Toward Open-World Software: Issues and Challenges
pp. 36-43
Luciano Baresi, Elisabetta Di Nitto, and Carlo Ghezzi

A high degree of platform and language independence for a GUI library can be achieved through middleware built using reflection and controlled with XML-based specifications. Other technologies would also benefit from greater portability, such as the libraries associated with speech or gesture recognition and handwriting translation.

A growing trend is wearable computing devices, which require tangible user interfaces. The software that drives these devices will undergo the same rapid development cycle cell phones have, and new versions will constantly emerge for newer models. In this domain, the essence of the TUI can be encapsulated, enabling the drivers or toolkits to migrate in time.

Service Engineering: Linking Business and IT
pp. 45-55
Tiziana Margaria and Bernhard Steffen

Service-oriented computing radically alters the way business processes are modeled, realized, and maintained. Domain-specific services virtualize complex functions of the underlying business applications so that they can be loosely coupled to form transorganizational processes.

By separating the processes from the providers of individual functions, virtualization and loose coupling enable seamless cross-platform, -provider, and -domain process management. This horizontally seamless process management requires new tools to guarantee a vertically seamless realization of the modeled processes.

First Steps in the Verified Software Grand Challenge
pp. 57-64
Jim Woodcock

Industry has become increasingly aware that something must be done about software reliability. The 2005 International Technology Roadmap for Semiconductors states, “Without major breakthroughs, verification will be a nonscalable, show-stopping barrier to further progress in the semiconductor industry.”

What’s to be done about this state of affairs? Can we ever expect software to come with warranties of fitness for purpose? A growing number of academic and industrial researchers believe that the way to revolutionize the production of software is by using formal methods, and they also believe that doing so is now feasible.

What Can We Expect from Program Verification?
pp. 65-71
Michael Jackson

Software verification techniques could include both checking a given program against specifications contained in the text or provided in a separate document, and correctness by construction, in which a systematic and formal development procedure guarantees the developed program’s correctness or facilitates its verification in some way.

Proponents of the software verification Grand Challenge envision a world in which computer programs are always the most reliable component of any system or device that contains them. Verification tools and techniques can help make sure that the programmer builds the program right. They could also contribute to building the right program.

The ‘Grand Challenge’ in Informatics: Engineering Software-Intensive Systems
pp. 72-80
Manfred Broy

Information is essential for understanding our reality, and as the science of information and information processing, informatics contributes fundamentally to our comprehension of the world.

Informatics must respond to the challenges of providing the software needed for future embedded systems. In this case, software engineering will be integrated into systems engineering and, we expect, combined ultimately into a completely new discipline focused on the engineering of software-intensive systems.

What Have We Not Learned about Teaching Programming?
pp. 81-82
David Gries

Based on the premise that we need to look seriously at how we teach programming, the author contends that we can teach students how to think about the programming process far more effectively than we do now and that, in addition to increasing our effectiveness as teachers, doing so would make our entire curriculum more efficient.