Standards & Emerging Technologies

Editor: Sumi Helal ■ University of Florida ■ helal@cise.ufl.edu

SODA: Service-Oriented Device Architecture

Scott de Deugd, Randy Carroll, Kevin E. Kelly, Bill Millett, and Jeffrey Ricker

Monitoring and controlling a physical environment has long been possible through device interfaces ranging from basic sensors and actuators to complex digital equipment and controllers. Such devices and the systems they enable have traditionally been the domain of embedded systems developers. We now see an increasing need and opportunity to create interfaces between the physical world of sensors and actuators and the software world of enterprise systems.

Wholesalers, retailers, and distributors demand immediate monitoring and control of shipments, enabled by RFID sensor data piped directly into their manufacturing, billing, and distribution software. Home healthcare monitoring can be implemented by providing devices such as EKG monitors and glucose monitors and pulse oximeters that can continuously monitor ambulatory patient status and alert healthcare providers of conditions requiring immediate care. A military control center must combine sensor data from various logistics and tactical environments—including the monitoring and control of RFID readers, vehicle control buses, GPS tracking systems, cargo climate controllers, and specialized devices—to provide situational awareness, preventive fleet maintenance, and real-time logistics.

The types of devices available for such scenarios continues to grow, while the cost of deploying them in the physical world and connecting them to all manner of networks continues to drop. However, the device interfaces, connections, and protocols are multiplying at a corresponding rate, and enterprise system developers are finding that integrating devices into the information technology (IT) world is daunting and expensive.

We now see an increasing need and opportunity to create interfaces between the physical world of sensors and actuators and the software world of enterprise systems.

To eliminate much of the complexity and cost associated with integrating devices into highly distributed enterprise systems, we propose leveraging existing and emerging standards from both the embedded-device and IT domains within a Service-Oriented Device Architecture (SODA).

MODELING DEVICES AS SERVICES

SODA is an adaptation of a service-oriented architecture (SOA),[1] which integrates business systems through a set of services that can be reused and combined to address changing business priorities. Services are software components with well-defined interfaces, and they are independent of the programming language and the computing platforms on which they run.[2,3] The SODA approach to designing and building distributed software is to integrate a wide range of physical devices into distributed IT enterprise systems. At the simplest level, as figure 1 shows, SODA lets programmers deal with sensors and actuators just as business services are used in today's enterprise SOAs.

SODA focuses on the boundary layer between the physical and digital realms. In other words, a sensor, such as a digital thermometer or an RFID tag reader, can translate a physical phenomenon into corresponding digital data. Or, an actuator, such as a heater relay or alarm beacon, can translate a digital signal into a physical phenomenon. Sensors and actuators combine either physically or conceptually to create complex devices and services—such as a climate-control module or a geo-fencing system that monitors a vehicle's position within a restricted area.

Viewing SODA in this way places few bounds on the device type or its usage, thus encompassing many types of devices from government, industry, and scientific enterprise. The devices range from basic sensor interfaces to complex diagnostic equipment. Whether providing the location of critical cargo, the blood-sugar level of a loved one, or the critical status of an energy distribution system, what devices share in common is that their data, functions, and events are critical services to the enterprise in which they're used.[4]

We wouldn't need to think of devices in the broader terms of a digital realm without the types of complex distributed

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/06/$20.00 © 2006 IEEE

systems enabled by ubiquitous networks such as Bluetooth and the Internet. Without such networks, a signal would exist inside a single isolated machine or a well-defined isolated system. Networks enable a single processor to access signals from any number of devices. Thus, the challenge of wide-scale device integration is predicated on the existence of a universal network capable of supporting complex distributed systems.

Before the networking capabilities enabled by the Internet, two or more devices would be associated only within a single well-defined system. The engineers would define the devices' interfaces and interdependencies in the system design. With the Internet, enterprise systems can now access signals from numerous devices on an ad hoc basis. The ability to access and control aspects of the physical realm, which is critical to an enterprise, opens new opportunities and advantages but can be self limiting. The protocols, connections, and interfaces to devices are extremely diverse, and programming to them is often unfamiliar territory to system and Internet developers. Ancillary device interface software often is as critical to the completion of the IT project, with limited reuse and relatively high maintenance and support costs.

