Q: Can the security, usability, and software engineering life cycles be blended into a comprehensive and effective single process for developing software?

A: Yes, by shaping curriculum models for early exposure

Heather Richter Lipford **Dugald Ralph Hutchings**

We contend that the one of the difficulties in blending security, usability, and software engineering processes arises from the educational experience typical for undergraduate students in computer science, computer information systems, and other IT-oriented disciplines. In early programming and technology courses, instructors often face difficult decisions of what basic topics to include and exclude from the course, let alone whether to address "other" topics of security and usability. Students are then unlikely to see these topics until later, more advanced courses. This early lack of exposure leads to an attitude among software developers that security, usability, and software engineering are separate, rather than interrelated, concerns. In turn, this may contribute to developers' over-reliance on others, such as designers or a software security group, to address those issues. Conversely, we believe that if students are exposed to these topics early, then those attitudes should reverse.

A main challenge of implementing an *early exposure* strategy is selecting security and usability topics that relate to, and do not interfere with, the important primary content of the basic building blocks of software development. Yet we believe that the computing community could identify the most easily adopted, adapted, or understood elements of security or usability that would be a natural fit with existing course material. Take for example the topic of input validation, in which software code first checks the user-provided input prior to processing or analyzing it. This topic is not only a sound programming topic, but also can be treated as a security topic (helping to prevent unanticipated and possibly dangerous program behavior) and as a usability topic (altering the interface when provided input was invalid, or designing the software to prevent erroneous input in the first place) even in early courses. If the instructor has the resources to *easily* describe such examples from the perspective of security and usability, then the student receives the exposure. Efforts at such integration into early curricula are occurring with respect to security, usability, or software engineering separately, but integrated approaches are needed.

In addition to identifying specific integrated topics, we need to develop examples, assignments, and tools that embody those topics. For example, eliminating security bugs from examples in programming textbooks, and providing example applications with usable security. Code comments or additional course resources should highlight the variety of implications of those examples. This will allow students to learn good habits early on, even if they are not fully taught the complex concepts behind them until later.

While early exposure to security and usability topics within the development process should facilitate positive student attitudes about an integrated model of software engineering, it is unlikely enough to truly change the view of "standard" SDLC elements. We believe however that successful early introduction of security and usability topics will encourage educators to include these topics *throughout* the curricula and perhaps even influence the model curricula of ACM, AIS, and other professional organizations. We briefly highlight two examples of existing curricula that provide students with opportunities beyond the initial classes.

At Bowling Green State University (where one author held a position for two years), all undergraduate computer science students must enroll in an upper-level *Usability Engineering* course that emphasizes an integrated approach to software engineering (i.e., the course is neither a human-computer interaction course nor a software engineering course). The course involves inspection of traditional software engineering and usability design life-cycles and highlights the many areas where the approaches overlap or complement one another. It is clear from experience working with the students that program graduates are sensitive to the needs of users and create software that exhibits better usability than students who are otherwise not exposed to the topic. Simply stated, the students know how to create usable software.

Whereas BGSU has a CS curriculum that explicitly integrates usability and software engineering in a required class, Elon University's computer information systems students experience elements of usability and security in a wider variety of classes. The security-oriented class includes a discussion of the level of security that is appropriate for a proposed system and the tradeoffs made between system security and user memorability. The usability-oriented class employs a project-based method in which students build Web sites for real community organizations that allow organization members and interested community members different levels of access to the site. Students consider the appropriate tasks for each type of user and use role-based security methods to implement and test Web sites in addition to performing usability testing, all as part of a typical design life-cycle. While the Elon and BGSU majors and approach differ, in both cases students leave the institutions with a SDLC model that incorporates usability or security.

UNC Charlotte is examining a different approach in addressing early exposure of secure programming concepts. We are developing the Assured Software IDE plug-in (ASIDE) to warn programmers of potential vulnerabilities in their code, and assist them in addressing these vulnerabilities. For example, when input is detected, the tool would alert the programmer of the need to do input validation, allow the user to interactively choose the type of input, and automatically generate validation code. This tool will serve as an educational opportunity and reinforcement of security concerns during any programming activities throughout a student's education. We believe that there likely exist similar opportunities to integrate other security, usability, and software engineering concerns into the tools students regularly use during their education, reinforcing these concepts both within and outside of the classroom.

Given the variety of software development process models taught and used in practice, we believe that there is not going to be only one way to integrate security, usability, and software development processes. However, by promoting the integration of these concepts early on, developers will in turn create and adopt effective and integrated processes and practices. Thus, in conclusion, we contend that by *starting* with the integration of small, easily understood, and broadly applicable security and usability principles, two ripple effects can result: (1) students will carry positive attitudes and expectations about incorporating security and usability into their work in industry; and (2) instructors will be more likely to evaluate the role of usability and security *throughout* the curricula of their departments and schools, thus allowing even deeper experiences for students.

About the authors

Dr. Heather Richter Lipford is an Assistant Professor in the Department of Software and Information Systems at UNC Charlotte. With a background in Human Computer Interaction, her primary area of research is currently usable security and privacy, focused in particular on improving personal privacy management through developing usable interaction and interface mechanisms. She teaches courses on Human Computer Interaction and Usable Security.

Dr. Dugald Ralph Hutchings is an Assistant Professor in the Computing Sciences Department at Elon University. His current research interests include usable security, visual analytics, and multi-display interaction. He teaches courses in introductory programming, user-centered Web design, and systems analysis & design.

Both Heather and Duke completed their graduate research work with the GVU Center at Georgia Tech. They are currently collaborating on the investigation of the educational uses of the ASIDE plug-in.