An Online Learning Approach to Information Systems Security Education

Norman Bier, Marsha Lovett, and Robert Seacord, *Carnegie Mellon University*

Abstract –The demand for information systems security education has never been higher, while the availability of high-quality information systems security instruction and of well-qualified instructors are both extremely limited. Meeting the demand requires converting teaching from an individual activity to a community-based research activity. As a result, Carnegie Mellon University's Open Learning Initiative and the Software Engineering Institute's CERT® Program have collaborated in the development of an online secure coding module that exemplifies how to capture expert content, ensure high-quality learning, and scale to meet rapidly growing demand. This paper describes this effort and how high-quality information systems security instruction can be scaled to meet existing and projected demand.

Index terms – Information Systems Security, Secure Coding, Distance Education, Education, Best Practices.

I. Introduction

Current and projected demands for software developers with skills in creating secure software systems demonstrate that, among other things, there exists a clear need for additional capacity in secure coding education. The National Strategy to Secure Cyberspace contains a specific priority to create a national cyberspace awareness and training program [1]. That priority recognizes two of the barriers to the improvement of cybersecurity as "a lack of familiarity, knowledge, and understanding of the issues" and "an inability to find sufficient numbers of adequately trained…personnel to create and manage secure systems" [1]. One of the National Strategy's major initiatives is to "foster adequate training and education programs to support the Nation's cybersecurity needs" [1].

Increased capacity can be addressed, in part, by an increase in the productivity and efficiency of learners, that is, moving ever more learners ever more rapidly through course materials. However, the need for throughput is matched by the need for quality. Students must be able to apply what they have learned and be able to learn new things. Effective secure coding requires a balance between high-level theory, detailed programming-language expertise, and the ability to apply both in the context of developing secure software. Educating software developers properly requires great expertise. While this expertise does exist, it tends to reside in individuals and organizations that are isolated from one another. These pockets of excellence, effective within their spheres, do not scale to meet the national demand. Furthermore, their isolation often means that even when practitioners do achieve significant improvement in the effectiveness of their instruction, this success is not shared or systematized.

Just as contemporary models for software development have rejected the isolated "hero programmer" in favor of a team- and process-driven engineering approach, current best practices in educational technology and research in learning science point away from the solo educator. In the words of Herbert Simon, "Improvement in post-secondary education will require converting teaching from a 'solo sport' to a community based research activity."[1]

II. Background

Carnegie Mellon University's (CMU) Open Learning Initiative (OLI) builds learning environments that are dynamic, flexible, and responsive. Design and implementation are data driven because all learning activities in OLI courses are, with the student's permission, digitally recorded in considerable detail. This enables the system to adapt to what the learner is doing and, over time, informs course refinements and overall system improvements. OLI constitutes a new approach to course development, evaluation, and improvement methodologies. It offers specific web-based learning interventions that can improve both the productivity and the quality of instruction. Using intelligent tutoring systems, virtual laboratories, simulations, and frequent opportunities for formative assessment with expert feedback, OLI takes full advantage of advances in the cognitive and learning sciences.

OLI is an open educational resources project that began in 2002 with a grant from The William and Flora Hewlett Foundation. Like many open educational resources projects, OLI makes its courses openly and freely available. OLI courses are much more than collections of

Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

[1] Remarks made at a lecture given at Carnegie Mellon University, April 1996.

ISBN 1-933510-96-X/$15.00

material created by individual faculty to support traditional instruction. Rather, OLI strives to create courses that *enact instruction*; these courses provide structure, information, activities, and feedback, all arranged so that students can learn, even if they do not have the benefit of an instructor or classmates. The same features that can effectively support an independent learner can also be leveraged to support classroom instruction; instructors and course authors have successfully taught OLI courses, and current OLI research efforts continue to find ways to improve the experience and outcomes for students and instructors using a blended-learning approach.

