Adept: A Unified Assessment Method for Small Software Companies
by Fergal Mc Caffery, Philip S. Taylor, and Gerry Coleman, pp. 24–31. Software process assessments are typically the first step to commencing software process improvement. Small software companies find that many assessment methods are linked to plan-driven improvement models and can be expensive in terms of the resources required. The authors support these claims with statistics from the Irish software industry and present a lightweight assessment method called Adept. Adept unifies an existing plan-driven assessment method and an adapted, risk-based Agility/Discipline Assessment method. Adept has a low resource overhead and does not dictate either a plan-driven or agile process improvement model, making it an attractive assessment method for small software companies.

Value-Oriented Requirements Prioritization in a Small Development Organization
by Jim Azar, Randy K. Smith, and David Cordes, pp. 32–37. Requirements prioritization and selection play a critical role in project development. In small companies, this often-difficult process can affect not only project success but also overall company survivability. A value-oriented prioritization (VOP) framework can help this process by clarifying and quantifying selection and prioritization issues. A case study of a small development company shows a successful VOP deployment that improved communications and saved time by focusing project requirements decisions on core company values.

Introducing Version Control to Database-Centric Applications in a Small Enterprise
by Jan Ploski, Wilhelm Hasselbring, Jochen Rehwinkel, and Stefan Schwierz, pp. 38–44. Employing version control and separating test and production systems from each other are well-known software engineering practices. However, small software development organizations sometimes neglect the corresponding tools and processes, partly because they lack experience and partly because of genuine technical challenges. A new method of database-schema version control converts schema elements into textual configuration items that an organization can manage using freely available open source tools.

An Open Source Approach to Developing Software in a Small Organization
by Ken Martin and Bill Hoffman, pp. 46–53. The software development approach that developers at Kitware use borrows techniques from agile development and Extreme Programming and emphasizes long-term, ongoing projects. The company has used this approach on open source and closed-source projects over a wide range of sizes.

Achieving Quality in Open Source Software
by Mark Aberdour, pp. 58–64. The open source software community has published a substantial body of research on quality. The research reviewed for this article shows a consensus emerging on the key components of successful OSS delivery. This article will help both open source and closed-source software developers better understand how to achieve software quality.

A Quantitative Approach to Software Development Using IEEE 982.1
by Norman Schneidewind, pp. 65–72. Although software’s complexity and scope have increased tremendously over the past few decades, advances in software engineering techniques have been only moderate at best. Software measurement has remained primarily a labor-intensive effort and thus subject to human limitations. The Space Shuttle’s avionics software provides an excellent example of how to apply standards such as IEEE 982.1 and help alleviate human limitations. The standard establishes a quantitative roadmap to answer key software dependability questions.

Practitioner Tools and Workstyles for User-Interface Design
by Pedro Campos and Nuno Jardim Nunes, pp. 73–80. Literature that qualitatively studies UI-related software development work practices is relatively rare. Supporting workstyle transitions in UI practices is nonetheless important. A survey of 370 practitioners about workstyles and tool use offers concrete examples of design tools that support the most important workstyle transitions. The survey also led to recommendations for making design tools more human-centric and appealing to practitioners interested in the UI aspects of software development.