Chapter 7

Architecture Standardization for WoT

7.1 Platform Middleware for WoT

Current markets of the Internet of Things (IoT) and Web of Things (WoT) are highly fragmented. Various vertical WoT/IoT solutions have been designed independently and separately for different applications, which inevitably impacts or even impedes large-scale WoT deployment. A unified, horizontal, standards-based platform is the key to consolidate the fragmentation.

We talked about communication middleware for IoT in Chapter 5. Communication middleware and platform middleware are closely related to and sometimes tightly integrated with each other. However, there are differences between them. We will talk about platform middleware (also called application frameworks, or sometimes, directly, the three-tiered application server) for IoT in this chapter, especially frameworks at the application level, or at the "M" level of the DCM (direct, change, manage) value chain. One main goal of platform middleware is to bring the IoT applications (including Intranet of Things and Extranet of Things) to the World Wide Web, so we will use the term *Web of Things* more in this chapter.

According to the WoT/IoT vision, everyday objects such as domestic appliances, actuators, and embedded systems of any kind in the near future will be connected with each other and with the Internet. These will form a distributed network with sensing capabilities that will allow unprecedented market opportunities, spurring new services, including energy monitoring and control of homes, buildings, industrial processes, and so forth. In this chapter, we concentrate on the actual implementation of the multitiered application-level technologies.

An interesting observation is that many software architectures and technologies have long before used the term *object* in many modeling methodologies such as the well-known object-oriented design, object-oriented software engineering and programming, CORBA (common object request broker architecture), DOM (document object model), POJO (plain old Java object), COM (component object model) and DCOM (distributed COM), OPC (object linking and embedding for process control), OID (object identification), SOAP (simple object access protocol), JSON (JavaScript object notation), and so on. The entire software industry is already an object-oriented world, as shown in Figure 7.1. The representation and programming of objects has a profound supporting base in the software world.

Now that IoT/WoT brings the real-world objects into the game, there must be many natural fits for mapping the IoT

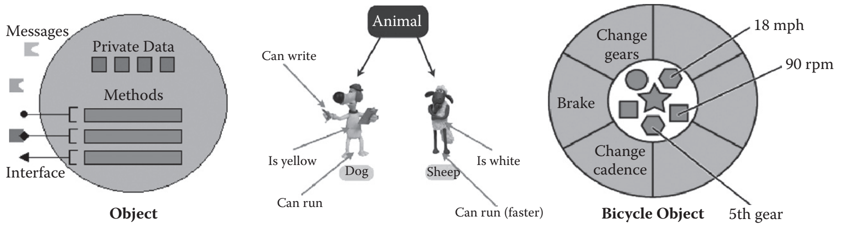

Figure 7.1 Object-oriented and real-world object programming.

objects to software objects. The transition from object-oriented to real-world objects programming is a natural one. In fact, the Mango open-source software platform for IoT and machine-to-machine (M2M) has found a natural fit between JSON and M2M applications and used the technology in its products.

7.1.1 Standards for M2M

The European Telecommunications Standards Institute (ETSI's) Global Standards Collaboration (GSC) M2M Standardization Task Force (MSTF) considers as M2M any automated data exchange between machines including virtual machines such as software applications without or with limited human intervention as described in the previous two chapters. The Technical Committee's overall objective is creating open standards for M2M communications to foster the creation of a future network of objects and services so that already-existing and rapidly growing M2M businesses based on vertical applications using a multitude of technical solutions and diverse standards can be turned into interoperable M2M services and networked businesses. An ETSI M2M architecture diagram (http://www.telit.com/img/images%20market%20intelligence/m2msystemarchitecture.jpg) shows the high-level approach to invert the pipes. Vertical proprietary applications shall be substituted by a horizontal architecture, wherein applications share common infrastructure, environments, and network elements. An M2M system described by clearly structured and specified network transitions, software and hardware interfaces, protocols, frameworks, and so forth shall ensure the interoperability of all system elements. The Technical Committee's work is based on the general guideline of using existing standardized systems and elements. It evaluates them according to M2M requirements, filling gaps as necessary by either enhancing existing standards or producing supplemental ones.

The key elements of the ETSI M2M architecture are described below:

- M2M device: A device capable of replying to requests or transmitting data contained within those devices autonomously.
- M2M area network (MAN): A network providing connectivity between M2M devices and gateways. Examples of M2M area networks include personal area network technologies such as (wireless) IEEE 802.15, short range devices (SRD), UWB, ZigBee, Bluetooth, and others, and (wired) CanBus, Modbus, KNX, LonWorks, PLC (Power Line Communication), and others.
- M2M gateway: The use of M2M capabilities to ensure that M2M devices interwork and interconnect to the communications networks.
- M2M communications networks: Communications networks between M2M gateways and M2M applications servers. They can be further broken down into access, transport, and core networks. Examples include xDSL, PLC, satellite, LTE, GERAN, UTRAN, eUTRAN, W-LAN, WiMAX, and others.
- M2M application server: The middleware layer where data goes through the various application services and is used by the specific business-processing engines.

The M2M platform middleware normally covers the layers from M2M gateway to the M2M application server. As an example, Actility (Active Utility) offers core infrastructure components and software enabling mass-scale, mission-critical applications of the Internet of Things with a specific focus on smart grid applications. Actility designed ThingPark® (http://www.actility.com/thingpark), a hosting infrastructure and marketplace for M2M/IoT applications managing data flows based on open architectures such as ETSI M2M. Realizing the IoT was still missing an architectural framework capable of handling such scale and truly enabling interoperability, Actility became a contributor to the ETSI M2M architecture-level standard and decided to develop an open-source implementation for IoT gateway developers, embedding all major existing M2M, sensors, and automation protocols. All referenced hardware platforms support OSGi (Open Services Gateway initiative framework) execution of ThingLets® and are remotely configured by the ThingPark infrastructure.

ComSoc Communities, an IEEE (Institute of Electrical and Electronics Engineers) collaboration of industry professionals, reports that United Parcel Service and Cinterion with TZ Medical have adopted ETSI standardized M2M applications, each for different purposes. UPS was able to achieve a 3.3 percent reduction in the amount of fuel consumed per package in the United States, and to reduce engine idling time by 15.4 percent in 2010. The system uses M2M technology comparable to iMetrik's to monitor and wirelessly report vehicle performance and driving habits, and route information to a central location. Cinterion and TZ Medical collaborated to launch a new heart-monitoring device to detect cardiac abnormalities in patients and communicate the diagnostic data to physicians through mobile networks and the Internet. Designated caregivers can track patient data at any time, from any place, to make treatment decisions.