CERT is part of the Software Engineering Institute (SEI), a federally funded research and development center at CMU in Pittsburgh, Pennsylvania. Among other security-related activities, CERT regularly analyzes software vulnerability reports and assesses the risk to the Internet and other critical infrastructure elements. Researchers at CERT have observed, through an analysis of thousands of vulnerability reports, that most vulnerabilities stem from a relatively small number of common programming errors [2]. By identifying insecure coding practices and developing secure alternatives, software developers can take practical steps to eliminate known code-related vulnerabilities.

As part of the CERT® Secure Coding Initiative,[2] CERT identifies common programming errors that lead to software vulnerabilities. Then it establishes and publishes well-vetted secure coding standards that mitigate known problems. Members of the Secure Coding team educate students and professionals by working with individual developers and software development organizations. The objective is to reduce the number of vulnerabilities being deployed to a level where they can be successfully managed by existing vulnerability handling teams.

The remainder of this paper describes the course design approach followed by OLI, how this approach was applied to the creation of a secure coding course module, and a pilot offering of the module in an undergraduate computer science class at CMU. We also describe a proposal for community-based information systems security instruction.

III. Course Design Approach

The traditional process of having every instructor design his or her own course is inefficient; what may be less obvious is that the traditional course design and delivery process is often ineffective. Much is known about student learning and effective course design, but translating scientific results from the learning sciences into effective instruction requires significant knowledge, expertise, and effort. Such an effort by one faculty member for a single class is rare and, even when accomplished, typically has an impact on comparatively few students. In contrast, each OLI course is designed by a multidisciplinary team including a learning scientist, faculty content expert(s), human-computer interaction expert, and software engineer, so that these different areas of expertise can be brought to bear in the course design. The team's goal is to create and refine the course so that it can be effectively used by many instructors and learners. This type of dynamic, team-based course development process is an important characteristic of the open educational resource (OER) approach to course development [14].

OLI course development begins with a study of the teaching and learning challenges in the domain under development. This study includes literature reviews, reviews of existing artifacts of student learning, classroom observations, lab studies, or classroom-based studies. The design team then articulates a set of student-centered, measurable learning objectives—descriptions of what students should be able to do by the end of the course. These learning objectives then guide and inform the design of instructional activities and assessments that will support students in achieving those objectives. That is, the course designers deliberately create instructional activities and assessments that are well aligned with each other and with the articulated learning objectives, producing more effective learning experiences (see Figure 1).

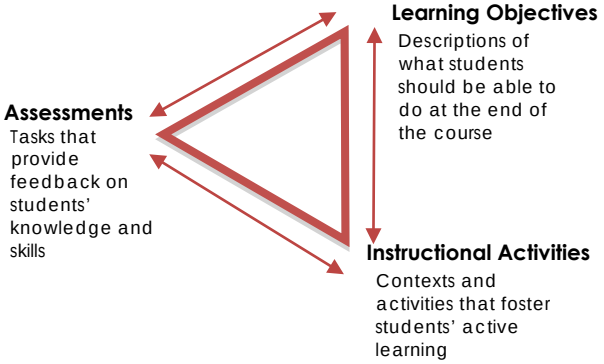

Figure 1. Course design triangle.

Interestingly, in OLI courses, the distinction between instructional activities and assessments is blurred because instructional activities not only support learning but also offer feedback on students' knowledge and skills (to both the student and the instructor), and assessment activities not only evaluate students' developing knowledge and skills but also offer opportunities to learn. Consequently, one of the most powerful features of OLI learning environments is the ongoing presence of embedded formative assessment and feedback throughout the learning process.

