Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques
by Christof Ebert, pp. 19–25. Requirements are the initial and basic building blocks combining processes in a product’s life cycle. A field study of 246 industry projects in the domains of software platforms, embedded systems, and software applications reveals that four life-cycle techniques must be used simultaneously to achieve tangible performance improvement measured by schedule adherence.

The Usage Model: Describing Product Usage during Design and Development
by Erik Simmons, pp. 34–41. For industrial and interaction designers to be effective, they require a rich description of product usage and the resulting requirements. Intel designers have used the term usage model to describe product use in a stated context, but agreeing on the usage model’s structure and content has proved challenging. A new usage model structure has three tiers: supporting data, overview, and usage details. Teams can use the description during product planning, design, development, and validation.

Schedule Estimation and the Uncertainty Surrounding the Cone of Uncertainty
by Todd Little, pp. 48–54. One software product company found that its project duration estimation accuracy followed a lognormal distribution, and the uncertainty range was nearly identical at all stages in the project life cycle. This conflicts with popular interpretation of the “cone of uncertainty” concept.

Reflections on Software Engineering Education
by Hans van Vliet, pp. 55–61. Although the SWEBOK and Software Engineering 2004 initiatives have increased SE education’s visibility, they focus largely on SE’s aspects at the expense of its human and social dimensions. SE educators can fall into traps that result from a pure engineering approach, and then in turn mislead students as to SE’s true nature.

Certifying Software Component Attributes
by Jørgen Bøegh, pp. 74–81. System integrators considering software components depend on component suppliers to adequately describe their components. A component description must be trustworthy and contain sufficient detail for integrators to understand its real abilities. The European Clear and Reliable Information for Integration project has developed a flexible, property-value approach that lets you similarly manage simple and complex properties.

What Do We Know about Defect Detection Methods?
by Per Runeson, Anneliese Andrews, Carina Andersson, Tomas Berling, and Thomas Thelin, pp. 82–90. Using 12 empirical studies, including 10 experiments and two case studies, the authors compare inspection and testing techniques and then derive practical implications. Although the research results were somewhat inconclusive, the authors recommend that practitioners generally use inspections for requirements and design and testing for code. Also, because different defect detection methods find different types of defects, the methods might be complementary. Finally, they list factors that help frame the question and guide practitioners in integrating and evaluating the detection methods in their environments.