The existing OMA (Open Mobile Alliance) and its M2M Task Force is producing a white paper that identifies M2M standards gaps and recommendations for OMA actions. Several OMA standards provide building blocks that map into the ETSI M2M framework:

- Device management can provide ETSI's remote entity management service
- Gateway management object fulfills some ETSI gateway service requirements
- Firmware updates, software updates, provisioning, diagnostics, and monitoring
- Converged personal network services maps into ETSI M2M area network

- Reachability, address mapping, inter/intra-area-network messaging, service publication and discovery
- Some OMA enablers (e.g., location) support services that can be used in M2M applications

An OMA graphic [235] shows how OMA enablers map to ETSI generic M2M framework (that applies to all M2M applications) using telematics application components as examples. OMA Converged Personal Network Services maps to ETSI M2M Area Network, GwMO maps to ETSI Gateway, OMA Device Management maps to ETSI Remote Entity Management, and so forth.

The Car-to-Car Communication Consortium, originally initiated by European vehicle manufacturers and now open to other partners, aims at providing a means to improve road safety by defining vehicle-to-vehicle and vehicle-to-infrastructure communications mechanisms, as described in Chapter 2. Multiple roadside units are deployed along roadsides. These devices communicate and act as a gateway for vehicular on-board units. Vehicle identifiers can use Internet Protocol version 6 (IPv6) addresses for the vehicles' on-board units. It is enhanced to enable geographical routing, which allows a sender's application to issue a message targeting recipients located in a certain geographical area. Also, International Organization for Standardization (ISO) TC204 WG16 (working together with ETSI Technical Committee Intelligent Transport Systems [TC ITS]) is drafting a series of standards under the acronym CALM (continuous communications air interface for long and medium range). The objective of this standard is to provide an architecture framework and a set of protocols for vehicle-roadside communications that separate applications from the communications media.

7.1.2 Frameworks for WSN

The Open Geospatial Consortium, Sensor Web Enablement (OGC SWE) standardization effort is intended to be a

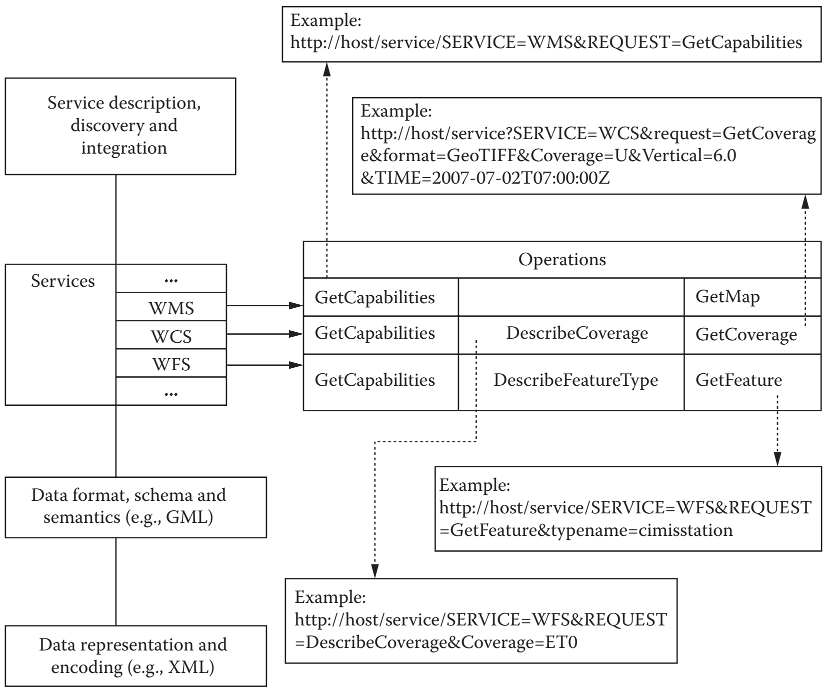
Example:
http://host/service/SERVICE=WMS&REQUEST=GetCapabilities
Service description, discovery and integration
Example:
http://host/service?SERVICE=WCS&request=GetCoverage&format=GeoTIFF&Coverage=U&Vertical=6.0&TIME=2007-07-02T07:00:00Z
Services
...
WMS
WCS
WFS
...
Operations
GetCapabilities
GetMap
GetCapabilities
DescribeCoverage
GetCoverage
GetCapabilities
DescribeFeatureType
GetFeature
Data format, schema and semantics (e.g., GML)
Example:
http://host/service/SERVICE=WFS&REQUEST=GetFeature&typename=cimisstation
Example:
http://host/service/SERVICE=WFS&REQUEST=DescribeCoverage&Coverage=ET0
Data representation and encoding (e.g., XML)

Figure 7.2 OGC services, operations, and example calls (indicated by dotted lines) for web map service (WMS), web coverage service (WCS), and web feature service (WFS). (From Michael Gertz, Carlos Rueda, and Jianting Zhang, "Interoperability and Data Integration in the Geosciences," in Arie Shoshani and Doron Rotem (Eds.), *Scientific Data Management: Challenges, Technology, and Deployment*, Boca Raton, FL: CRC Press, 2010.)

revolutionary approach for exploiting web-connected sensors such as flood gauges, air pollution monitors, satellite-borne earth-imaging devices, and so forth. The goal of SWE is creation of web-based sensor networks to make all sensors and repositories of sensor data discoverable, accessible, and where applicable, controllable via the World Wide Web (Figure 7.2).

SWE standards are developed and maintained by OGC members who participate in the OGC Technical Committee's SWE Working Group. SWE is a suite of standard encodings and web services that enable [106] the following:

- Discovery of sensors, processes, and observations
- Tasking of sensors or models
- Access to observations and observation streams
- Publish–subscribe capabilities for alerts
- Robust sensor system and process descriptions

The following web service specifications have been produced by the OGC SWE Working Group (in addition to the encoding specifications described in Chapter 6):

- Sensor observation service—standard web interface for accessing observations
- Sensor planning service—standard web interface for tasking sensor systems and model and requesting acquisitions
- Sensor alert service—standard web interface for publishing and subscribing to sensor alerts
- Web notification service—standard web interface for asynchronous notification

The USN (Ubiquitous Sensor Networks) standardization of ITU-T is another effort being carried out under the auspices of the Next-Generation Network Global Standards Initiative (NGN-GSI). USN is a conceptual network or framework built over existing physical networks that makes use of sensed data and provide knowledge services. Its main components are as follows:

- USN applications and services platform: technology framework to enable the effective use of a USN in a given application or service
- USN middleware: including functionalities for sensor network management and connectivity, event processing, sensor data mining, and so forth
- Network infrastructure: mainly based on NGNs, USN is not a physical network but rather a conceptual network making use of existing networks

- USN gateway: A node that interconnects sensor networks with other networks
- Sensor network: Network of interconnected sensor nodes (IP-based nodes with direct connection to NGN, non-IP-based nodes connected to NGN via gateways and others)

7.1.3 Standards for SCADA

ISO 16100-1:2009, one of the components of ISO 16100 standard for industrial automation systems and controls–IT convergence integration, specifies a framework for the interoperability of a set of software products used in the manufacturing domain and to facilitate its integration into a manufacturing application. This framework addresses information exchange models, software object models, interfaces, services, protocols, capability profiles, and conformance test methods.