[2] https://www.cert.org/secure-coding/

ISBN 1-933510-96-X/$15.00

Instructional activities and assessments in OLI capitalize on the computer's capability to display digital images and simulations, promote students' interactive engagement, and collect data on students' interactions with the system. In particular, OLI benefits from inheriting some of the best work done in the area of computer-based tutoring by CMU and University of Pittsburgh faculty. Many OLI courses feature Cognitive Tutors [3] and *mini-tutors* that give students feedback within the problem-solving context. A Cognitive Tutor is a computerized learning environment whose design is based on cognitive principles and whose interaction with students is modeled after that of a human tutor, that is, making comments when the student errs, answering questions about what to do next, and maintaining a low profile when the student is performing well. This approach differs from traditional computer-aided instruction in that it offers context-specific assistance to students *throughout* the problem-solving process rather than only giving feedback on the final answer. Consequently, a hallmark of all OLI courses is the frequent opportunity for students to apply what they are learning through problem solving and receive individualized feedback on their work.

OLI's feedback to students includes corrections, suggestions, and cues that are tailored to the individual's current performance and that encourage the student to revise and refine their performance. Many learning studies have shown that students' learning improves and understanding deepens when they receive timely and targeted feedback on their work [4, 5, 6, 7]. The best learning outcomes occur when feedback comes as soon as possible after the student's response but not before the student is ready to revise his or her understanding.

OLI also offers feedback to instructors who are teaching with OLI in *blended* mode, that is, where the instructor assigns students to work through a segment of the OLI course and also has some face-to-face class time. Such instructors can use a tool called the Instructor's Learning Dashboard to see, at a glance, where students are succeeding and where they are struggling. This tool works by collecting and analyzing data from students' interactions in the course to estimate the students' current knowledge state for each learning objective. With this information in hand, an instructor can identify what material requires significant remediation versus what can be reviewed rather quickly, thereby adapting his or her instruction to students' needs.

The combination of this rich feedback to instructors and the context-sensitive feedback given to students has enabled OLI courses to show significantly greater learning gains compared to traditional courses [8, 9, 10]. For example, in one series of studies, students learning with OLI-Statistics completed a full semester's worth of material in half the time and achieved greater learning gains than students in a traditional class [7].

Beyond the immediate benefit to students and instructors, OLI also leverages system-generated student performance data to provide feedback to the course development team, who can then use this information to enact evidence-based, iterative improvements to the course materials. Similar types of information are also provided to the learning science community, contributing to the development and further refining of new knowledge about human learning. In both of these contexts, the open approach is an essential component of improving the quality of education. These feedback loops are dependent upon courses being used by a large number of students with varied background knowledge, relevant skills and future goals. This large population of learners is one benefit that open access provides, and is one reason that the OER approach is able to improve the quality and effectiveness of courses [14].

In sum, OLI is much more than a technology. It is a set of strategies for course design, development, delivery, and evaluation. OLI development teams use learning science research results to inform course design and use learning science research methods both to unpack the cognitive tasks and to design the instructional interventions. Moreover, OLI courses hold the collective memory of what works and does not work, so time and resources are spent on improvements rather than on reinventing existing successes or failures. That is, OLI collects data to provide feedback loops to students, instructors, and course design teams for continuous evidence-based improvement.

IV. Designing an OLI Course for Secure Coding

The design team began the task of creating an OLI course in secure coding by selecting the topic of integral security, that is, security related to the use of integers in C language programs. Integers are ubiquitous in C language programs, but even seasoned software development professionals have a poor understanding of their behavior. Consequently, vulnerabilities resulting from an incorrect understanding of integral behavior in C language are commonplace. Material for the course module on integer security was derived from the book *Secure Coding in C and C++* [10] and developed by a team of software security and C language programming experts, including the current chair of the ANSI-C Standards Committee (now INCITS PL22.11).

