The virtual reality field is transitioning into work influenced by video games and thus now influences that industry as well. Because much of the research and development being conducted in the games community parallels the VR community’s efforts, it has the potential to affect a greater audience.

Given these trends, VR researchers who want their work to remain relevant must realign to focus on game research and development. Research in the games arena affects not just the entertainment industry but also the government and corporate organizations that could benefit from the training, simulation, and education opportunities that serious games provide.

Iterative Rework: The Good, the Bad, and the Ugly
pp. 34-41
Richard E. Fairley and Mary Jane Willsire

In iterative development, each iteration subsumes the previous iteration’s software and adds capabilities to the evolving product to produce a next version. Each iteration involves a certain amount of rework to enhance and fix existing capabilities (the good). However, excessive rework could indicate problems in the requirements, the developers’ skills and motivation, the development processes or technology used, or all of the above (the bad). Exorbitant levels of rework result in truly untenable situations (the ugly).

On the other hand, too little rework could indicate insufficient review and testing or too little anticipation of the product features needed to support the next version (bad that can turn ugly). Understanding and correcting the root causes of problems that result from too much or too little rework can significantly increase productivity, quality, developer morale, and customer satisfaction.

Test-Driven Development: Concepts, Taxonomy, and Future Direction
pp. 43-50
David Janzen and Hossein Saeidian

The test-driven development strategy requires writing automated tests prior to developing functional code in small, rapid iterations. XP is an agile method that develops object-oriented software in very short iterations with little upfront design. Although not originally given this name, TDD was described as an integral XP practice necessary for analysis, design, and testing that also enables design through refactoring, collective ownership, continuous integration, and programmer courage.

Along with pair programming and refactoring, TDD has received considerable individual attention since XP’s introduction. Developers have created tools specifically to support TDD across a range of languages and have written numerous books explaining how to apply TDD concepts. Researchers have begun to examine TDD’s effects on defect reduction and quality improvements in academic and professional practitioner environments, and educators have started to examine how to integrate TDD into computer science and software engineering pedagogy.

Transforming Software Development: An MDA Road Map
pp. 52-58
Thomas O. Meservey and Kurt D. Fenstemacher

While cartographers create different kinds of maps in varying detail to model different geographical aspects, software developers have traditionally focused on a single model: code, which is the map equivalent of a photograph, displayed actual size. Models are commonly used to flexibly represent complex systems and can be viewed at many levels of abstraction. Complementary model views can be combined to give a more intelligible, accurate view of a system than a single model alone.

Software development experts have long advocated using models to understand the problems that a system seeks to address, yet development teams commonly employ models only in the early stages of modeling. Often, once construction begins, the teams leave the model behind and never update it to reflect their changing conceptions of the project.

In 2001, the Object Management Group launched the Model Driven Architecture initiative, with the ambitious goal of shifting the focus of software development from writing code to modeling.

Distributed Access Management in Multimedia IDCs
pp. 60-69
Rafae Bhatti, Basil Shafiq, Mohamed Shehab, and Arif Ghafoor

Businesses can use the Web-based e-enterprise model not only to offer their services to a diverse and distributed clientele from a single online location but also to simplify the administration of such services. An underlying Internet data center architecture supports the delivery of the massive amounts of multimedia data from a single virtual location to a huge clientele.

While using IDCs lets the original source provide services to its subscribers through a third-party infrastructure, this environment’s dynamic nature raises serious concerns regarding the management of access-control policies across heterogeneous enterprise domains. The authors describe a prototype architecture that integrates access-management mechanisms into the design of multimedia IDCs for secure dissemination of information content in a distributed collaborative environment. The architecture also provides a service-driven, context-aware policy-design methodology that fits the host IDC’s organizational needs.