ANSI/ISA-95 is an international standard for developing an automated interface between enterprise and control systems. This standard has been developed for global manufacturers. It was developed to be applied in all industries and in all sorts of processes, like batch processes, continuous and repetitive processes aiming to reduce cost, and risk and errors associated with implementing interfaces between enterprise and production control systems. It continues to be developed and refined by the Instrumentation, Systems, and Automation Society (IAS) in collaboration with major vendors of ERP and MES solutions around the world.

The objectives of ISA-95 are to provide consistent terminology that is a foundation for supplier and manufacturer communication, to provide consistent information models, and to provide a consistent operations model as a foundation for clarifying application functionality and how information is to be used.

The five parts of the ISA-95 standard are as follows:

- ANSI/ISA-95.00.01-2000, Enterprise-Control System Integration, Part 1: Models and Terminology
- ANSI/ISA-95.00.02-2001, Enterprise-Control System Integration, Part 2: Object Model Attributes
- ANSI/ISA-95.00.03-2005, Enterprise-Control System Integration, Part 3: Models of Manufacturing Operations Management
- ISA-95.04, Object Models & Attributes, Part 4: Object models and attributes for Manufacturing Operations Management
- ISA-95.05, B2M Transactions, Part 5: Business to Manufacturing Transactions

OPC Unified Architecture [118] brings two elementary innovations into the OPC world (Figure 7.3). On the one hand, the Microsoft Windows–specific protocol DCOM is replaced by open, platform-independent protocols with integrated security mechanisms. On the other, the proven OPC features, such as data access, alarms and events, and historical data access, are

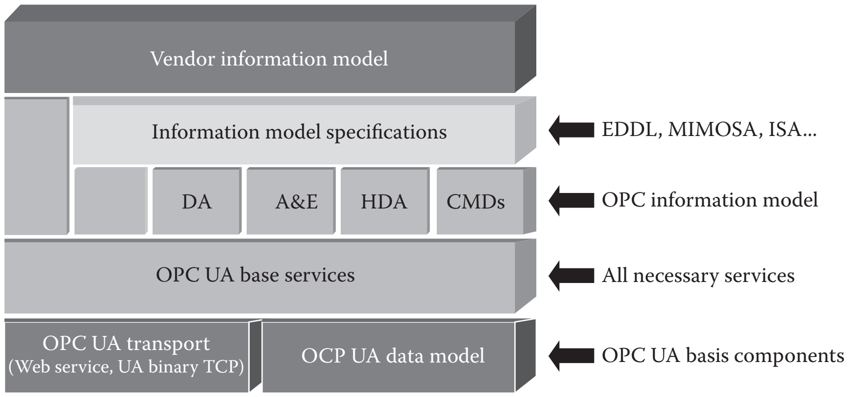

Figure 7.3 OPC unified architecture. (From Tuan Dang and Renaud Aubin, "OPC UA," in J. David Irwin (Ed.), *Industrial Communication Systems*, Boca Raton, FL: CRC Press, 1998.)

summarized in an object-oriented model and supplemented by new and powerful features, such as methods and type systems. As a result, not only can the OPC interface be directly integrated into systems on arbitrary platforms with different programming languages, but arbitrary complex systems can also be described completely with OPC UA. It can be implemented with JavaEE, Microsoft.NET, or C, eliminating the need to use a Microsoft Windows–based platform of earlier OPC versions. UA combines the functionality of the existing OPC interfaces with new technologies such as XML (extensible markup language) and web services to deliver higher-level MES and ERP support. The OPC Foundation and the MTConnect Institute announced their cooperation to ensure interoperability and consistency between the two standards in 2010.

ISA-95 and ISA-88 standards define information models for production control systems, batches, and MES. Their mapping to OPC UA is planned as an ISA-95 companion standard. Oracle provides ISA-95 standard–based integration capability between MES. Many other IT software vendors also provide ISA-95/88– and OPC UA–compliant products. For example, Wonderware's ArchestrA™ Platform provides support (http://global.wonderware.com/EN/PDF%20Library/Enterprise_Integration_Application_White_Paper.pdf) for open information standards ISA-95 and the message structures defined in ISA-95's B2MML (business-to-manufacturing markup language) messages.

On a different front, the SmartProducts [108] consortium aimed at demonstrating practical research that resulted in a platform that supports stand-alone or integrated, context-aware products for a range of application scenarios. It developed a scientific and technological basis for building smart products with embedded "proactive knowledge." Details about the different components of the middleware platform can be found in the following categories:

- Interaction: components for supporting the interaction between user and smart products
- Communication: components for supporting the information exchange and cooperation between different smart products
- Context: components for sensing, processing, and distributing context information
- Proactive knowledge base: components for handling the knowledge of a smart product
- Secure distributed storage: components for storing knowledge of a smart product in a secure and distributed way
- Tools: tools for developing smart products, such as for automatically extracting relevant information from manuals, editors

7.1.4 Extensions on RFID Standards

The EPCglobal-defined radio-frequency identification (RFID) architecture and frameworks are probably the most comprehensive and complete standards among the four pillar segments of IoT. Again, we will not talk about the EPCglobal effort because it was described in previous chapters of the book.

Many efforts to define IoT as described in Chapter 1 are of RFID origin. CASAGRAS (Coordination and Support Action for Global RFID-related Activities and Standardization) [110] was one of them, an FP7 project that ended in 2009 after 18 months (FP7 is Seventh Framework Programme, an initiative that bundles all research-related European Union initiatives together under a common roof playing a crucial role in reaching the goals of growth, competitiveness, and employment). The goal of CASAGRAS was to provide a framework of foundation studies to assist the European Commission and the global community in defining and accommodating international issues and developments concerning RFID with particular reference to the emerging Internet of Things. It seems that nothing particularly useful or better than EPCglobal was generated by this effort.

CASAGRAS2 started in June 2010 and ended in June 2012. The consortium consists of partners from Europe, the United States, China, Japan, Brazil, and Korea. The stated goal is to address the key international issues that are important in providing the foundations and cooperation necessary for realizing the Internet of Things as a global initiative.

BRIDGE [111] (Building Radio-frequency IDentification solutions for the Global Environment) is a European Union–funded three-year integrated project addressing ways to resolve the barriers to the implementation of RFID in Europe, based upon GS1 EPCglobal standards, by extending the EPC network architecture. One of the core aspects of BRIDGE related to IOT-A lies in the Discovery Service, which manages the exchange of RFID and aggregated information between nodes.

The Cross UBiQuitous Platform (CUBIQ) [112] project, in which nine organizations in Japan participate, aims to develop a common platform that facilitates the development of context-aware applications. The idea is to provide an integrated horizontal platform that offers unified data access, processing, and service federation on top of existing, heterogeneous IoT-architecture-based ubiquitous services. A unified data model was defined using USDL (universal service definition language). The CUBIQ architecture consists of three layers (and serverless real-time location search with RFID tags is an application example of the CUBIQ project)

- Mobile terminals with RFID tag reader collect RFID tag info and record location.
- The mobile terminals are connected via the core CUBIQ infrastructure and share RFID tag information.
- Observers can search RFID tag information to estimate the location of target person.

