Past, Present, and Future Trends in Software Patterns
by Frank Buschmann, Kerwin Henney, and Douglas C. Schmidt, pp. 31–37. This article discusses software patterns’ past, present, and future trends to help software developers and managers understand where the field has been, where it’s headed, and how learning more about patterns can help projects.

Using Patterns to Capture Architectural Decisions
by Neil B. Harrison, Paris Avgeriou, and Uwe Zdun, pp. 38–45. Researchers are investigating various methods and tools to help architects effectively document their decisions. However, such documentation remains difficult, so we often lose knowledge of key architectural decisions. Architecture patterns can be an important way to capture this knowledge.

Understanding the Power of Abstraction in Patterns
by Linda Rising, pp. 46–51. This unusual experience report about writing a set of patterns describes how Linda Rising and Mary Lynn Manns ended up on a journey outside the software domain that gave them a broader and deeper understanding of their patterns.

Organizing Security Patterns
by Munawar Hafiz, Paul Adamczyk, and Ralph E. Johnson, pp. 52–60. Every empire, after a period of rapid expansion, needs some time for consolidation or it risks disintegration. The expansion of software patterns has produced a large body of work that now needs organization. This article documents early efforts to consolidate and organize a subset of software patterns in the security domain. Lessons learned through this process can help people trying to organize patterns for other domains.

The Growing Divide in the Patterns World
Dragos Manolescu, Wojtek Kozaczynski, Ade Miller, and Jason Hogg, pp. 61–67. Microsoft’s patterns and practices group conducted a survey that indicates a significant gap between the patterns expert community and the software practitioners attempting to use and leverage patterns in their daily work. Drawing on the authors’ experience using patterns to package development guidance as well as input from practitioners using patterns, the article analyzes the key causes of this gap and recommends a set of actions aimed at pattern users in general and the patterns community in particular.

Intelligent Assistance in German Software Development: A Survey
by Jörg Rech, Eric Ras, and Björn Decker, pp. 72–79. A survey of 135 participants in Germany sheds light on the usage of and demand for intelligent assistance in software engineering activities. There’s high demand and acceptance for unobtrusive, quickly executable, and reactive assistance in core SE phases to help solve problems. In addition, the survey revealed several challenges for the future in SE work environments.

Interaction-Based Design for Mobile Collaborative-Learning Software
by María Ester Lagos, Rosa Alarcón, Miguel Nussbaun, and Francisca Capponi, pp. 80–89. The main difficulties in designing collaborative-learning applications are understanding which interactions between learners and teachers are the appropriate ones and then building software that takes such dynamics into account. An approach based on an interaction model lets designers and educational experts specify the interactions in a learning activity. The resulting flexible model is the basis for a flexible architecture that lets you create, extend, reuse, and compose diverse educational software products.

The Case for Frame-Based Engineering
by Paul G. Bassett, pp. 90–99. Frame technology can synthesize any information structure (such as a program) from machine-adaptable frames. The author contrasts FT with object-oriented classes. By canonically defining each information structure in terms of its unique properties, frames avoid the complexities induced by code-level redundancies. Frames also solve the problem of how to regenerate domain-specific-language programs without destroying prior customizations.