trend affecting the job market in Seattle, leading to the decision to introduce IA into SU's MSSE curriculum.

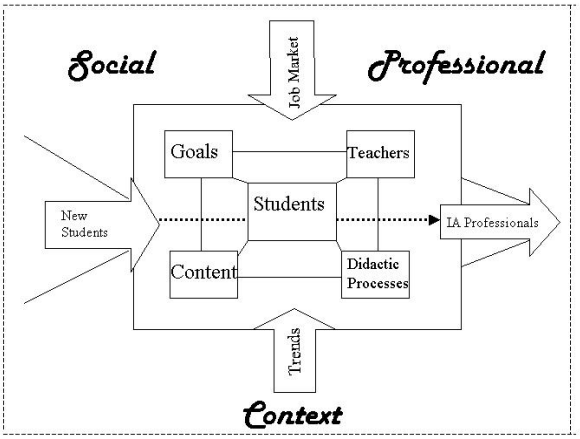

Figure 1 Pedagogical Model for IA Curriculum Development

Within the pedagogical model, five elements exist—students, goals, content, the teacher and didactic processes. Acting as a system, each component is influenced by the other, and then implemented through teaching technologies that provide students' learning experiences. By describing components in relation to the learning objectives and each other, an educational plan is developed.

In this model, the **student** is central. They enter the pedagogical system as new students and exit as IA professionals. In the case at hand, students were already professionals, but lacked background in IA.

Continuing to read the model from left to right, **goals**, in the form of learning objectives, are established for each

instance of curriculum development, based on the characteristics of **students** entering the system and the social/professional context in which the system exists. In this case, developing IA courses, the **goal** of the learning experience was to induce in students a *change* in their approach to software development so that they permanently adopt secure coding techniques, rather than giving them their *first* introduction to software development.

The **teacher** influences the pedagogical system interacting with all the other elements through their educational background, style, strengths and weaknesses, etc. And finally, a mix of **didactic processes** are identified—course organization, teaching methods, techniques, etc.

The characteristics of the MSSE **student** body—experienced professionals, set in their ways, in whom change of behavior is to be induced, drove this model. In this case, **didactic processes** become a prescription, or set of educational "remedies," that deliver the content, given the natures of both **student** and **teacher**. Thus the relative preparedness of the **student** is taken into account (the level, background knowledge, age, experience, etc.) along with the approach of the **teacher** (influenced by their classroom experience).

Applying the model, the **content** to be delivered was passed through the filter of **goals** and adjusted to accommodate the strengths and weaknesses of the **teacher**'s approaches, as well as the **students**' background and nature. [6] With students set in their ways, these remedies emphasized active forms of education (business games, projects, cooperative education) where students learn experientially.

The more precisely the five components in the pedagogical model are defined, along with the connections between them, the more repeatable and predictable the learning results. [6, 7] In this view, the curriculum designer spends much time understanding the characteristics and nature of each element and how each affects the characteristics and nature of the others in an iterative process. A precise description of each element in the system follows, starting with **students**.

C. Students

Students in the Masters of Software Engineering program at Seattle University on average have 10 years of development experience with well-respected Fortune 500 companies. These students are somewhat entrenched in their current method of coding, and may or may not be in class to learn. Some need the degree for promotion; others are satisfying a personal goal to receive an advanced degree; still others enjoy the networking. For an instructor in the MSSE program, the challenge of the tie to relevancy is heightened above and beyond a traditional resident campus.

In addition, teaching these students is a bit like coaching star athletes. Some have earned recognition at the national and international level as experts; some have written books; some have accumulated wealth during the high tech and dot.com booms. They can be resistant to criticism or correction.

To reach this group and induce a major epiphany requires that they be led, experientially, through the 'aha' process. They don't appreciate being lectured to, a conclusion based on years of experience one of the authors has reaching and teaching this group of individuals.

D. Goals

Setting goals and learning objectives were determined according to the Psycho-Pedagogical Theory of Activity. [8, 9, 10]

The primary purpose of the Secure Code curriculum was to deliver an authentic learning experience resulting in changed behavior.

1. Learning Principles

The following principles infused the process of setting learning objectives:

a) The learning plan must emphasize students' professional development, rather than the transmission of information.
b) The ending result must be the personal and professional growth of the student, manifesting in their ability to work creatively and to continue self-education beyond the classroom.
c) Educational efficiency must be based on specific desired outcomes (learning objectives) and must be measurable.