7.2 Unified Multitier WoT Architecture

Apart from the standard efforts such as the ETSI, 3rd Generation Partnership Project (3GPP), and Open Mobile Alliance (OMA), many research projects and industrial products aim to define and build a common middleware platform for WoT/IoT applications.

Niagara^{AX} is a software framework (http://www.neopsis.com/cms/en/solutions/niagara/) product and development environment that solves the challenges associated with building device-to-enterprise applications and distributed Internet-enabled automation systems, which are deployed in over 160,000 installations worldwide. Tridium's Niagara introduced the concept of a software framework that could normalize the data and behavior of diverse devices, regardless of manufacturer or communication protocol, to enable the implementation of seamless, Internet-connected, web-based systems. The data from diverse device systems are transformed into uniform software components. These components form the foundation for building applications to manage and control the devices. The Niagara^{AX} component model goes beyond unifying protocols and data from diverse systems to unifying the entire development environment used to build applications. Here are the Niagara^{AX} highlights:

- New graphics presentation framework and graphic development tool
- Comprehensive library of control objects
- New data archive model and flexible archive destinations
- New alarming capabilities that provides better visualization and user experience
- Reporting and business intelligence supports
- Open driver development toolkit
- Open application programming interfaces (APIs) for developers

FI-WARE of the European Union's Future Internet Core Platform project [236] aims to create a novel service infrastructure, building upon elements called generic enablers that offer reusable and commonly shared functions, making it easier to develop future Internet applications in multiple sectors.

This infrastructure will bring significant and quantifiable improvements in the performance, reliability, and production costs linked to Internet applications, building a true foundation for the future Internet. The reference architecture of the FI-WARE platform is structured along a number of technical chapters:

- Cloud Hosting
- Data/Context Management
- IoT Services Enablement
- Applications/Services Ecosystem and Delivery Framework
- Security
- Interface to Networks and Devices

However, details have not yet been worked out in this FI-WARE project. In the following sections, we will talk about some of the existing technologies that should be leveraged to build a FI-WARE-like unified horizontal framework.

7.2.1 SOA/EAI versus SODA/MAI

As described in the previous chapter, WoT/IoT applications should inherit and enhance the existing data formats and protocols, and the matching software frameworks to build platform middleware for WoT applications. SOAP (simple object access protocol), the successor of XML-RPC, is a protocol framework specification for exchanging structured information in the implementation of web services in computer networks. It relies on XML for its message format, and usually relies on other application layer protocols—most notably hypertext transfer protocol (HTTP), simple mail transfer protocol (SMTP), and Java messaging services (JMS)—for message negotiation and transmission. SOAP, which unified the CORBA, JavaEE, and .NET camps under one umbrella, can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built. SOAP can be a good generic WoT data exchange protocol considering it can tunnel easily over firewalls and proxies of existing infrastructure, among other advantages. Because of the verbose XML format, SOAP can be considerably slower than other competing middleware technologies such as CORBA; however, this may not be an issue when only small messages are sent, which is the case for machine-type communicaiton (MTC) of WoT.

The REST (representational state transfer interface) architecture was developed in parallel with HTTP/1.1, based on the existing design of HTTP/1.0, but it is not limited to the HTTP protocol. RESTful architectures can be based on other application layer protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state (Figure 7.4). REST is a lightweight SOAP. RESTful applications maximize the use of the preexisting, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on top of it. REST also attempts to minimize latency and network communication, while at the same time maximizing the independence and scalability of component implementations. The simple

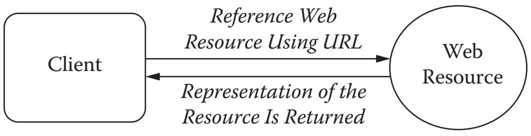
Client
Reference Web Resource Using URL
Representation of the Resource Is Returned
Web Resource

Figure 7.4 RESTful web services. (From Bhavani Thuraisingham, *Secure Semantic Service-Oriented Systems*, New York: Auerbach Publications, 2010.)

semantic of REST and its wide adoption helped provide services that have been reused in other domains like smartphone application. So, REST is a better framework protocol for MTC-based WoT/IoT applications.

For resource-constrained devices, CoAP (constrained application protocol) [127] is a specialized RESTful transfer protocol for use with constrained networks and nodes for M2M applications such as smart energy and building automation. These constrained nodes often have eight-bit microcontrollers with small amounts of ROM and RAM, while networks such as 6LoWPAN often have high packet error rates and a typical throughput of tens of kbit/s. CoAP, similar to SENSEI, provides the REST method/response interaction model between application endpoints, supports built-in resource discovery, and includes key web concepts such as URIs and content types. CoAP easily translates to HTTP for integration with the web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments. The Internet Engineering Task Force (IETF) recently approved a new working group called Constrained RESTful Environments (CoRE) based on CoAP's work. This new group aims at specifying a RESTful web service protocol for even the most constrained embedded devices and networks.

SOAP, REST, and CoAP are standard technologies for B2B-like integration of systems on the Internet at the M2M/IoT communication networks layer as described in Section 7.1.1, which should be part of the unified application framework data standards that works over the Internet. There are other standardized technologies such as ESB (enterprise service bus, Figure 7.5) based on MQ (message queue, and MQ_TT for resource constrained networks) and JMS for internal enterprise application integration (EAI) within intranet and extranet. Those technologies can be used for IoT application integration within an intranet or extranet, and they can also be used or extended to work over the Internet.

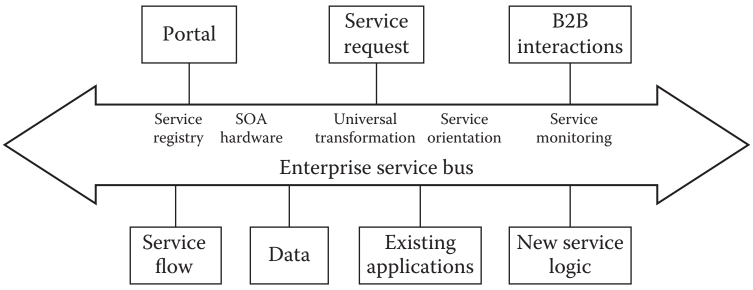

Figure 7.5 Enterprise service bus. (From Yurdaer Doganata, Lev Kozakov, and Mirko Jahn, "Software Architectures for Enterprise Applications," in Mostafa Hashem Sherif (Ed.), *Handbook of Enterprise Integration*, New York: Auerbach Publications, 2010.)

The JCA (Java Connector Architecture) is another good approach for WoT data collection and integration based on connectors or adaptors (the .NET architecture also has adaptors), which has been used in the author's team for the ^{ez}M2M platform middleware for WoT applications supporting many vertical sectors. JCA is also for internal EAI applications somewhat like OPC for SCADA (supervisory control and data acquisition). JCA-like architecture (http://en.wikipedia.org/wiki/Java_EE_Connector_Architecture) can be used at the M2M/IoT gateway layer. Examples that use adapter/connector architecture include http://www.opengate.es/, http://www.idigi.com/, and so on. Efforts should be spent on making JCA-like architecture work over the Internet if needed.

