The effective use and widespread adoption of Web services technologies require new frameworks and systems for the conceptual modeling, analysis, management, simulation, and testing of service models and abstractions. These frameworks can provide a basis for matching service specifications or checking compatibility and replaceability among services as well as conformance with standards based on high-level models.

The authors propose a conceptual framework that provides a context for analyzing existing Web services technologies to better understand their goals, benefits, and limitations. Using this framework to view the different approaches to interoperability can help identify the tool support needed to leverage these technologies. According to the authors, their Web services survey work thus differs from the fragmented, topic-specific, or nonholistic efforts others have pursued.

Can We Make Operating Systems Reliable and Secure?

A nasty little secret those in the computer industry do not like to discuss is that TV sets, DVD recorders, MP3 players, cell phones, and other software-laden electronic devices are more reliable and secure than computers. Yet what consumers expect from a computer is what they expect from a TV set: to buy it, plug it in, and have it work perfectly for the next 10 years. IT professionals must take up this challenge and make computers as reliable and secure as TV sets.

The worst offender when it comes to reliability and security, according to the authors, is the operating system. Although application programs contain many flaws, if the operating system were bug free, bugs in application programs could do only limited damage, so the authors focus here on operating systems.

Achieving Scalability in Real-Time Systems

Most embedded computing systems are powered by batteries. Energy consumption is a critical aspect of these systems, directly affecting both performance and lifetime. Unfortunately, high performance and a long battery life create conflicting objectives: To achieve high performance, the system must operate at high speed, which consumes more power, whereas achieving a long battery life requires the system to consume less energy, which means operating at lower power.

In the presence of timing and resource constraints, a real-time system’s performance does not always improve as processor speed increases. Similarly, when reducing processor speed, the delivered service’s quality might not always degrade as expected.

Although many researchers have looked at reducing the energy consumption of real-time applications, the problem of scaling performance with speed variations requires further exploration. A proposed computational model varies task response times continuously with processor speed, enabling the system to predictably scale its performance during voltage changes.

Fault Evaluation for Security-Critical Communications Devices

International standards, such as the Common Criteria for Information Technology Security Evaluation, offer frameworks and bookkeeping tools for performing security evaluations, but such standards provide little specific technical guidance, especially for high-grade evaluations. Traditional techniques for analyzing electronic circuitry also fall short of addressing specific security evaluation needs.

From a security perspective, the focus is on checking for silent failures, which compromise the device’s security function in some way not obvious to its operator. To that end, the authors have evolved a rigorous way to structure the search for information security fault modes that combines techniques from information flow and risk analysis. In a case study of a prototype cryptographic device, they found that security evaluations were more thorough and that it was easier to organize a sound overall case for evidence that supports or refutes the device’s alleged security level.

Web Services Interoperability Specifications

Deterministic ends should be accomplished with deterministic means. Nondeterminism should be judiciously and carefully introduced where needed, and it should be explicit in programs. This principle seems obvious, yet threads do not accomplish it. They must be relegated to the engine room of computing, to be suffered only by expert technology providers.