As an industry matures, different things become important to it. An analogy with human growth offers an excellent example: what is important to adolescents differs from what is important to adults and, similarly, to an aging population. Our world of software is now more than half a century old, and although novelty attracts all players, the more mature the industry gets, the more vital are aspects of reliability and security. While many will agree with this sentiment, it might not be foremost in the conscious of every designer or architect. We’re rudely reminded of this every time we face a technological disaster. Part of the challenge is that reliability and security mean different things to different people.

Reliability is an often-overused term to describe everything from products that work well to products that are simply “good” or durable. However, reliability is integral to design, availability, maintainability, testability, diagnostics, prognostics and health management, integrity, security, quality, supportability, human engineering, and system safety.

In this issue of S&PE, we take a slightly more rigorous viewpoint than what you might find on Wikipedia (http://en.wikipedia.org/wiki/Reliability):

- In general, reliability (systemic def.) is the ability of a person or system to perform and maintain its functions in routine circumstances, as well as hostile or unexpected circumstances.

Reliability may refer to:

- Reliability (engineering), the ability of a system or component to perform its required functions under stated conditions for a specified period of time.
- Reliability (statistics), of a set of data and experiments
- Reliability (engineering) and epistemology
- Data reliability, a property of some disk arrays in computer storage
- Reliability theory, as a theoretical concept, to explain biological aging and species longevity
- Reliability (computer networking), a category used to describe protocols
- Reliability (semiconductor), outline of semiconductor device reliability drivers

Also according to Wikipedia (http://en.wikipedia.org/wiki/Cyber-physical_system), “a cyber-physical system (CPS) is a system featuring a tight combination of, and coordination between, the system’s computational and physical elements.”

Today, we find a precursor generation of cyber-physical systems in areas as diverse as aerospace, automotive, chemical processes, civil infrastructure, energy, health-care, manufacturing, transportation, entertainment, and consumer appliances. This generation is often referred to as embedded systems, where the emphasis tends to be more on computational elements and less on an intense link between computational and physical elements.

Thus, when you take these two definitions together, you’re talking about a complicated set of quality-based attributes (depend-
able, trustworthy, available, maintainable, fault-tolerant, robust, failure immune, secure, confidential, data integrity, safe, resilient, reliant, and several others) that are layered on top of a highly complicated base system; that is, you’re asking that a system’s computational and physical elements perform and maintain their functions in both routine circumstances and in hostile and unexpected circumstances. This challenge is daunting, but its outcome is desirable.

The list of issues in this rapidly growing field explodes as the footprint of embedded software increases in consumer and industrial products across numerous infrastructures. This aggressive growth is coupled with rising consumer expectations—consumers assume that each new feature will offer a richer user experience in their applications. However, this assumption is fragile—for example, patients might expect that their CAT scans and diagnoses will be emailed to only them and that the transfer process is verifiable online. Similarly, construction workers might assume that a do-it-yourself rented bulldozer is able to automatically steer clear of any hazardous material.

Program management of such requirements and expectations presents tremendous challenges. Interfacing closed systems with open ones must be conducted with attention to the reliability, security, and privacy compromises that occur. Compromises? Yes—each of the aforementioned attributes of cyber-physical systems carries both financial and technical trade-offs. Thus, an iPhone connected to a medical system or tractor trailer creates value and risk. Thus, we need tools, methods, architectures, protocols, and verification practices to achieve the next level of embedded systems capability with minimum compromises of safety, security, and privacy.

In the first article we selected for this special issue, “Kernel Service Protection for Client Security,” Hui Jun (Kevin) Wu explores a means to enhance system security at the kernel level by only allowing specific applications to invoke BIOS embedded services. The authentication process is implemented by asymmetric cryptography that uses one private key to encrypt the code that calls the kernel service and one corresponding public key to decrypt the code for checking caller integrity.

In the second article, “Embedded Software Assurance for Configuring Secure Hardware,” authors J. Ryan Kenny and Craig Robinson argue that the security embedded into processors isn’t useful unless the processors are correctly used and configured by the embedded software and development tools. They argue that it’s an institutional issue of correctly building security policies into hardware—just as it has been for years in software code. If done appropriately, the embedded software can greatly leverage the security benefits of these processors.

In addition to these articles, we conducted a roundtable with Sean Barnum (MITRE), Shankar Sastry (University of California, Berkeley), and John A. Stankovic (University of Virginia). In it, you’ll find an interesting discussion of the reliability and security trade-offs in embedded and cyber–physical systems. We hope you find this special issue unique: it’s still rare to find discussion between reliability and cyber–physical systems.

Acknowledgments
We might identify certain products in this document, but such identification doesn’t imply recommendation by the US National Institute of Standards and Technology or other agencies of the US government, nor does it imply that the products identified are necessarily the best available for the purpose. This article was not co-authored by Jeff Voas as a NIST employee; it reflects Voas’s opinions. The article doesn’t reflect the opinions of the US Department of Commerce or NIST.

Ram Chillarege is president of Chillarege Inc., with a consulting practice in orthogonal defect classification. He chairs the steering committee for the International Symposium on Software Reliability Engineering and is a fellow of IEEE. Contact him at info@chillarege.com.

Jeffrey Voas is a computer scientist at the National Institute of Standards and Technology. He is currently the president of the IEEE Reliability Society and a fellow of IEEE. Contact him at j.voas@ieee.org.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.