The design team then proceeded to articulate a set of learning objectives to specify what students should be able to do by the end of this course module. Note that, beyond being stated in student-centered terms, these learning objectives specify action-oriented, measurable outcomes. This helped the course design team ensure that

ISBN 1-933510-96-X/$15.00

the module's instructional activities and assessments were designed to help students achieve the module's objectives. The learning objectives for the module are:

- Explain and predict how integer values are represented for a given implementation.
- Reason about type ranges.
- Identify error conditions.
- Select appropriate type for a given situation.
- Predict how and when conversions are performed and describe their pitfalls.
- Understand integer types, representations, and conversions.
- Recognize when implicit conversions and truncation occur as a result of assignment.
- Programmatically detect erroneous conditions for assignment, addition, subtraction, multiplication, division, and left and right shift.
- Identify security flaws and vulnerabilities resulting from erroneous integer operations.
- Explain how vulnerabilities from erroneous integer operations can be exploited.
- Identify applicable mitigation strategies, evaluate candidate mitigation strategies, and select the most appropriate mitigation strategy (or strategies) for a given context.
- Apply mitigation strategies to reduce the introduction of errors into new code or repair security flaws in existing code.

To develop mastery of the skills required by these learning objectives, students need goal-directed practice and targeted feedback on integrating the component skills in progressively more realistic contexts. Consequently, multiple activities were designed to support each learning objective, starting with more basic comprehension checks and progressing to more complex, contextualized problems.

A sample of an advanced activity designed to support the learning objective to "programmatically detect erroneous conditions for division" is the following:

> What is wrong with the following test for checking overflow in the signed multiplication of **a** and **b**?

```
signed int a = /* some value */;
signed int b = /* some value */;

if ((a > 0 && b > 0 && a > INT_MAX / b) ||
    (a > 0 && b < 0 && a > INT_MIN / b) ||
    (a < 0 && b > 0 && a < INT_MIN / b) ||
    (a < 0 && b < 0 && a < INT_MAX / b)) {
  /* handle error condition. */
}
```

> A. A false negative could occur (that is, a multiplication that overflows can pass this check).
>
> B. In the worst case, four division operations are required.
>
> C. Undefined behavior can occur as the result of a division operation.
>
> D. A false positive could occur (that is, a valid multiplication could be flagged as an error).

If the student is unsure where the problem exists in this code, the student can request the following hint:

> Consider the case where **a > 0** and **b == -1**.

This focuses the student's attention on a particular range of values. If this is insufficient, the student can request further guidance:

> What is the result of **INT_MIN / -1**?

Here the student is asked to consider the particular case where the dividend is **INT_MIN** and the divisor is **-1**. If this hint is still insufficient, a third and final level of guidance is available:

> What is the behavior if the quotient of **a/b** is not representable?

Feedback is also provided for both correct and incorrect responses. For example, if the student incorrectly selects "B. In the worst case, four division operations are required," the student will receive the following feedback:

> Incorrect. Because of the short-circuit evaluation of the **&&** operator, only one division operation is required.

On the other hand, if the student selects the correct answer, he or she still receives feedback to ensure this answer was selected for the correct reason and to reinforce the material:

> Correct! If **a > 0** and **b == -1**, the test **a > INT_MIN / b** results in undefined behavior when using two's complement representation because the result of **INT_MIN / -1** is not representable as a **signed int**.

A demo of the Secure Coding in C course module can be publicly accessed at https://oli.web.cmu.edu/ using the course key: *scode-demo.*

V. Secure Coding Education at Carnegie Mellon University

The National Security Agency designated CMU as a Center of Academic Excellence in Information Assurance Education in 2001 in recognition of its significant contribution to meeting the national demand for information assurance education, developing a growing number of professionals with information assurance

ISBN 1-933510-96-X/$15.00

expertise in various disciplines, and, ultimately, protecting the national information infrastructure. Many university departments participate in activities related to information assurance and computer security.

For example, the Computer Science Department at CMU has offered CS 15-392 "Secure Programming" as a computer science elective since 2007. The DHS-sponsored Software Assurance Curriculum Project includes this course as an example of an undergraduate course in software assurance that could be offered in conjunction with a variety of programs [11]. CMU's Information Networking Institute has also offered 14-735 "Secure Software Engineering" in its Master of Science in Information Technology Information Security Track (MSIT-IS). Both courses partially map to the National Training Standard for Information Systems Security (INFOSEC) Professionals 4011 national training standard [13]. The topic of "Integer Security" is a common module to both these courses. As a result, the OLI course module on integer security is appropriate to both.

