# Guest Editorial Preface

*Nancy R. Mead, CERT, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA*

*Ivan Flechais, Department of Computer Science, University of Oxford, Oxford, UK*

*Dan Shoemaker, Department of Computer and Information Systems, College of Liberal Arts & Education, University of Detroit Mercy, Detroit, MI, USA*

*Carol Woody, CERT, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA*

Since computer programming began for commercial applications 55 years ago, trillions of lines of code have been produced, much of which is still running in some computer somewhere. Nobody knows precisely how that vast landscape of code works, and that raises some very serious questions when it comes to protecting our national security and way of life. We know that unknown or undocumented code poses a major security risk and most code contains defects. So identifying the subset of those defects that are exploitable becomes a matter of finding one black grain of sand on a vast beach of white sand. As a result, it's very hard to find bugs and impossible to say with any certainty whether a given application is secure. In fact, it's much easier to just assume that all applications—no matter how rigorously developed—will contain some exploitable flaw that could cause a potential security problem.

We make this assumption in part because software is an invisible, yet highly complex, product. If software engineering as a discipline had been around as long as civil engineering, the engineering approach used for building a bridge, for example, would be applied consistently in a standard way, with universally accepted checkpoints. Unfortunately, our profession is not quite as mature as those that have been around for centuries, so we cannot have the same confidence level in the typical software engineering product as we might have in a civil engineering product.

While we could confirm, through testing and other development methods, correct software functionality, unless we engage in use of formal methods for every element of the software system, whether developed or acquired, and under every possible usage scenario, we cannot say with absolute confidence that it is impervious to a buffer overflow or command injection. Even worse, we cannot say for sure that a malicious object is not lurking somewhere in it. This is precisely why having some form of standard, objective measurement is so important to the security of software.

Accurate and trustworthy measures would allow the developer, or sustainer, to observe and judge the code, and standard measures of performance accumulated over a period of time would help make the coding process more effective and efficient. However, because software is intangible, it's hard to measure. For tangible civil engineering structures like a bridge, we have traditional, well-known, and, in many cases, ancient measurement processes, along with standard units of measure. For instance, we can answer a question like "What kind of

load can the structure sustain?" But for software, rigorous research is still needed to define both what to measure, how to measure it, and what that measure translates to in terms of meaningful information.

Previously, the only available answers to "how big is the software?" were management-type responses, such as a $5 million development or two-year project. Project managers accept such measures because those measures inform their decisions. But information that vague is not very helpful to software engineers, leaving them to make important decisions based on guesswork and creativity. Standard performance measures in software engineering define concrete, quantifiable code attributes. More importantly, persistent and commonly understood measures make the software engineering process more reliable.

Reliability is the probability that a system will operate without failure for a given time in a given environment. The given time in this definition may represent any number of actual data items, such as number of executions; number of lines of code traversed, or time of day. The challenge is to turn all of these potentially meaningful items into a standard and commonly accepted system of measurement. Measurement has gotten a lot of attention lately. Defects in code have always had obvious effects on a product's security, yet we still do not have a commonly accepted and standardized way to reliably characterize those defects. As a result, reliable, standard measurement data has become critical in the discussion of how best to produce secure code.

The right measure will ensure that engineers can monitor the right things. In effect, reliable measurement data makes the process and product visible to all participants, and that visibility helps establish assurance. Metrics research provides the link between development work and our need to truly understand the nature of the product. Nevertheless, we need to know how to describe software in ways that are meaningful to engineers, developers, and managers, while including practical considerations such as the nature of the collected data and the units of measurement. That need leads us to ask this question: "What is the current state of the art in the process of making software visible?" The aim of this edition is to open this discussion up to the profession. To that end, we present five views on how to make the software process and product more visible. Each view presents a different aspect of the problem and provides its own individual insight into the solution. We believe this sort of wide-ranging dialogue is the first step in overcoming existing hurdles to maturing the software assurance discipline.