2. Learning Objectives

The course was designed to inspire students to adopt trustworthy computing best practices as reflected in the six learning objectives established for the course:

1) Describe and demonstrate threat modeling and threat modeling techniques
2) Demonstrate secure code techniques
3) Describe, design and write code that mitigates against cyber attack
4) Develop systems that protect data and information
5) Apply security concepts during their software development activities

ISBN 1-933510-99-4/05/$15.00 © 2005 IRI/CISSE

6) Recount IA principles and practices

These learning objectives are traceable to two to three learning objectives associated with each of 10 weekly lessons.

E. Content of Secure Code Course

The **content** consists of a body of knowledge defined either by standards or at administrative discretion. In this instance, **conten**t is defined by 1) the CNSS standards and 2) the IEEE/ACM curriculum model for Computer Science education. [12, 13]

In selecting the precise topics out of this large body of knowledge, the authors applied a graph method, using a series of matrices to establish the necessary computer science and information assurance topics that matched learning objectives for each course. Additionally, the volume of topics covered was matched to the time available for learning--in this case, ten weeks.

Course content was structured for the best possible learning (distribution across the term, schedules, educational elements). Methods of control and knowledge evaluation were developed according to learning goals set for each course and each lesson within each course. Rubrics for each assignment were created that traced back from the assignment to the learning objectives.

1. Learning Levels

Students were evaluated across a continuum of assignments that ranged from testing their ability to reproduce what they learn, to assessing their ability to produce, creatively, on their own, by applying new knowledge. Assignment difficulty progressed from reproductive to independently productive, culminating in an end-of-course project that required students integrate everything they learned to solve a complex, real-world problem.

A learning plan for each course included the following artifact mapping expected levels of learning to assignments:

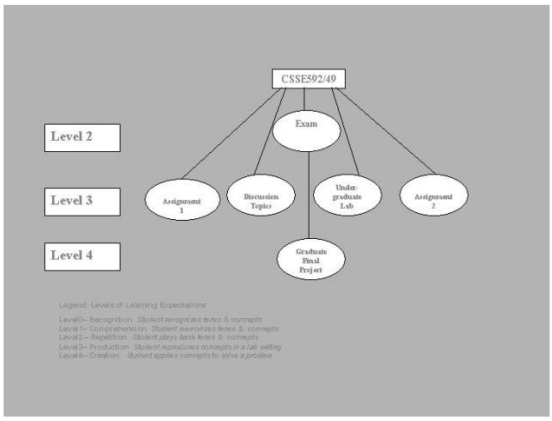

Figure 2 Levels of Learning by Assignment

In Figure 2, the level of reproductive/productive skill expected from students is ascribed to each assignment. In this case, an exam tests a student's ability to reproduce knowledge (Level 2), while a final project allowed students to work independently, testing their ability to produce creatively and independently (Level 4). The balance of the work was performed at a Level 3, which is defined as structured work they could do on their own, but with the dictates of well-defined protocols.

By progressing students through these levels of learning—from lowest to highest, students achieved success in the simpler, reproductive tasks, building confidence to tackle an independent project on their own. This progressive process also engages students in their own learning and development, teaching them an object lesson in how to learn on their own, the goal being to make them adopt new practices and become independent.

2. Cognitive Lessons

The specific topics, addressed in the course, introduced students to coding approaches that mitigate against the threats of cyber attack and raise awareness of information assurance issues. Material covered provided students with an understanding of secure coding techniques and a range of security subjects related to database and web development with the goal of making the software engineering student a better developer.

The major focus was on learning threat modeling techniques as a part of the software engineering process. Successful threat modeling requires a fairly sophisticated background in information assurance and software development so that students are able to identify a wide array of possible threats and code vulnerabilities, improving the end product in any development activity.

ISBN 1-933510-99-4/05/$15.00 © 2005 IRI/CISSE

F. Didactic Processes

Conventional pedagogy, which involves delivering information in the stand-up, lecture mode in a one-way communication, accompanied by forced learning techniques such as "cramming" for exams, would probably fail to induce true behavior change in this particular student body. For this reason, the authors decided to use active forms of education.

An example of such an active form is the business game designed for the Computer Forensics course in the IA track. [2] This game was built around an actual computer intrusion investigation and culminates in a dramatic end-of-course, role play in a mock courtroom in which students “testify” as computer forensics experts.

