Software Engineering on Internet Time

For Internet-style software development to succeed, developers must adapt and apply classical software engineering principles and practices to the Web environment.

Michael J. Lutz
Rochester Institute of Technology

Software engineering and Internet development seem to be worlds apart. Software engineering is often perceived as comprising ponderous bureaucratic processes appropriate to the design, development, deployment, and evolution of big, complex, slowly adapting systems. Internet applications, on the other hand, are characterized by terms like nimble, flexible, and adaptable. It would appear, therefore, that software engineering principles and practices have little relation to software development at Internet speed.

In this issue, we present three articles that challenge this perception by highlighting areas in which software engineering methods are not only applicable but also potentially critical to the success of rapidly developed, flexible software. The areas these articles cover—software architecture, metrics, and quality—are themselves well within the mainstream of contemporary software engineering practice. Each of these articles advocates an approach that addresses key issues in the world of intense high-velocity software development.

In "Accelerating Development with Agent Components," Martin L. Griss and Gilda Pour focus on the application of agent technology, in the context of XML and HTTP, as a mechanism for extending component-based software engineering to distributed Web-based applications. The goal is to achieve the flexibility and adaptability the market demands, but in the context of a robust and disciplined engineering approach that fits hand in glove with flexible, rapid application development.

"Software Engineering Metrics for COTS-Based Systems," by Sahra Sedigh-Ali, Arif Ghafoor, and Raymond A. Paul, addresses the problem of assessing the vulnerability of systems constructed mainly from commercial off-the-shelf components. COTS components are attractive because they provide leverage for a software development organization's unique abilities. These components can help define a modular software structure and allow the development staff to concentrate on areas where the organization can add true value to a system. However, COTS components are often available only in binary executable form, making the organizations that use them vulnerable to performance and reliability problems that the component providers' processes inject. This article provides a framework for measuring the effects of incorporating COTS components into an existing or proposed system, thus aiding engineers in making informed judgments about the development course to pursue.

Finally, in "How Internet Software Companies Negotiate Quality," Richard Baskerville and colleagues investigate the consequences of assuming that perfection is neither required nor even desirable in all circumstances. Instead, the authors consider various factors influencing the decision to deploy a software system. They explore the relationship between formal quality (conformance to requirements) and quality in use (fulfilling user expectations) to provide a methodological framework for determining when a product is "good enough" for its targeted market.

On the whole, these articles demonstrate that we can adapt and apply classical software engineering principles and practices to Internet-style software development. We hope this opens the door to more productive and fruitful exchanges between traditional software engineers and developers engaged in high-speed, high-stakes software development.

***Michael J. Lutz** is the Motorola Professor of Software Engineering at the Rochester Institute of Technology, where he heads RIT's undergraduate software engineering program. Contact him at mjl@cs.rit.edu.*

0018-9162/01/$10.00 © 2001 IEEE