HOW A SERVICE-ORIENTED ARCHITECTURE CAN HELP

SODA aims to

- provide higher-level abstractions of the physical realm,
- insulate enterprise system developers from the ever-expanding number of standard device interfaces, and
- bridge the physical and digital realms with a known service or set of services.

SODA implementations can use existing and emerging device standards and SOA standards. SODA should be straightforward to implement, unlocking the potential to enable a new level of Internet-based enterprise systems.

Consider, for example, an enterprise scenario to process shipments of hazardous biological material. The application must track the materials during manufacturing and shipping, monitor and control their environment during shipment, and verify their delivery. Enterprise-system developers must build numerous complex distributed device interfaces for such an application:

- RFID readers and various digital I/O installed in the factory, delivery vehicles, shipping container, and at the receiving site;
- GPS devices on the shipping container and transport vehicle;
- a climate-control system in the shipping container;
- an interface to a vehicle control bus to determine the transport vehicle's status;
- various wired and wireless networks and protocols connecting all these devices; and
- numerous versions of all these items, depending on the specific device and on the protocol and networking vendor contracted by the system integrator.

In contrast, consider the enterprise system application developer's job if he or she could design an application to register interest in, and control, device services that can

- confirm, using an RFID-tagged item, that a shipment has cleared quality control, safety inspection, and the loading dock;
- confirm that the tagged shipment is secured in the shipping container in its assigned vehicle;
- provide a named vehicle and container's geographic location;
- provide notification of vehicle breakdown or of cargo leaving its geofence boundary;
- provide cargo's environmental status and listen for climate-control adjustments;
- provide status of vehicle safety and mechanical systems; and
- confirm the environmental transit log and compare a shipping notice to a tagged shipment secured at a delivery location.

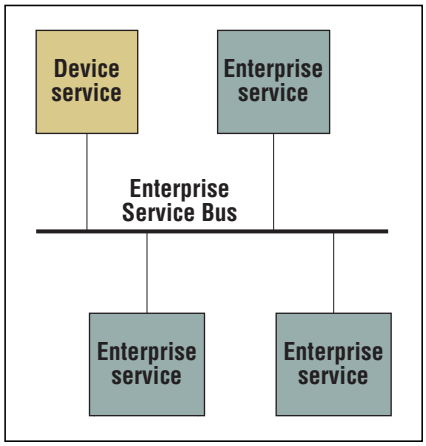

Figure 1. When modeled as services, device access and control can be made available to a wide range of enterprise application software using service-oriented architecture mechanisms.

A device integration developer would be responsible for encapsulating devices as services, dealing with the device-specific connections and protocol as well as with network interfaces needed to publish the data over a defined SOA protocol. A standard specified device service can have a wide variety of underlying hardware, firmware, software, and networking implementations that don't affect the consumer of the service.

The overall system design would specify the required service interfaces. Suppliers would be responsible for the device adapters and service logic required to provide the specified service for their devices. Other developers could build more complex or composite services from lower-level device services. The system integrator could bid out components from multiple suppliers and avoid maintaining multiple versions of device-specific interfaces in the application code. Enterprise developers could code to a common or even standard set of services. They could not only build this application with clean