1. Weekly Outline of Topics

The Secure Code course began with a brief review of information assurance (covered extensively in the two previous classes) and provided students an opportunity to develop a full threat model of a system they were developing using the techniques taught in the class.

Weekly course topics stepped through the following outline. Each week had a lesson plan with learning objectives that mapped back to the learning objectives for the course. Reading materials, homework assignments and discussion topics aligned with each week's learning objectives.

WEEK 1	Review of IA Issues and Concepts
WEEK 2	Security Principles
WEEK 3	Threat Modeling
WEEK 4	Threat Modeling (cont’d.)
WEEK 5	Buffer Overflow and Input
WEEK 6	Database / Web Specific Input
WEEK 7	Cryptographic Issues / Protecting Data
WEEK 8	Access Control / Least Privilege
WEEK 9	Advanced Topics: Testing
WEEK 10	THREAT MODELING PROJECTS
WEEK 11	FINAL EXAM

2. Textbooks

The following book was adopted as a textbook for the class. This was supplemented by online readings.

Howard, Michael and Brian LeBlanc. (2003). *Writing Secure Code*. Redmond, WA: Microsoft Press. [15]

Lists of potential educational elements (readings, assignments, projects, lecture) were developed with learning outcomes associated with each. Then elements for the curriculum were selected from this list based on the specific levels of learning that the developers wished students to achieve.

3. Community Resources

The successful outcome for the Secure Code course relied on resources provided by a variety of sources both inside and outside the university. Table 1 summarizes these resources, their source and the contributions to the course.

Source	**Resources Provided**	**Contribution**
Community experts	Guest lecturers –CTO local network engineering firm –Textbook authors –Software developers– –Penetration tester	Provided guest lectures & Demos (i.e. Cross- site scripting)
Conference	Microsoft Dev Days MS Writing Secure Code track. Seattle: Mar 18	Demos of various attacks
Publication	• Textbooks • Journal articles • Magazines • Online sources • Newspapers	Provided updated info. on trustworthy computing

Table 1: Sources/Resources Contributing to Course Success

Professionals used as guest speakers throughout the course included the authors of the text, themselves, as well as various experts in different aspects of the course content.

One of the most successful lecturers was an ex-hacker, now employed as a penetration tester for a well known local firm, who demonstrated his experience breaking-in to a variety of systems in a closed lab session where machines were disconnected from both the Internet and intranet. Students overwhelmingly recommended he be a permanent fixture in the course. He reminded them of what they are up against!

4. Assignments

Educational activity emphasized the application of theoretical knowledge to solving practical problems. By bringing practical, professional experiences into the classroom, new knowledge is quickly gained.

ISBN 1-933510-99-4/05/$15.00 © 2005 IRI/CISSE

Students were given the following assignments in Table 2 across the 10 weeks of the class. Note the progression in Levels, from lowest to highest.

Assignment	Description
1) My Doom / http://www.securityfocus.com Level 2	Choose between: 1) analyzing why MyDoom spread so fast, and recommended mitigations, 2) going to http://www.securityfocus.com, to determine the patterns and trends in the content of the information.
2) Operating System Vulnerabilities Level 2-3	Examine the OS at the 2 web sites below. Discuss a major vulnerability in each; how to mitigate against each vulnerability; how to defend this vulnerability against attack; how to compare both OS from a vulnerability view. http://rhn.redhat.com/errata/rh9-errata-security.html http://www.microsoft.com/technet/treeview/?url=/technet/security/current.asp?frame=true.
3) Analyzing Code Level 3	Given several examples of vulnerable or malicious code, analyze each, describing line by line the vulnerability or exploit (how each works) and then identify/demonstrate a means for mitigating against that vulnerability or exploit. .
4) Discussion Topic--Leadership Assignment Level 4	Select a topic that interests you for class discussion. (list provided) Review appropriate literature; prepare a class presentation; provide the class a list of readings so they can come to class prepared to participate.
5) Threat Modeling Level 4	Apply threat modeling to an e-commerce example found available (free) at: PetShop URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/petshop3x.asp
6) Final Exam: Reflection essays Level 4	Answer questions that require identification of any observed changes in approaches to software development and coding.

Table 2. Assignments

G. Teaching Responsibilities

A full time faculty member from Seattle University's Computer Science Department, with specialization in IA, assumed primary responsibility for teaching the course. However, with the breadth of information covered and the caliber of the students in the class, guest lecturers with significant backgrounds in certain topics were invited to half of the classes. The faculty member drew extensively on connections within the security community to attract top experts.