EAI and B2B seem related but they vary radically in their details. The first is entirely within a single administrative domain. If a new protocol does not work perfectly, it can be ripped out and replaced. In the cross-business environment, ripping it out affects customers, who may have no incentive to upgrade to the new protocol and will be annoyed if it changes constantly. Within a business, demand for a service can be fairly easily judged. On the external interface, demand can spike if a service turns out to be wildly popular with customers. Within a business, security can (to a certain extent) be maintained merely by firing people who abuse it. On the external interface, a much lower level of trust should be extended. Those principles apply to WoT (over the Internet and M2M application integration [74] within an Intranet) applications too.

A service-oriented architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services, usually over the Internet. Services comprise unassociated, loosely coupled units of functionality that have no calls to each other embedded in them. SOA requires metadata (unified WoT architecture also needs metadata) in sufficient detail to describe not only the characteristics of the promised services but also the data that drives them. The web services description language typically describes the services, while the SOAP protocol describes the communication protocols. One can, however, implement SOA using any service-based technology, such as REST, CORBA, or Jini, and any programming language.

Web services make functional building blocks accessible over standard Internet protocols independent of platforms and programming languages. These services can represent either new applications or just wrappers around existing legacy systems to make them network enabled. The Web Services Business Process Execution Language is an XML-based execution language that can be used to compose the coarse-grained services into broader services or complete applications. These powerful services are usually orchestrated into processes. The Universal Description Discovery and Integration specification defines a way to publish and discover information about web services as shown in Figure 7.6, which is also a function that WoT applications need.

The combination of the existing SOA (across Internet and extranet) and EAI (intranet) technologies is a good foundation for WoT/IoT applications. EAI can be extended for MAI (M2M

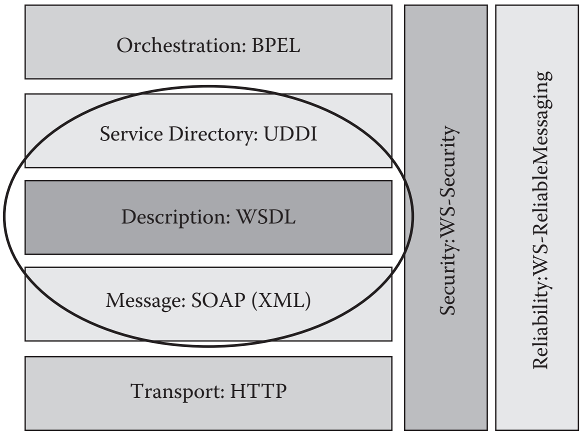

Figure 7.6 B2B technologies. (From Setrag Khoshafian, *Service Oriented Enterprises*, New York: Auerbach Publications, 2007.)

application integration) within an intranet. SOA can be used for WoT/IoT integration over the Internet and extranet.

In fact, a service-oriented device architecture (SODA) is proposed to enable device connection to an SOA (Figure 7.7). The SODA Alliance is an open, customer-driven, broad community chartered to promote consistent integration of the physical world into an SOA network [131]. As described before, developers have connected enterprise services to an ESB using the various web service standards since the advent of XML in 1998. With SODA, which can be based on the OSGi [177] framework described in the next section, developers are able to connect devices to the ESB, and users can access devices in exactly the same manner that they would access any other web service.

The core of the SODA standard is the DDL (device description language) based on XML encodings. DDL classifies devices into three categories: sensors, actuators, and complex devices. Figure 7.8 shows the DDL device model and a sample DDL file of an analog sensor [134]. The ATLAS platform of University of Florida is an implementation of the SODA standard.

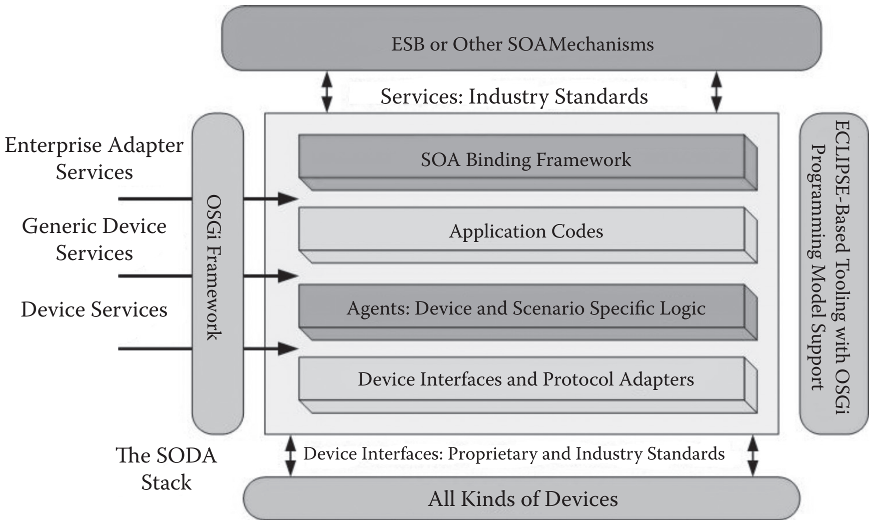
ESB or Other SOAMechanisms
Services: Industry Standards
Enterprise Adapter Services
Generic Device Services
Device Services
OSGi Framework
SOA Binding Framework
Application Codes
Agents: Device and Scenario Specific Logic
Device Interfaces and Protocol Adapters
ECLIPSE-Based Tooling with OSGi Programming Model Support
The SODA Stack
Device Interfaces: Proprietary and Industry Standards
All Kinds of Devices

Figure 7.7 SODA architecture.

```
<Sensor>
<Description>…</Description>
<Interface>
<Signal id = ″ADC1″>…</Signal>
<Reading id = ″Temp 1″>
<Type>Physical</Type>
<Measurement>Temperature</Measurement>
<Unit>Centigrade</Unit>
<Computation>
<Type>Formula</Type>
<Expression> Temp 1 = (((ADC1/1023 * 3.3)-0.5)*
(1000/10)</Expression>
</Computation>
</Reading>
</Interface>
</Sensor>
```

Figure 7.8 Example of the device description language of SODA.

The Open Healthcare Framework (OHF) is a project based on SODA formed for the purpose of expediting healthcare informatics technology including mHealth [132], or mobile health, a term used for the practice of medicine and public health, supported by mobile devices. The project is composed of extensible frameworks and tools that emphasize the use of existing and emerging standards to encourage interoperable open-source infrastructure, thereby lowering integration barriers. OHF currently provides tools and frameworks for devices and the HL7 (Health Level Seven), IHE (Integrating the Healthcare Enterprise), and other data formats and protocols.

