**GLOBAL SOFTWARE DEVELOPMENT**

**A Practical Management and Engineering Approach to Offshore Collaboration**
by James Cusick and Alpana Prasad, pp. 20–29. Leveraging offshore staff for software development is rapidly becoming the norm in the Corporate Legal Services Division at Wolters Kluwer. Its Global Software Development model has evolved over the years into a relatively mature approach supporting new development as well as ongoing production. This model offers a tactical approach to the offshore process for companies pursuing or planning to expand into offshore development. This article also documents key practices that lead to success in this environment and details traps that can limit project effectiveness.

**Managing Knowledge in Global Software Development Efforts: Issues and Practices**
by Kevin C. Desouza, Yukika Awazu, and Peter Baloh, pp. 30–37. Global software development’s knowledge-intensive nature poses interesting challenges. Organizations must manage knowledge in all software development stages—from encapsulating design requirements to creating and testing programs, installation, and maintenance. Knowledge even extends to improving organizational software development processes and practices. Examining the recent knowledge management practices of more than 50 software organizations illuminates software development efforts on a global scale.

**Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing**
by Jyoti M. Bhat, Mayank Gupta, and Santhosh N. Murthy, pp. 38–44. Requirements engineering practices have changed significantly to address the needs of global software development characterized by teams working across geographies, time zones, and organizational boundaries. By sharing real-life case studies based on experience gained in working with an Indian IT services outsourcing firm, the authors provide insights into the root causes of RE phase conflicts in client-vendor offshore-outsourcing relationships.

**Distribution Dimensions in Software Development Projects: A Taxonomy**
by Dorina C. Gumm, pp. 45–51. Distribution in software development projects is a complex phenomenon with many dimensions. In addition to the obvious physical dimension entailed by geographically dispersed project personnel and resources, projects must contend with organizational distribution among various company departments, for example, as well as temporal and stakeholder distributions. A review of the literature on globally distributed software development projects produced a preliminary taxonomy based on a dimensional analysis of distribution. Developers can use this taxonomy to identify and visualize their project’s distribution dimensions and scopes. A real-world case study of the taxonomy’s use illuminates its value, as well as common challenges and possible solutions to distribution problems.

**Enabling Collaboration in Distributed Requirements Management**
by Vibha Sinha, Bikram Sengupta, and Satish Chandra, pp. 52–61. Requirements management, one of the most collaboration-intensive activities in software development, is particularly difficult when stakeholders are distributed, as in today’s global projects. EGRET is a collaborative requirements management tool for distributed software development teams. A preliminary usability study indicates the tool’s potential.

**How Much Software Quality Investment Is Enough: A Value-Based Approach**
by LiGuo Huang and Barry Boehm, pp. 88–95. A classic problem facing many software projects is determining when to stop testing and release the product for use. Risk analysis helps address this issue by balancing the risk exposures of doing too little or too much. A quantitative analysis approach based on the COCOMO II cost-estimation model and the COQUALMO quality-estimation model helps answer the question, “How much software quality investment is enough?” The authors use the models and some representative empirical data to assess the relative pay-off of value-based testing as compared to value-neutral testing. They include examples of the approach’s use under differing value profiles.

**Improving Software Development through Three Stages**
by Tetsuro Motoyama, pp. 81–87. In three stages covering 12 years and 16 projects, a software development group worked on attaining visible schedule control and usable software design documentation. The first stage identified two major problems: unstable code and schedule delays. The group used code inspection to address the immediate issue of code construction. In the second stage, the group formulated and implemented a software development process that addressed the entire software development life cycle. The third stage improved that process. The major factor in process improvement was the group’s willingness to identify its mistakes and problems.