Variations in Software Development Practices
by Capers Jones, pp. 22–27. Thousands of companies develop software. In the past 35 years, over a million software applications have been put into use and more are being developed every day. But the methods and personnel used to develop software vary enormously. The application’s size and type are two key factors that determine what development methods will likely be used and what kinds of specialists will be employed. Building larger software projects is much more complex than building small programs, requiring many kinds of specialists and development activities.

Software Development Worldwide: The State of the Practice
by Michael Cusumano, Alan MacCormack, Chris F. Kemperer, and Bill Crandall, pp. 28–34. This research program focuses on collecting quantitative data about several countries’ current software practices and performance, their adoption of competing development models, and analysis of these practices’ impact on performance. The authors surveyed 104 projects in India, Japan, Europe, and the US.

How Software Engineers Use Documentation: The State of the Practice
by Timothy C. Lethbridge, Janice Singer, and Andrew Forward, pp. 35–39. Three large industrial studies explored how software engineers use and maintain documentation. The studies confirm the widely held belief that most software engineers don’t update most software documentation in a timely manner. The only notable exception is documentation types that are highly structured and easy to maintain, such as test cases and inline comments. The studies also show that out-of-date documentation, especially containing high-level abstractions, might remain useful.

Requirements Engineering: The State of the Practice
by Colin J. Neill and Phillip A. Laplante, pp. 40–45. Little contemporary data exists on the actual practices of software professionals for software requirements elicitation, requirements specification document development, and specification validation. The authors conducted an exploratory survey of several hundred software and systems practitioners. This article reports quantitative results for the purposes of further interpretation and comparison with other available data.

Software Reviews: The State of the Practice
by Marcus Ciolkowski, Oliver Laitenberger, and Stefan Bifr, pp. 46–51. The authors surveyed companies on industry adoption of software review technologies. They found that many companies take advantage of reviews for reasons such as early defect detection, quality control, and better team communication. However, many companies use reviews unsystematically, with mismatches between expected outcomes and review implementations.

Product Line Engineering: The State of the Practice
by Andreas Birk, Gerald Heller, Isabel John, Thomas von der Maßen, Klaus Müller, and Klaus Schmid, pp. 52–60. Software product lines are powerful tools for ensuring quality, economic efficiency, and manageability for software system families, but can be difficult to implement and maintain. Members from five organizations formed a work group to investigate and compare SPL practices. The results detail the SPL state of the practice.

Embedded Software Engineering: The State of the Practice
by Bas Graaf, Marco Lormans, and Hans Toetenel, pp. 61–69. Embedded software’s specific properties, such as hardware dependencies, make its development different from nonembedded software. So, you might expect that this domain would employ specific software development technologies. An inventory conducted at several European embedded-software-development companies shows that, remarkably, this isn’t true.

Is Internet-Speed Software Development Different?
by Richard Baskerville, Balasubramaniam Ramesh, Linda Levine, Jan Pries-Heje, and Sandra Slaughter, pp. 70–77. In a multiphase study conducted from 2000 to 2002, the authors examined the relationships between Internet-speed software development practices and agile and traditional development principles, and compared the two sets of principles. They found some overlap between traditional software development principles and agile principles but also some important differences.

Is the Software Engineering State of the Practice Getting Closer to the State of the Art?
by Donald J. Reifer, pp. 78–83. How do the state of the art and the state of the practice of software engineering differ? The author examines the divergence between the two. Having surveyed them using current frameworks, he next identifies missing research areas and good practices. Finally, eight critical success factors are identified that can help managers close the gap and take better advantage of the research that exists in their practice.

Signpost: Matching Program Behavior against Known Faults
by Mike Andrews, pp. 84–89. The causes of a computer program’s failure are often a complex mix of interactions between developer-written code and library code. To reduce debugging time and effort, the Signpost system uses a program’s behavior to query a knowledge base and automatically retrieve articles that describe known bugs and approaches to solving them.