**COTS INTEGRATION**

An Active-Architecture Approach to COTS Integration

by Brian Warboys, Bob Snowdon, R. Mark Greenwood, Wykeen Seet, Ian Robertson, Ron Morrison, Graham Kirby, Dharini Balasubramaniam, and Katherine Mickan, pp. 20–27. COTS products’ developers and source code are rarely available, and the products’ ongoing development is unpredictable. The ArchWare framework approach addresses these problems by recognizing COTS products as part of an information system’s ambient environment. An active-architecture model that changes as the system evolves captures the composition and integration of COTS components.

Coordinating COTS Applications via a Business Event Layer

by Wilfried Lemahieu, Monique Snoeck, Frank Goethals, Manu De Backer, Raf Haesen, Guido Dedene, and Jacques Vandebulcke, pp. 28–35. The BECO approach to COTS integration uses existing, one-to-one communication technologies for event notification and exploits business events as higher-level, many-to-many units of coordination. Business processes can then be implemented as sequences of business events. This consistent, flexible integration approach can be layered on top of existing technologies.

Performance Techniques for COTS Systems

by Erik Putrycz, Murray Woodside, and Xiuping Wu, pp. 36–44. The increased adoption of COTS-based systems creates challenges for managing performance because the source code of the components involved isn’t available. However, the availability of the components before the system is built provides early data on their performance properties. If you combine modeling based on this data with new tracing techniques, you can better predict performance, plan deployment, and diagnose problems.

Evaluating COTS Component Dependability in Context

by Paolo Donzelli, Marvin Zelkowitz, Victor Basili, Dan Allard, and Kenneth N. Meyer, pp. 46–53. Selecting the right component to integrate into an application is challenging and risky, particularly without empirical evidence that a component will be dependable in a particular context. This empirical COTS evaluation method reduces selection risks by focusing on a specific context. It also supports integration by highlighting the component weaknesses that must be addressed to achieve success.

Value-Based Processes for COTS-Based Applications

by Ye Yang, Jesal Bhuta, Daniel N. Port, and Barry Boehm, pp. 54–62. Traditional software process models assume that software systems are developed largely by writing code from scratch, and thus fail to address many of the challenges associated with developing COTS-based applications. This often leads to selecting best-of-breed but incompatible COTS products, without considering the increased costs and reduced benefits incurred by trying to glue these together. A value-based set of processes can help steer such projects toward avoiding or minimizing such value losses.

Improving Software Development Management through Software Project Telemetry

by Philip M. Johnson, Hongbing Kou, Michael Paulding, Qin Zhang, Aaron Kagawa, and Takuya Yamashita, pp. 76–85. Software project telemetry, a new project management approach, is a style of software metrics definition, collection, and analysis that uses sensors to unobtrusively monitor development. Project Hackystat, which provides an open source reference framework for telemetry, observes behavior in client systems, gathers raw data, and abstracts the data into numerical telemetry values that support decision making.

Structural Shifts in the Chinese Software Industry

by Nir Kshetri, pp. 86–93. Although China has left India far behind in terms of most economic and technological indicators, China has failed to catch up in one area: the commercial software industry. While India’s development has been one-dimensional and has focused on the skill base, the Chinese software industry in recent years has been undergoing major structural shifts with respect to markets, participants, technology, focus, and products.

**FEATURES**

Rapid Embedded System Testing Using Verification Patterns

by Wei-Tek Tsai, Feng Zhu, Lian Yu, and Ray Paul, pp. 68–75. A verification pattern approach for rapidly testing real-time embedded systems first classifies system scenarios into patterns. For each scenario pattern, the VP approach then develops a test script template for all the scenarios belonging to that pattern. In this way, instead of developing numerous scripts to test the system, test engineers can customize and reuse a set of script templates to test the entire application, saving significant effort and time.