7.2.2 OSGi: The Universal Middleware

The OSGi (Open Services Gateway initiative framework) [121] is a module system and service platform for the Java programming language that implements a complete and dynamic component model. The OSGi Alliance is an open standards organization founded in 1999 that originally specified and continues to maintain the OSGi standard. Using OSGi, applications or components (coming in the form of bundles for deployment) can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot. Management of Java packages/classes is specified in great detail. Application life cycle management (start, stop, install, etc.) is done via APIs that allow for remote downloading of management policies. The service registry allows bundles to detect the addition of new services or the removal of services, and adapt accordingly. OSGi can have a very small footprint [178] and run on ARM-based devices (e.g., ProSyst OSGi middleware) and operating systems such as Wind River, Android (also on top of a JVM), and so on.

The OSGi specifications have moved beyond the original focus of service gateways and are now used in applications

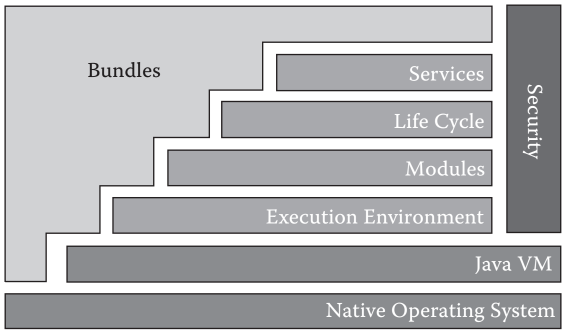

Figure 7.9 OSGi architecture.

ranging from mobile phones to the open-source Eclipse IDE (which dominates the IDE market). Other application areas include automobiles, industrial automation, building automation, PDAs, grid computing, entertainment, fleet management, and application servers (e.g., BEA Systems/Oracle Micro-kernel and SpringSource dm Server). It seems that it is specifically built for IoT/M2M applications considering it can fit in many places in the DCM value chain from device agents to cloud servers. OSGi is a universal middleware [130] and is going to play an important role, as a unified multitiered middleware architecture, in building WoT/IoT applications in many vertical segments (Figure 7.9).

The graphic at http://www.nec.co.jp/techrep/en/journal/g10/n02/100220-122.html depicts an M2M platform and device (gateway and agents) architecture based on the OSGi middleware framework. The ^{ez}M2M middleware platform (Figure 7.10) built by the author's team was also migrated from Java application servers to OSGi. OSGi-based platform middleware can provide both a traditional RCP (rich client platform) client-server and a web-based user interface on one platform, which is a very important feature needed by SCADA applications. Another WoT middleware platform built on top of OSGi is the Everyware Software Framework (http://esf.eurotech.com/doc/1.2systemArchitecture.html).

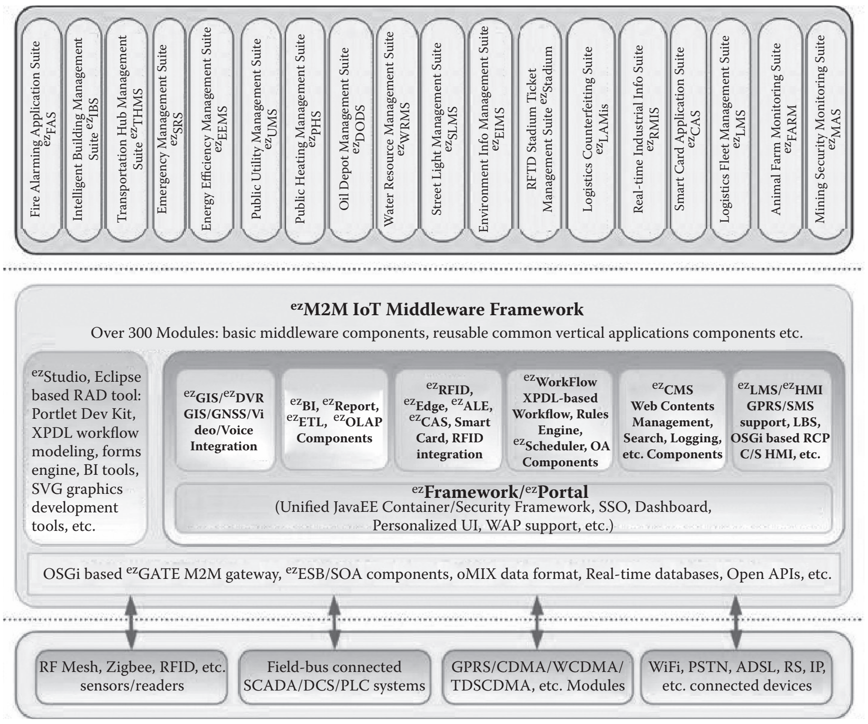
Fire Alarming Application Suite ezFAS
Intelligent Building Management Suite ezIBS
Transportation Hub Management Suite ezTHMS
Emergency Management Suite ezSRS
Energy Efficiency Management Suite ezEEMS
Public Utility Management Suite ezUMS
Public Heating Management Suite ezPHS
Oil Depot Management Suite ezDODS
Water Resource Management Suite ezWRMS
Street Light Management Suite ezSLMS
Environment Info Management Suite ezEIMS
RFTD Stadium Ticket Management Suite ezStadium
Logistics Counterfeiting Suite ezLAMis
Real-time Industrial Info Suite ezRMIS
Smart Card Application Suite ezCAS
Logistics Fleet Management Suite ezLMS
Animal Farm Monitoring Suite ezFARM
Mining Security Monitoring Suite ezMAS
ezM2M IoT Middleware Framework
Over 300 Modules: basic middleware components, reusable common vertical applications components etc.
ezStudio, Eclipse based RAD tool: Portlet Dev Kit, XPDL workflow modeling, forms engine, BI tools, SVG graphics development tools, etc.
ezGIS/ezDVR GIS/GNSS/Video/Voice Integration
ezBI, ezReport, ezETL, ezOLAP Components
ezRFID, ezEdge, ezALE, ezCAS, Smart Card, RFID integration
ezWorkFlow XPDL-based Workflow, Rules Engine, ezScheduler, OA Components
ezCMS Web Contents Management, Search, Logging, etc. Components
ezLMS/ezHMI GPRS/SMS support, LBS, OSGi based RCP C/S HMI, etc.
ezFramework/ezPortal
(Unified JavaEE Container/Security Framework, SSO, Dashboard, Personalized UI, WAP support, etc.)
OSGi based ezGATE M2M gateway, ezESB/SOA components, oMIX data format, Real-time databases, Open APIs, etc.
RF Mesh, Zigbee, RFID, etc. sensors/readers
Field-bus connected SCADA/DCS/PLC systems
GPRS/CDMA/WCDMA/TDSCDMA, etc. Modules
WiFi, PSTN, ADSL, RS, IP, etc. connected devices

Figure 7.10 ^{ez}**M2M platform middleware.**

7.2.3 WoT Framework Based on Data Standards

As discussed in Chapter 5, the platform middleware of WoT can itself be multitiered, just like the multitiered application servers for web applications. In fact, the best realistic approach should be using the existing platform middleware described in the previous two sections to build web-based WoT/IoT applications. An example of such a multitiered architecture is IBM's WebSphere Everyplace (now part of MQ-TT) Device Manager that is based on the three-tier WebSphere application server.