VI. Pilot

An initial version of the OLI Integer Security module was completed in January of 2011 and used during the 2011 spring semester offering of CS 15-392. The previous semester's offering of 14-735 served as a loose comparison. The integer module used in the pilot consists of 43 HTML pages and includes 27 instructional or assessment activities.

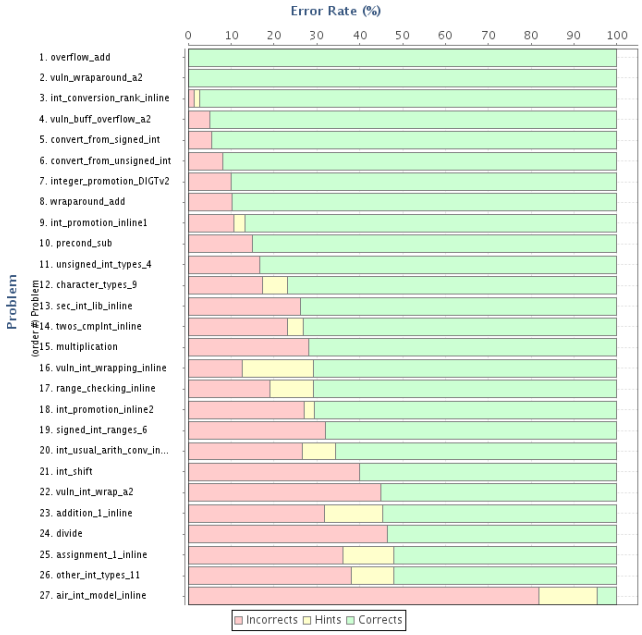

Table 1. Student performance.

On average, the students reported taking 6 hours to complete the module. System measures included[3]:

Average time engaging with course material/activities: 4.29 hours (Min: .57 hours, Max: 16 hours)
Average number of sessions per student: 3.85 sessions
Length of average session: 1.13 hours

In addition to time spent taking the online course, four 50-minute class sessions were devoted to the material, less than half the class time normally consumed by this topic.

During this pilot, 22 of 23 registered students worked on the module, completing 92 percent of the assessments. On average, the number of errors students made plus the number of hints they requested per question—also called *average help needed*—was 0.3, but the range of this metric was 0 to 2.57. Overall, seven questions' *help needed* was greater than 0.75. We take this cutoff as a good threshold to examine the results of the assessment to determine if a question was deficient or if the students simply required further instruction in this area. Overall student error rates per problem are shown in **Error! Reference source not found.** The green bar represents the proportion of students who answered that question correctly. The yellow bar represents the proportion of students who requested a hint; and red bar represents the proportion of students who answered the question incorrectly.

A review of the question at the high end of the metric determined that it was deficient and required rework. On the other hand, the following question, which had 45 percent correct responses out of 22, showed no obvious signs of deficiency:

> The following function accepts three arguments: `elem_count`, `p_max`, and `p_current`. The `p_max` argument points to one past the last element of an array object, and the `p_current` argument points to an element in the same array object. The `test_ptr` function determines if there are at least `elem_count` elements following `p_current` in the array.

```
int test_ptr(size_t elem_count,
    int *p_max, int *p_current) {
  T subscript_diff = p_max - p_current;
```

[3] Time is captured from start of a session (log-in) to the last action of a session. This means for a student who opens a page and spends 20 minutes reading, but takes no subsequent system-captured action after reading, that 20 minutes would not be included in our total. This is a possibility for 21% of sessions. To account for students who open a window, leave their computer, then return (potentially hours later) internal session gaps that are larger than 60 minutes were filtered (these were rare—15 incidents, representing an additional 7.25 hours).

