Why Culling Software Colleagues Is Popular
by Peter Middleton, Ho Woo Lee, and Shahrukh A. Irani, pp. 28–32. Cutting staff is generally seen as an unpleasant duty to be carried out when a company’s financial position deteriorates. However, regular culling of staff can be popular with the workforce because poor performers cause much more damage than is apparent to management. The authors illustrate the severe impact of poor work using queuing theory and running a simulation.

Inventorising Information Technology Systems: Supporting the “Paradigm of Change”
by Mordechai Ben-Menachem and Garry S. Marliss, pp. 34–43. Information technology systems permeate organizations, supporting almost everything—apart from managing IT items themselves. Software configuration management systems aren’t designed to handle the totality of a distributed enterprise or to act as a management information resource. Software sources constitute major organization assets, but they can’t be treated as such until they’re inventoried. Software inventorying is one of a set of integrated technologies developed to address problems of managing constantly changing technological and business processes. Those technologies are aspects of an evolving business paradigm called the paradigm of change.

Network Effects and Social Dilemmas in Technology Industries
by Glenn J. Browne and Nirup M. Menon, pp. 44–50. The authors show that network effects can lead to a “social dilemma,” in which consumers’ actions result in serious negative consequences for the same consumers and for society as a whole in the long term.

Beyond the Black Box: Knowledge Overlaps in Software Outsourcing
by Amrit Tiwana, pp. 51–58. The well-known black-box model of software development outsourcing is typically effective. A field study of 209 global projects explored the approach’s limits as well as potential solutions to those limits. The study’s key finding is that the black-box approach usually works well in routine projects but fails in projects involving novelty. Such projects require careful deviations from the black-box model depending on novelty type. A knowledge congruence framework is offered to apply the findings to software development practice.

Finding Reusable UML Sequence Diagrams Automatically
by William N. Robinson and Han G. Woo, pp. 60–67. Software analysts create many artifacts, and until recently, these have been cumbersome to reuse. Reuser is a CASE (computer-aided software engineering) tool that lets analysts automatically retrieve related artifacts for reuse. Evaluations of Reuser suggest that its approach to UML artifact reuse is effective. This article reports on its effectiveness in assisting analysts to reuse UML sequence diagrams.

Provoking Creativity: Imagine What Your Requirements Could Be Like
by Neil Maiden, Suzanne Robertson, and Alexis Gizikis, pp. 68–75. Requirements engineering research, with its focus on elicitation, analysis, and management, offers little to support requirements creation or invention. This article describes innovative techniques to encourage creative thinking about requirements for an air traffic control system. It describes results from three creativity workshops and lessons learned about integrating such activities into structured requirements processes.

A Field Study of Developer Pairs: Productivity Impacts and Implications
by Allen Parrish, Randy Smith, David Hale, and Joanne Hale, pp. 76–79. Researchers have reported varied, even disparate, findings about the productivity of paired versus independent programmers. The authors conclude that the role-based coordination protocol associated with agile software methodologies overcomes a significant productivity loss that is otherwise associated with concurrent software development pairs.

Test-Driven Modeling for Model-Driven Development
by Yuefeng Zhang, pp. 80–86. A new software development process, called test-driven modeling, applies the XP test-driven paradigm in a model-driven development environment. TDM involves automatic testing through simulation and using executable models as living system architecture documents. Compared to traditional plan-driven methods, TDM can save a significant amount of time because it reuses message sequence charts and modeling diagrams.

Measuring Software Product Quality: A Survey of ISO/IEC 9126
by Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung, pp. 88–92. The international standard ISO/IEC 9126 defines a quality model for software products. Based on a user survey, this exploratory study empirically investigates whether the ISO/IEC 9126 categorization is correct and reliable in evaluating user satisfaction with judgment of the quality of a packaged software product. The results help clarify our understanding of product-quality attributes and provide guidance for revising the standard.