The authors propose a methodology for describing the quality features of domain-specific software packages uniformly and comprehensively using ISO/IEC 9126-1 quality standards as a framework.

Managing Requirements for Medical IT Products
by Stewart A. Higgins, Maurice de Laat, Paul M.C. Gieles, and Emilienne M. Geurts, pp. 26–33. The authors have built a process, supported by tools, in which they capture and refine requirements as they come in. In this way, they pool requirements ready for allocation to release projects. They publish requirements valid for the next release project in a document for that release. In this process, requirements analysts work closely with requirements stakeholders.

Using Quality Models in Software Package Selection
by Xavier Franch and Juan Pablo Carvallo, pp. 34–41. The absence of structured descriptions of quality features and user quality requirements makes selecting the right software package difficult. The authors propose a methodology for describing the quality features of domain-specific software packages uniformly and comprehensively using ISO/IEC 9126-1 quality standards as a framework.

Using Split Capabilities for Access Control
by Alan H. Karp, Rajiv Gupta, Guillermo J. Rozas, and Arindam Banerji, pp. 42–49. The split-capabilities method offers improved scalability and revocation of privileges in controlling access to resources. Early releases of Hewlett-Packard’s e-speak product used this method to secure the system from a variety of common attacks. Split capabilities have the advantages of traditional capabilities without their limitations. The basic idea is to divide the capability into two parts: a handle to the resource being accessed and a handle to a separate resource representing the access rights being requested. Although such separation of name from authority is potentially problematic, this system brings these two elements together in the resource’s computing infrastructure.

Reducing Internet-Based Intrusions: Effective Security Patch Management
by Bill Bryczynski and Robert A. Small, pp. 50–57. Effective security patch management is critical to defending against Internet-based attacks. Despite its importance, surprisingly little has been written on the processes that organizations use to implement patch management. This article examines eight key practices intrinsic to effective, systematic, and repeatable patch management and proposes performance measures for evaluating it. The authors derived these key practices from industry pilot projects for improving patch management.

Misuse Cases: Use Cases with Hostile Intent
by Ian Alexander, pp. 58–66. A misuse case is the negative form of a use case; it documents a negative scenario. Its actor is an agent with hostile intent toward the system under design. The relationships between use and misuse cases document threats and their mitigations. Use- and misuse-case diagrams (shown in the article) are valuable in threat and hazard analysis, system design, eliciting requirements, and generating test cases.

C# and the .NET Framework: Ready for Real Time?
by Michael H. Lutz and Phillip A. Laplante, pp. 74–80. Microsoft’s next-generation platform, the .NET Framework, offers a new language, C#, which aims to offer the power of C++ with the ease of Visual Basic. In this article, the authors consider the application of C# and the .NET Framework to real-time systems. Their performance tests show that C and C# performance is comparable in some cases. However, they show that C# is not currently suitable for hard real-time applications and should be used for firm and soft real-time applications only with care.

Aspect-Oriented Development with Stratified Frameworks
by Colin Atkinson and Thomas Kühne, pp. 81–89. Aspect-oriented programming, one of the most promising separation-of-concerns technologies, focuses on concerns that cross-cut a chosen decomposition strategy. Despite its promise, however, AOP has so far failed to make the transition to mainstream software development. The authors describe a strategy, based on architecture stratification, that exploits the advantages of aspect-oriented development in model-driven and component-based frameworks.