ISBN 1-933510-96-X/$15.00

```
  if ( (p_max > p_current) &&
    (subscript_diff > elem_count) ) {
    return 0;
  }
  return -1;
}
```

Which type `T` should be used?

```
unsigned int
int *
size_t
ptrdiff_t
```

The students who answered this question incorrectly all chose `ptrdiff_t`, revealing a common misconception. Indeed, seeing this reasonable error, the instructor recognized the issue as a good candidate for in-class discussion.

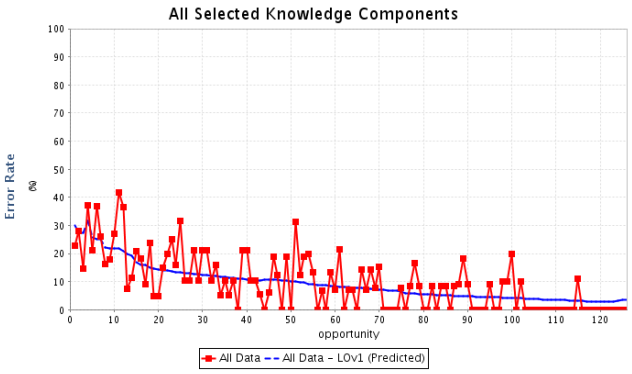

Figure 2. Learning curve.

Undergraduate computer science students who participated in the pilot correctly answered 76 percent of integer-related questions on the midterm. Graduate students[4] who spent significantly more class time covering the material (480 minutes as opposed to 200 minutes) achieved the same results on the previous semester's final. It is expected that, with further improvements to the online course material and the corresponding in-class instruction, future course deliveries will produce further learning improvement. This is a further advantage of the online format in that it is not necessary to reprint a text book to get a new set of materials/content incorporated into the course.

VII. Community-based Instruction

One goal of the OLI is to develop exemplars of high-quality, online courses that support individual learners in achieving the same learning goals as students enrolled in similar courses at CMU. Although OLI courses were originally designed to support individual learners, instructors inside and outside of CMU increasingly use OLI courses to complement their instructor-led courses. OLI courses can help instructors address the challenges of the increasing variability in their students' background knowledge, relevant skills, and future goals.

In addition to OLI courses, CMU has also developed an autograding system used to automate the evaluation of student programming assignments [12]. The current Autolab system was developed by Hunter Pitelka, David Kosbie, and David O'Hallaron in 2010. It is hosted at CMU and consists of a Linux front-end machine with an 8-Tbyte RAID array and 10 Linux back-end autograding machines. The front end runs a Web server written in Ruby on Rails, a MySQL database, a Tashi cluster manager that manages virtual machines (VMs), and a Tango daemon that provides the interface between the Web server and Tashi. The back-end machines run KVM virtual machines on behalf of the front end. The design allows untrusted code to be run in network-isolated VMs without (virtual) network cards.

Autograding has improved the quality of the learning experience for students at CMU [12]. Labs are no longer limited by the instructor's ability to grade them, but rather by imagination and cleverness in developing autograding software. Autograding also expands the potential reach of our labs beyond CMU. A hosted Autolab service could provide autograding of a shared repository of labs to all of the world's universities.

The capacity of CERT and other organizations involved in information systems security research, development, and education to produce educational material is extremely limited. This reflects Herbert Simon's idea that improvement in post-secondary education requires a community-based research approach.[5] OLI courses are open and free, and any university, college, or other learning institute can offer them as credit-earning courses, as CMU is currently doing. The courses can be clearly branded with the names of the contributing instructor(s) and their organization, adding to the reputation of both. By establishing a library of high-quality course content and autograded labs with a scalable delivery platform, the demand for high-quality information systems security education can be satisfied.

VIII. Conclusions and Future Plans

The Integer Security module presents a proof of concept for developing an online course for secure coding using the OLI approach to course development.

[4] The exam questions were not identical or matched for difficulty—they were simply matched for content.

[5] Remarks made at a lecture given at Carnegie Mellon University, April 1996.