III. Course Outcomes

Fourteen graduate students completed the course. Student outcomes were evaluated in several ways.

1) Rubrics for assignments were created that traced to the learning objectives. Thus how students did on individual assignments is a direct measurement of how students are faring in meeting the learning objectives of the course.

Table 3 shows how assignments mapped to learning objectives and what curriculum developers hoped to achieve as desired outcomes for each objective.

Learning Objective	Assignment	Desired Outcome
1) Describe, demonstrate threat modeling	Threat Mod'ling	Students learn techniques for eliminating stwe. vulnerabilities
2) Demonstrate secure code techniques	Analyze Code Discuss. Topics	Students apply what they learn successfully
3) Describe, design, write secure code	My Doom OS Analysis Analyze Code Threat Mod'ling Final Exam	Students apply what they learn in and outside of class
4) Develop systems that protect data	Analyze Code Threat Mod'ling Final Exam	Students apply what they learn in and outside of class
5) Apply security concepts to software development	Analyze Code Threat Mod'ling Final Exam	Students apply what they learn in and outside of class
6) Recount IA principles/ practices	My Doom OS Analysis	Students refresh their IA knowledge

Table 3. Learning Objectives Mapped to Assignments

2) The "Analyze Code" assignment, asking students to analyze code and develop mitigation, was a direct test of whether students had become sensitized to code vulnerabilities. Toward the end of the course, a dozen code fragments were distributed with over 30 known secure code violations. 11 of the 14 students identified all code violations. The remaining 3 identified 95%.

ISBN 1-933510-99-4/05/$15.00 © 2005 IRI/CISSE

The next time this course is taught, a useful experiment would be to more precisely gauge coding skill change. This might be done by administering a similar hands-on test at the beginning of the course and at the end to measure learning. In addition, if this were followed up a few years later, it would help identify how much of the material was retained, though the sample size would likely make it difficult to determine specific conclusions without a broader application of such testing.

Comparisons with students who had not taken such a course would also be of value, though in a way one could, as an initial gauge, consider the entry-level skills of the mature professionals in this classroom to be indicative of the general population.

3) As one element of the final exam, students could choose to write reflection essays answering the questions:

- What were 3 new ideas you learned in this course?

- Describe how you will/have changed your code writing as a result of what you have learned

Six of the 14 students chose to write reflection essays on their final exams. Of the six, five report significant change and one very experienced and difficult student acknowledged learning new ideas.

The responses grouped into three categories: personal discoveries, how they experienced permanent change, how they recognize the need for continuous learning beyond the classroom. A set of responses are quoted below to provide an indication of the impact this learning experience had on students.

A. Personal Discoveries

The following statements indicate students made significant discoveries:

1) "I was always under the impression that I wouldn't have to worry about security because high level languages will already protect me. The lectures on SQL injection and Integer overflows certainly proved me wrong."
2) ..."I have a greater understanding for ... how easily a malicious hacker can compromise an unsuspecting application or computer."
3) " 'All input is evil.' This is probably the most important concept for me. This is a mindset more than a particular technique."
4) "This class raised my awareness of how vulnerable the Internet is."
5) "As a software developer, I would not take security that much seriously and assumed that is the Network and System Administrators' job to handle all aspects of software system security. Now I know that many serious security holes can fall into the programming category and insecure code."
6) "The concept of threat modeling was a new methodology to add to my security arsenal."

B. Permanent Change

The following statements indicate students experienced permanent change in their coding habits:

1) "My perspective on coding and security certainly changed with this course. As a direct result, I will adjust how I create SQL code to prevent SQL injection."
2) "Each day now I find myself looking at software development in a whole new way."
3) "My code writing will change a lot after considering the threats we have covered."
4) "I surely watch out for the "Public Enemy"--buffer overruns, every time using string library functions."
5) "This will add a security pass review to my code in the future when I will be working with valuable information."
6) "Data validation has been an area where I've practiced increased vigilance in recent years. Now more than ever, I'm thinking of how to build containers to handle validation and verification before the data is passed on for further processing."

C. Learning Beyond the Classroom

The following statements indicate students applied their learning outside the classroom:

1) "Additional education and training will be necessary to stay on top."
2) "I re-analyze past projects and realize errors that I've made in the past and how I could improve that code now or how I could make the application more secure and robust if I were to do it again."
3) "Actually, we have already applied several security techniques we learned in the course in an application we developed for the programming methods class. ...after the vulnerabilities presentation, we went back to our project's code and added a lot of input validations."
4) "I use this new method (threat modeling) to...decompose application architecture and discover its vulnerabilities."
5) "I am planning to continue my studies in secure socket programming and other network security aspects."