Another three-tiered IoT platform middleware named ^{ez}M2M was built by the author's team starting in 2003. It is based on the JavaEE technology and runs on top of three-tiered Java application servers such as JBoss, WebSphere,

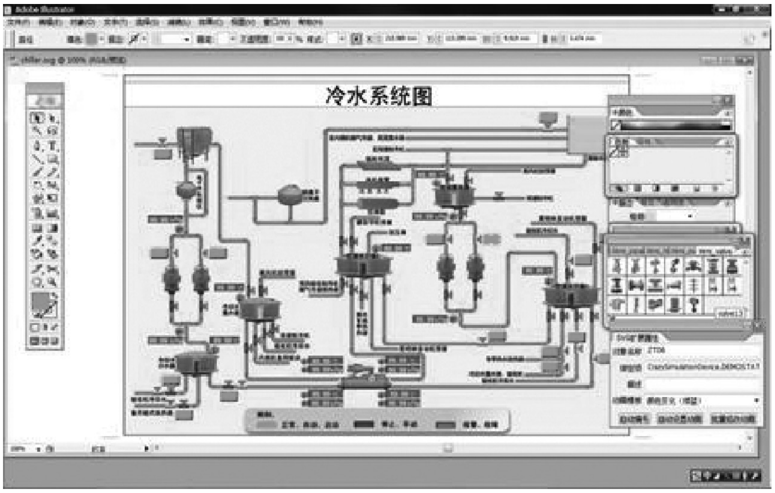

Figure 7.11 ^{ez}Studio IoT application IDE.

WebLogic, and others. More than 18 vertical IoT application suites have been developed on top of ^{ez}M2M, and more than 800 hundred IoT projects have been implemented worldwide, mostly in China.

Figure 7.11 depicts the ^{ez}Studio RAD (rapid application development) environment with an SVG graphic created based on the graphic library that comes with the IDE tool.

Also, a number of new and existing software makers are riding the IoT wave and created middleware for IoT. Some new paradigms have been introduced. For example, Axeda introduced device relation management, OMA proposed MDM (mobile device management), another is intelligent device management, and so forth. An example of MDM implementation is the Fromdistance MDM framework (http://www.empower.com.my/Fromdistance%20MDM.pdf).

Based on the sample middleware platform, a unified multi-tiered IoT middleware can be categorized as having layers as shown in Figure 7.12. The bold outlined blocks are extra tiers that are added to the existing three-tiered application server architecture.

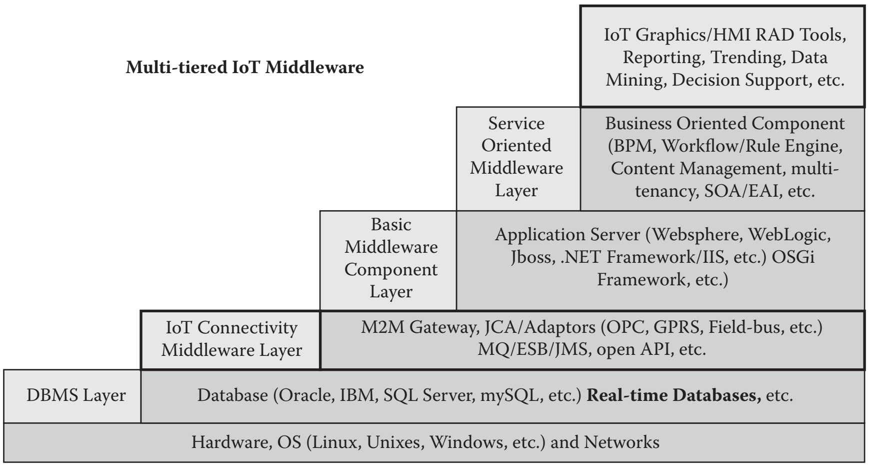
Multi-tiered IoT Middleware
IoT Graphics/HMI RAD Tools, Reporting, Trending, Data Mining, Decision Support, etc.
Service Oriented Middleware Layer
Business Oriented Component (BPM, Workflow/Rule Engine, Content Management, multi-tenancy, SOA/EAI, etc.
Basic Middleware Component Layer
Application Server (Websphere, WebLogic, Jboss, .NET Framework/IIS, etc.) OSGi Framework, etc.)
IoT Connectivity Middleware Layer
M2M Gateway, JCA/Adaptors (OPC, GPRS, Field-bus, etc.) MQ/ESB/JMS, open API, etc.
DBMS Layer
Database (Oracle, IBM, SQL Server, mySQL, etc.) Real-time Databases, etc.
Hardware, OS (Linux, Unixes, Windows, etc.) and Networks

Figure 7.12 Multitiered IoT middleware.

The following additional functionalities and tools are added to IoT middleware:

- Drag-and-drop/WYSIWYG (what you see is what you get) graphics and animation development and deployment tools with embedded scripts (Figure 7.13); RAD tools without programming
- BPM/rules engine (no programming required)-based IoT event/alert handling and actions
- M2M gateway, communication adaptors, open and standard API, real-time databases, and so forth

The unified horizontal WoT platform middleware will collect data from the M2M/IoT gateway level and up (or similar level for other WoT pillar systems) as defined by the ETSI/3GPP GSC efforts noted in Section 7.1.1. As an example, the deployment scenario of the M2M software development platform built by InterDigital [137] conforms to ETSI M2M Release 1 standards; however, it doesn't have a unified reference architecture yet.

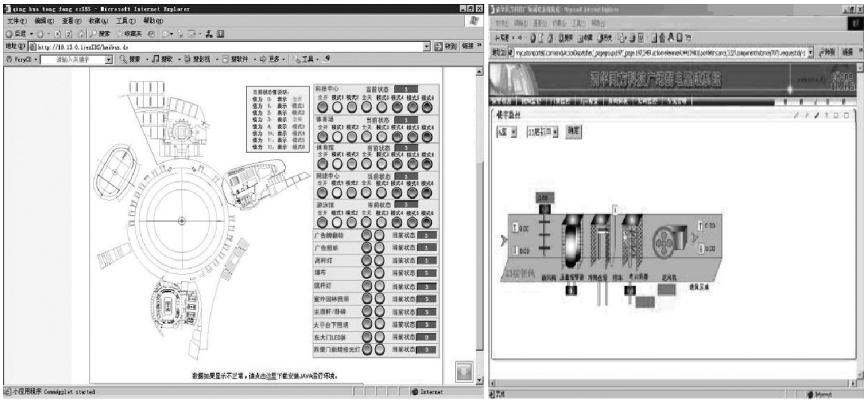

Figure 7.13 IoT graphics with animation.