ISBN 1-933510-96-X/$15.00

We successfully provided students with material and activities that support a clear set of learning activities, while giving tailored hints and feedback. We also demonstrated via a pilot study that the course is effective in terms of students' performance results and time spent. This pilot study has also provided insights into the aspects of the course that are most effective and those that still result in student difficulties and misconceptions. This information will be used in the next iteration of course development to refine existing materials and create additional activities to target the skills and outcomes that students failed to achieve.

We now have a procedure and a platform for building other modules, further evaluating the effectiveness of this approach, and incorporating other tools to support students' learning of this material. Consequently, CERT has begun development on a subsequent module on C language pointers and will develop other modules based on available resources.

Another goal is to incorporate an intelligent tutor that can compile, analyze, execute (securely), and test code submissions from students and provide timely, substantive, and targeted feedback. This would allow for the analysis of student submission using a variety of automated analysis techniques, including static and dynamic analysis, model checking, and traditional unit testing. This goal may be accomplished by integrating CMU's autograder system with OLI and developing autograded exercises for the secure coding course.

Beyond CMU, the Department of Computer Science at Stevens Institute of Technology has also used the secure integer module. Matt Bishop, from the Department of Computer Science at the University of California, also plans to use the integer module in his undergraduate computer security course starting March 24, 2011. The SEI is also using the course module in two course deliveries to professional audiences in March 2011.

IX. Acknowledgements

Doug Gwyn, David Keaton, Philip Miller, David Svoboda, and Tim Wilson all contributed to the development of the secure coding integer module content. Pennie Walters was responsible for technical editing of the course module and Paul Ruggiero for technical editing of this paper. Alexandra Drozd assisted in the course implementation.

X. References

[1] U.S. Department of Homeland Security (2003). *The National Strategy to Secure Cyberspace.* U.S. Department of Homeland Security.

[2] Seacord, R. C. (2005). *Secure Coding in C and C++ (SEI Series in Software Engineering).* Addison-Wesley Professional.

[3] Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). "Cognitive Tutors: Lessons Learned." *Journal of the Learning Sciences 4*: 167-207.

[4] Hattie, J., & Timperley, H. (2007). "The power of feedback." *Review of Educational Research, 77*(1), 81-112.

[5] Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010). *How Learning Works: Seven Research-Based Principles for Smart Teaching*. San Francisco, CA: Jossey-Bass.

[6] National Research Council. (2001). *Knowing What Students Know: The Science and Design of Educational Assessment.* Washington, DC: National Academy Press.

[7] National Research Council. (2004). *How People Learn: Brain, Mind, Experience, and School.* Expanded Edition. Washington, DC: National Academy Press.

[8] Lovett, M., Meyer, O., & Thille, C. (2008). "The Open Learning Initiative: Measuring the Effectiveness of the OLI Statistics Course in Accelerating Student Learning." *Journal of Interactive Media in Education*.

[9] Schunn, C. D., & Patchan, M. (2009). *An Evaluation of Accelerated Learning in the CMU Open Learning Initiative Course "Logic & Proofs."* Technical Report by Learning Research and Development Center, University of Pittsburgh.

[10] Steif, P. S., & Dollár, A. "Study of Usage Patterns and Learning Gains in a Web-based Interactive Static Course." *Journal of Engineering Education 98*, 4 (2009): 321-333.

[11] Mead, Nancy R., Hilburn, Thomas B., & Linger, Richard C. (2010). *Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines* (CMU/SEI-2010-TR-019). http://www.cert.org/mswa/

[12] Milojičić, Dejan. "Autograding in the Cloud: Interview with David O'Hallaron." *IEEE Internet Computing Magazine* January/February 2011 (Vol. 15, No. 1): 9-12.

[13] NSTISSI-4011 - INFOSEC Professionals, National Training Standard, 1994.

[14] Plotkin, Hal (2010). *Free to Learn: An Open Educational Resources Policy Development Guidebook for Community College Governance Officials*. San Francisco, CA: Creative Commons. http://wiki.creativecommons.org/Free_to_Learn_Guide

ISBN 1-933510-96-X/$15.00