STANDARDS & EMERGING TECHNOLOGIES

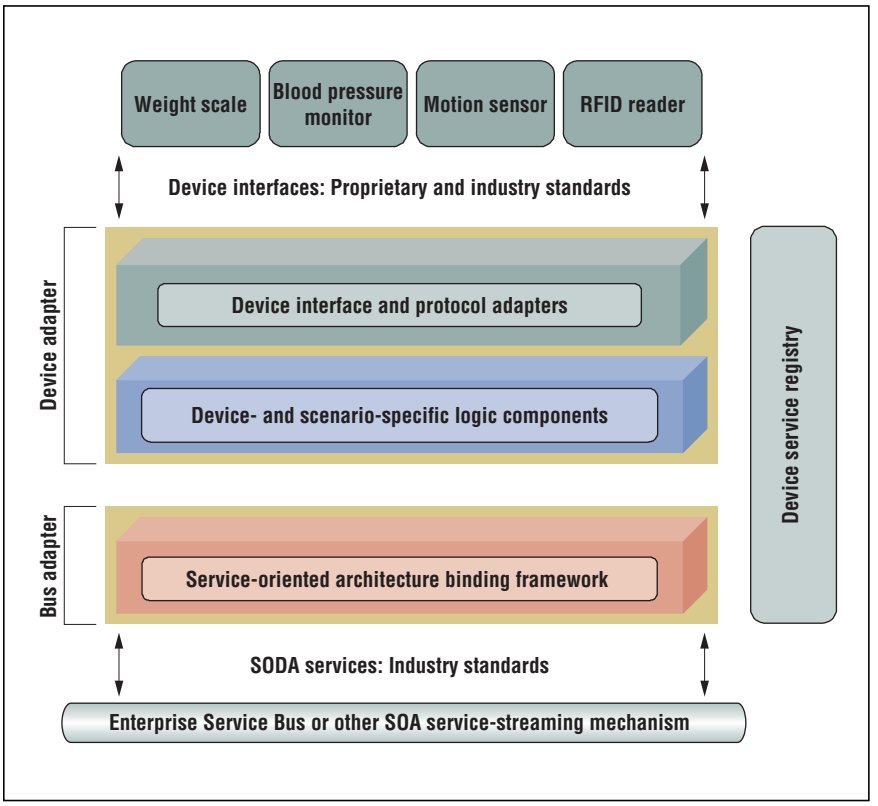

Figure 2. A Service-Oriented Device Architecture implementation provides an abstract services model of a device by talking to proprietary and standard device interfaces on the one side and, on the other, presenting device data as SOA services over a network through a bus adapter.

device interfaces but also build and integrate future applications and system enhancements reusing these same device services. The enterprise could upgrade device hardware, firmware, and even the lower-level device interfaces with little or no impact on the consuming applications.

THE ARCHITECTURE

SODA should be consistent with an enterprise SOA. An SOA provides the ability to map business processes and events across an open communication infrastructure, sharing data and functionality in an open, flexible manner.[5] We can apply SOAs' modular, componentized nature not only to traditional enterprise services but also to events and services from all data sources and devices, including embedded sensors and actuators and complex equipment.

Conventional approaches to device integration often center on custom interface software communicating to enterprise applications through a variety of IT middleware and API technologies. This approach has served enterprises well, but SOA, standards, and open software initiatives are moving beyond this middleware architecture. Although IT applications are being adapted to an SOA, standards for defining the low-level device interfaces are still emerging. However, technology exists today to leverage an SOA across the entire spectrum of critical events and data originating from devices.

Mechanisms for building and sharing service interfaces, capabilities for remote software maintenance, and loosely coupled messaging models[6] present highly effective technologies for SODA's implementation. SODA requirements include

- using a device adapter model to encapsulate device-specific programming interfaces;
- employing loosely coupled messaging on the services side, capable of supporting multiple streaming services commonly used in SOA enterprise systems, such as an Enterprise Service Bus;
- using open standards, where available, at the device- and services-interface level;
- providing a means to present standard or open service interfaces to devices that have proprietary protocols or where it might not be practical to drive standards into the low-level device interface;
- supporting the implementation of a spectrum of device adapters—from simple, low-cost sensor data to complex device protocols;
- supporting loading of remotely configurable logic components in device adapters for maintenance, upgrade, and extended functionality; and
- adapting security mechanisms as required for the domain.