The rest of this issue is organized as follows:

- "Principles and Measurement Models for Software Assurance" by Mead et al. presents an effective measurement model organized by seven principles that capture the fundamental managerial and technical concerns of development and sustainment.
- "Towards a More Systematic Approach to Secure Systems Design and Analysis" by Miller et al. presents research on measuring the variability in decision making among security professionals, with the ultimate goal of improving the quality of security advice given to software system designers.
- "A New Method for Writing Assurance Cases" by Matsuno and Yamamoto presents a new method for writing assurance cases and describes a preliminary experiment carried out on a web server demo system.
- "Analyzing Human Factors for an Effective Information Security Management System" by Alavi et al. identifies direct and indirect human factors that can impact information security.
- "Advancing Cyber Resilience Analysis Based on Metrics from Infrastructure Assessments" by Vugrin and Turgeon describes a hybrid infrastructure resilience assessment approach that combines both qualitative analysis techniques with performance-based metrics.

## ACKNOWLEDGMENT

We would like to acknowledge Valori Nicolai for her outstanding work in managing the review process and our colleagues who served as reviewers for this issue:

- Saeed Abu-Nimeh
- Chris Alberts
- Julia Allen
- Mark Ardis
- Matt Bishop
- Paul El Khoury
- Bob Ellison
- Shamal Faily
- Tom Hilburn
- Chris Lamb
- Seiya Miyazaki
- Adesh Rampat

*Nancy R. Mead*
*Ivan Flechais*
*Dan Shoemaker*
*Carol Woody*
*Guest Editors*
*IJSSE*

*Nancy R. Mead is a principal researcher with the CERT Program at the Software Engineering Institute (SEI). Mead is also a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at Carnegie Mellon University. She is currently involved in the study of security requirements engineering and the development of software assurance curricula. Mead has more than 150 publications and invited presentations, and has a biographical citation in* Who's Who in America*. She is a Fellow of the Institute of Electrical and Electronic Engineers, Inc. (IEEE) and a Distinguished Member of the Association for Computing Machinery (ACM). Dr. Mead received her PhD in mathematics from the Polytechnic Institute of New York, and received a BA and an MS in mathematics from New York University.*

*Ivan Flechais is a lecturer in the Software Engineering Program at the University of Oxford and has been researching computer security for over ten years. In particular, given that secure systems are frequently exploited by subverting their users, this involves researching how secure systems can be designed, implemented and tested to take human needs into account. Some recent research was conducted on webinos: an EU-funded project aiming to deliver a platform for web applications across mobile, PC, home media (TV), and in-car devices. Part of this work entailed applying tools and techniques from Usability, Security, and Requirements Engineering to ensure that users are accommodated in the design of the system. Other ongoing research involves exploring the usability of out-of-band authentication protocols for practical applications in different contexts of use, such as mobile device pairing, group authentication, and ad-hoc security.*

*Daniel P. Shoemaker, PhD, is Principal Investigator and Senior Research Scientist at UDM's Center for Cyber Security and Intelligence Studies. Dan is also a full time Professor and former Department Chair at University of Detroit Mercy. As the Co-Chair for the, National Workforce Training and Education Initiative he is one of the authors of the DHS Software Assurance Common Body of Knowledge (CBK). He also helped author the DHS IA Essential Body of Knowledge and he serves as a SME for the NIST-NICE workforce framework. Dan's doctorate is from the University of Michigan and within the State of Michigan he leads the International Cyber-Security Education Coalition. This Coalition covers a five state region with research partners as far away as the United Kingdom. Dan also spends his free time authoring some of the leading books in Cyber Security. His book* Cyber Security: The Essential Body of Knowledge*, is Cengage publishing's flagship book in the field. His first book,* Information Assurance for the Enterprise*, is McGraw-Hill's primary textbook in IA and is in use all over the globe. His next book* Engineering a More Secure Software Organization*, which is also published by Cengage, will be out next spring.*

*Carol Woody has been a senior member of the technical staff at the Software Engineering Institute since 2001. Currently she is the manager of the cyber security engineering team which focuses on building capabilities in defining, acquiring, developing, measuring, managing, and sustaining secure software for highly complex networked systems as well as systems of systems. Dr. Woody is an experienced technical researcher whose work has focused on government agencies, higher education, and medical organizations. She has helped organizations identify effective security risk management solutions, develop approaches to improve their ability to identify security and survivability requirements, and field software and systems with greater assurance. She holds a BS in mathematics from the College of William & Mary, an MBA from Wake Forest University, and a PhD in information systems from NOVA Southeastern University.*