**FOCUS: EXTREME PROGRAMMING**

**Introducing XP into Greenfield Projects: Lessons Learned**
by Jonathan Rasmussen, pp. 21–28. ThoughtWorks introduced XP into an organization and successfully completed a bleeding-edge technology project with client staff that had no previous experience using an agile development approach. This article illustrates not only how XP helped make the project a success but also gives other lessons learned regarding the introduction of XP at client sites.

**Exploring XP for Scientific Research**
by William A. Wood and William L. Kleb, pp. 30–36. Can we successfully apply XP in a scientific research context? A pilot project at the NASA Langley Research Center tested XP's applicability in this context. Because the cultural environment at a government research center differs from the customer-centric business view, eight of XP’s 12 practices seemed incompatible with the existing research culture. Despite initial awkwardness, the authors determined that XP can function in situations for which it appears to be ill suited.

**Assessing XP at a European Internet Company**
by Orlando Murru, Roberto Deias, and Giampiero Mugheddu, pp. 37–43. First, a European Internet services company, has been experimenting with introducing XP in its development work. The article describes the company’s experiences with XP, explores its implementation practice by practice, and discusses XP’s pros and cons in three key areas: customer relationships, project management, and ISO 9001 quality assurance.

**FOCUS: QUALITY ASSURANCE**

**Statistical Process Control to Improve Coding and Code Review**
by Alice Leslie Jacob and S.K. Pillai, pp. 50–55. Software process comprises activities such as estimation, planning, requirements analysis, design, coding, reviews, and testing, undertaken when creating a software product. Effective software process management involves proactively managing each of these activities. Statistical process control tools enable proactive software process management. One such tool, the control chart, can be used for managing, controlling, and improving the code review process.

**Best Practices in Code Inspection for Safety-Critical Software**
by Jorge Rady de Almeida Jr., João Batista Camargo Jr., Bruno ABRantes Bassetto, and Sérgio Miranda Paz, pp. 56–63. One of the most important issues in analyzing safety-critical systems is code verification through an inspection checklist, whose items must be applied to the source code. The attention given to this list will help ensure obedience to good coding rules and represents an important factor in the design of safety-critical systems. This list also helps in the development of non-safety-critical systems, improving their quality and correctness.

**Long-Term Testing in a Short-Term World**
by Vaughn T. Rokosz, pp. 64–67. Accelerated stress testing reduces the time required to test a system but can be hard to apply to functions running on a fixed schedule. This article describes how to accelerate the testing of scheduled functions by triggering them through automated tests, either by periodically advancing the system clock or through a programmable event interface. With this method, the accelerated stress tests don’t distort the system’s operational profile.

**Measurement Modeling Technology**
by Jim Lawler and Barbara Kitchenham, pp. 65–75. Although rigorous measurement has become a necessity in the software industry, many measurement programs fail to deliver real benefit to software managers. The required data is often missing, invalid, or late. But a properly automated measurement system can deliver timely reports that enable proactive management. The measurement modeling technology introduced here supports such automation. It also establishes standard measurements and metrics that organizations can share and combine across projects, departments, and companies.

**Automated Quality Assurance for Document Understanding Systems**
by Sherif Yacoub, pp. 76–82. To process high-volume input data—such as the scanned images of publishers’ book and journal collections—content understanding systems should run automatically, continuously, and without human attendance. Ensuring the output quality of such systems is a challenging task, however, and automated quality assurance techniques are thus essential to its success. In this article, the author discusses three automated QA techniques that were developed for Hewlett-Packard’s Digital Content ReMastering system.

**FEATURES**

**How the Work of Software Professionals Changes Everything**
by James Cusick, pp. 92–97. Software professionals’ work remains vital to modern industrial society as key definitions of work, the organization of work, and the relationship of modern software systems to society’s evolution demonstrate. This article discusses why the question of what work to pursue is interesting with regard to the software field; it also explores the meaning of work itself and its relation to the individual, society, and the economy.

**Generating Text Search Applications for Databases**
by Omar Alonso, pp. 98–105. To develop reusable domain-based assets, the author shows how to connect a domain analysis tool’s output to a program generator and then to a domain implementation’s input. The generator he built using Java, XML, and Oracle tools can produce code for JSP, PL/SQL, and PSP. This domain engineering research project demonstrated that it is possible to successfully implement software product lines in industrial environments using existing tools.