A SODA implementation comprises three main components (see figure 2). *Device adapters* talk to device interfaces, protocols, and connections on one side and present an abstract services model of the device on the other. The *bus adapter* moves device data over network protocols by mapping a device service's abstract model to the specific SOA binding mechanism used by the enterprise. The *device service registry* provides for the discovery and access of SODA services.

Where device interface standards don't exist, device interface and protocol adapters within SODA implementations provide a common model of devices to the software used to create service interfaces. Widespread adoption

of standards at the device-interface level will reduce the development and maintenance costs of device adapters and their corresponding SODA services. However, standards at the services layer can provide the largest leverage for both the device and enterprise markets.

Rapid standardization of device services, device-services transport mechanisms, and tooling will let device manufacturers develop their interfaces and provide SODA services to the enterprise, shifting development responsibility for device adapters and services to the appropriate point in the supply chain rather than forcing enterprise developers to deal with thousands of APIs. For this adoption to take place, the SODA model must evolve with open and accessible standards, which must cover the specific services used within and across enterprises. Reducing the barriers to acceptance requires that the standards be open and part of a community and that samples, examples, frameworks, and tooling be made available through reference implementations. We can develop an open community of professionals by having government, corporate, and academic entities jointly sponsor such standards and activities that result in prototypes, pilots, and deployed solutions.

REFERENCES

1. T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, 2005.
2. D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*, Prentice Hall, 2005.
3. T. Erl, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall, 2004.
4. H. Havenstein, "SOA App Quickly Boosts Storm Response," *Computer-World*, June 2006; www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=112207.
5. C. Koch, "How SOA Really Works," *CIO*, Aug. 2005; www.cio.com/blog_view.html?CID=10591.
6. A. Stanford-Clark, "Coupled or Decoupled Plus Heavyweight and Lightweight Delivery Considerations in an Enterprise Service Bus Context," *Middlewarespectra*, Aug. 2004, pp. 26–33.

Randy Carroll is a senior software engineer at IBM. Contact him at rwcarroll@us.ibm.com.

Scott de Deugd is a senior engineer with IBM's Sensors and Actuators Solutions group. Contact him at dedeugd@us.ibm.com.

Kevin E. Kelly is an IBM senior technical staff member. He's also a member of HL7 and serves as the HL7 W3C Liason, and is chair of the W3C Compound Document Formats Working Group. Contact him at kevin.kelly@us.ibm.com.

Bill Millett is a senior software architect in IBM's Sensors and Actuators Solutions group. Contact him at bmillett@ca.ibm.com.

Jeffrey Ricker is the founder and CEO of Distributed Instruments, a company dedicated to providing tools and servers that solve the challenges of sensor fusion. He founded XMLSolutions, the first company dedicated to XML technology. Contact him at ricker@distributedinstruments.com.

Continued from page 93

Resource management and configuration. During deployment, system management and configuration often becomes a major concern and a significant cost factor. Consequently, we need to investigate dynamic resource management and minimal (or zero) configuration.

Business cases and applications. As sensor network technologies mature, research on potential application areas and business cases becomes more important. But this area has seen little research so far. We need to assess real-world applications and business. In addition, research from areas other than computer science, such as marketing, business, and electrical engineering, can provide important input.

As the workshop demonstrated, applied pervasive technologies have great potential. Many technologies will soon mature and provide great opportunities for enhancing processes in industry areas beyond logistics. However, we need to resolve many technical and cooperation issues. Pervasive technologies seem to generate the most benefit when they're tailored to specific contexts. It's also clear that, regardless of technologies and application scenarios, user acceptance is essential; we need to take it into account from the beginning to ensure successful deployment.

ACKNOWLEDGMENT

We thank all the workshop authors and participants for the fruitful discussion and valuable input for this article. All papers are part of the *Proceedings of the Pervasive 2006 Workshop Pervasive Technology Applied—Real-World Experiences with RFID and Sensor Networks*; most are available at www.hcilab.org/events/pta2006/proceedings.htm.