In addition to the in-class evaluations of student progress, students were required by the School to complete a

ISBN 1-933510-99-4/05/$15.00

student survey. Student evaluations were 20% higher than the average for other department courses.

Another indicator of success: two students committed to pursuing doctoral studies, enrolling following graduation. One other took a special study with one of the authors to pursue interests in network security.

IV. Lessons Learned

These are lessons learned from this experience:

1) Applying pedagogical models can be useful in designing IA curriculum that produces desired outcomes and predictable results.
2) The Secure Code course developed using this approach appears to have produced independently functioning students.
3) Applying the concept of learning levels to the design of assignments allowed the instructor to guide students progressively towards independently adopting new techniques.
4) Challenging students can change long-held software development habits to include security awareness if led through a process of discovery.
5) IA courses like Secure Code can be developed with significant contributions from resources readily available in the security community.

It was also most helpful to be able to draw on the authors of the text to guest lecture for the course. They participated beyond the typical guest lecturer by providing oversight to the course materials developed and input on the assignments. In exchange, all course artifacts were delivered to them at Microsoft to allow dissemination to other interested parties. [5] An Australian university, and one in Tacoma, have already used these materials.

V. Conclusions and Future Work

The results of introducing IA curriculum at Seattle University were excellent. The Secure Code course has become a permanent part of the curriculum. As of 2005-2006 the University will officially offer a concentration in information assurance in its Software Engineering Master's program.

Additional measures of course success are planned in future course evaluations. A pre- and post-test of students' ability to recognize code vulnerabilities is one possibility. It also would be interesting to follow up after graduation, to determine if graduated students are still using secure code techniques and if they have found this course and/or the information assurance track valuable.

VI. References

[1] Endicott-Popovsky, B.E. and Frincke, D. (June, 2003). "A Case Study In Rapid Introduction of Computer Security Curricula," *CISSE 7th Colloquiam*. Washington, D.C.

[2] Endicott-Popovsky, B.E., Frincke, D., Popovsky, V.M., "Designing A Computer Forensics Course For an Information Assurance Track," in *Proceedings of CISSE 8th Annual Colloquiam* June 2004, USMA, West Point.

[3] Endicott-Popovsky, B.E. and Frincke, D. (March, 2004). "A Case Study in Rapid Introduction of an Information Assurance Track into a Software Engineering Curriculum," *IEEE Computer Society Press 17th Conference on Software Engineering and Training*. Norfolk, Virginia.

[4] Gates, Wm. "Trustworthy Computing," Memo published January 15, 2002. Microsoft Corporation, Seattle, WA.

[5] Endicott-Popovsky, B.E. "Writing Secure Code." Presentation at: *Microsoft Academic Days in Silicon Valley*, MS Research: Silicon Valley, Oct. 31, 2004.

[6] Kuzmina, U. F. (1972). *Fundamentals of pedagogy of higher education*. Leningrad: Lenizdat.

[7] Bespalko, V. P. (1977). *Fundamentals of theory of pedagogical systems*. Voronege: Voronege University.

[8] Talizina, N.F. (1975). *Management of the Learning Process*. Moscow, Russia.

[9] Roginsky, V.M. (1990). *Alphabet of Pedagogical Work*. Moscow, Russia: School of Higher Education.

[10] Hutton, G. "Backward curriculum Design Process" Viewed Online 5/1/2003 at http://www.g4v.com/~glen.hutton/ED3601/BackwardDesignFeb11_03.pdf

[11] Bloom, B.S., Mesia, B.B. and Krathwohl, D.R. (1964). *Taxonomy of Educational Objectives*. New York: David McKay.

[12] ACM/IEEE. "Computing Curricula 2001 Computer Science Volume December 15, 2001" http://www.sigcse.org/cc2001/.

[13] NSA. "National IA Education & Training Program," Retrieved from the Web: http://www.nsa.gov/ia/academia/cnsstesstandards.cfm

[14] Irvine, C., Chin S-K., and Frincke, D. “Integrating Security into the Curriculum.” *Computer*. 31(12). 12/1998. pp. 25-30.

[15] Howard, Michael and Brian LeBlanc. (2003). *Writing Secure Code*. Redmond, WA: Microsoft Press.