ADAPTING AGILITY

**Project Management in Plan-Based and Agile Companies**
by Martina Ceschi, Alberto Sillitti, Giancarlo Succi, and Stefano De Panfilis, pp. 21–27. The authors conducted an empirical study to assess, from a project management viewpoint, analogies and differences caused by adopting agile methods. Twenty managers of plan-based (traditional) and agile companies participated, and results indicate that adopting agile methods can improve management of the development process and customer relationships.

**Context-Adaptive Agility: Managing Complexity and Uncertainty**
by Todd Little, pp. 28–35. Landmark Graphics devised a way to categorize its software projects according to their complexity and uncertainty. The categories—dogs (simple projects with low uncertainty), colts (simple projects with high uncertainty), cows (complex projects with low uncertainty), and bulls (complex projects with high uncertainty)—help the team members know which development processes are necessary. They now adapt each project by adding practices according to its profile.

**Primavera Gets Agile: A Successful Transition to Agile Development**
by Bob Schatz and Ibrahim Abdelshafi, pp. 36–42. Primavera Systems, a vendor for enterprise project management software, reports on adopting the Scrum agile development process. The changes made helped the company start delivering higher-quality software while improving the development team’s quality of life. Today, Primavera’s development team is a model for other companies looking to adopt agile processes.

**Combining Agile Methods with Stage-Gate Project Management**
by Daniel Karlström and Per Runeson, pp. 43–49. Agile methods emerged from small, contract-driven projects. This study describes lessons learned from integrating agile methods with traditional stage-gate project management models at three large, product-driven companies. The study results support the feasibility of combining these approaches and suggests some adjustments to ease the integration.

COST AND VALUE

**Practical Guidelines for Expert-Judgment-Based Software Effort Estimation**
by Magne Jørgensen, pp. 57–63. Simple process changes can lead to more realistic software development effort estimates. Seven easy-to-implement guidelines, based on industrial experience and empirical studies, apply the most recent findings for judgment-based estimation of software project effort. The new guidelines supplement existing methods.

**Beyond Cost: The Drivers of COTS Application Value**
by Mark Keil and AmirTiwana, pp. 64–69. COTS software’s growing popularity has dramatically changed how companies acquire enterprise applications, yet surprisingly little is known about which attributes of enterprise COTS software buyers value most. A survey of 126 management information systems managers evaluated key COTS software attributes. The results helped create an enterprise COTS software analyzer that should benefit both buyers and suppliers.

FEATURES

**An Empirical Study of Programming Language Trends**
by Yaofei Chen, Rose Dios, Ali Mili, Lan Wu, and Kefei Wang, pp. 72–79. Predicting software engineering trends is hard because of the range of factors involved and the complexity of their interactions. Having developed earlier a tentative structure and some possible solutions for this problem, the authors now reduce the problem scope and try to gain some depth by focusing on a compact set of trends: programming languages. They measure the evolution of 17 languages over several years to draw statistical conclusions on what drives a language’s evolution.

**Leveraging Global Resources: A Process Maturity Framework for Managing Distributed Development**
by Narayan Ramasubbu, Prasad Kompalli, and M.S. Krishnan, pp. 80–86. Distributed software development is pervasive today as companies vie to leverage global resources. However, popular quality and process frameworks don’t specifically address the key processes needed for managing distributed development. This practitioner-oriented, evolutionary process-maturity framework features 24 new key process areas for managing distributed development and continuously improving product management capabilities.

**Toward an Acceptable Definition of Service**
by Steve Jones, pp. 87–93. Once architects, developers, designers, and clients agree on a definition of service, the question becomes how to define the service. Commercial developers have many options, and the evolving Web Services standards are giving service definition a structure. But challenges remain. If a service can’t define its security, availability, integrity, and environment, we’re no nearer an acceptable definition of service than we were 10 years ago. Without a true service-level agreement, can we measure quality?

**Mining Text for Expert Witnesses**
by Christopher Dozier and Peter Jackson, pp. 94–100. Text mining creates novel information resources from electronic text repositories. The authors created an expert-witness database based on text from legal, medical, and news documents that demonstrates the successful application of text-mining techniques.