The IoT-A project of the European Union has just delivered the specification version 1.2 [113] that created a reference architecture. However, this reference architecture considered WSN and RFID only in the unified communication layer, which is not a completely unified IoT architecture as of yet.

Figure 7.14 depicts the unified IoT middleware framework/architecture proposed by the author as a summarization of the previous chapters.

The IoT gateways (that behave as JCA-like IoT adaptors) will be connected to the M2M communication networks, on which the ESB-like (incorporating REST/SOAP functionalities) M2M/IoT communication middleware (we can call it the IoT bus) will reside. The IoT bus will be the IoT integration middleware similar to the SOA/EAI middleware that collect data from all the IoT adaptors, which represent or are the hubs connecting the IoT nodes or subsystems. The IoT platform middleware will finally integrate all the data from the IoT adaptors via the IoT bus. Figure 7.14 is the unified multitiered WoT application architecture framework based on the platform middleware. The multitiered architecture is summarized as follows

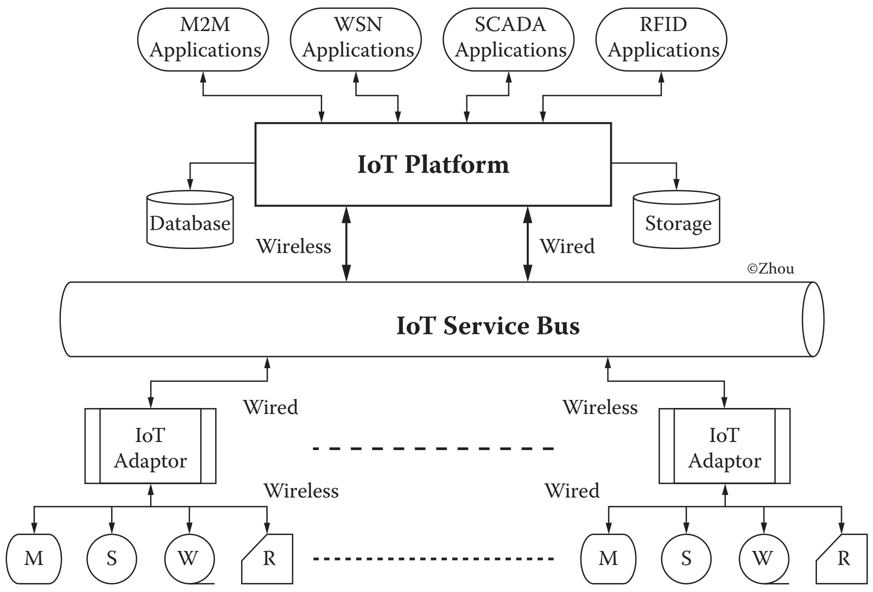

Figure 7.14 The unified IoT middleware framework.

(The ^{ez}M2M platform as shown in Figure 7.10 is a reference implementation of this architecture):

- Application framework SES (smart enterprise suite)–like layer for four-pillar applications
- The IoT platform middleware based on application server (container)
- IoT services bus based on ESB (REST/SOAP/MQ/JMS, etc.) and unified XML data format and protocol described in the previous chapter
- IoT adaptor based on the JCA-like adaptor technology for M2M/IoT gateway for device subnet or subsystems
- The back end of IoT hosted by cloud infrastructure and provides IoT cloud (MAI or XaaS) services
- The different devices or sensors of four pillars are connected via the IoT gateways to the IoT bus. They could be mixed or in small birds-of-a-feather groups.

Recent developments on PaaS and SaaS also adopted the approach of extending the application server platforms to have multitenant or massive multitenant supports for cloud computing. On top of it is integration middleware, which is the foundation of EAI. The application-level integration middleware layer is also called SES by some research firms, and together with the platform middleware they become application frameworks for different vertical applications. Some packaged applications such as ERP, Supply Chain Management (SCM), Manufacturing Execution System (MES), and others are built on top of those middleware frameworks. The multitiered WoT architecture can also use the PaaS/SaaS technology. This will be described in the following chapters.

To summarize, many individual standards development organizations want to incorporate existing standards into a unified conceptual framework as much as possible following the same approach described in this chapter. Rather than reinvent what already exists, these organizations prefer to identify and fill gaps and to integrate what already exists into the unified horizontal framework described in Chapter 3. This approach recognizes that it is impossible, or at least undesirable, to try to define new physical layer technologies, networking layer protocols, or platform middleware-based application frameworks for every current or future potential WoT/M2M application. Different vertical applications will optimize for individual cost and functionality requirements, while a standardized service layer will facilitate cross-vertical application development.

ABI Research believes that initial proposals for such a unified framework and service layer could be available by early 2012. It would likely take another 18–24 months for this initial proposal to be formally published as a standard or set of standards. ABI Research doesn't expect these efforts to start having an impact until late 2013 to early 2014. When such standards are in place, they will play an important role in driving overall WoT/IoT market development.

7.3 WoT Portals and Business Intelligence

A web portal or links page is a website that functions as a point of access to information in the World Wide Web. A portal presents information from diverse sources in a unified way. At the beginning of the web-based Internet revolution, web portals played a crucial role in making the web popular among the general public. Examples of public web portals include Yahoo, AOL, Excite, MSN, and more recently, iGoogle. Apart from the standard search engine feature, web portals offer other services such as e-mail, news, stock prices, information, databases, and entertainment.

In the portal craze of the late 1990s, the web portal was a hot commodity. After the proliferation of web browsers in the late 1990s, many companies tried to build or acquire a portal to have a piece of the Internet market. Netscape became a part of America Online, the Walt Disney Company launched Go.com, IBM and others launched Prodigy, Excite and @Home became a part of AT&T, and so forth.

There are two broad categorizations of portals: horizontal portals, which cover many areas, and vertical portals, which are focused on one functional area. A vertical portal called vortal consequently is a specialized entry point to a specific market or industry niche, subject area, or interest. WoT portals are vertical portals.

By the same token, WoT portals also started to appear; some of the well-known ones are as follows:

- Pachube (https://pachube.com): Pachube ("patch-bay") (renamed Cosm recently), the "Plumber" of Internet, connects people to devices, applications, and the Internet of Things. As a web-based service built to manage the world's real-time data (has been used to monitor the radiation in Japan caused by the 2011 earthquake and tsunami), gives people the power to share, collaborate, and make use of information generated from the world around them.

- SensorMap (Microsoft, http://atom.research.microsoft.com/sensewebv3/sensormap/): The portal and its accompanying tools will allow for more online live data. Microsoft Hohm is another WoT project similar to Google PowerMeter, which is to be discontinued in 2012.
- Google PowerMeter (http://www.google.com/powermeter/about/): PowerMeter, retired in September 2011, included key features like visualizations of energy usage, the ability to share information with others, and personalized recommendations to save energy.
- Sun SPOT (small programmable object technology): Programming the world with Java, the Oracle Sun SPOT project explores wireless transducer technologies that enable the emerging network of things, building a hardware and software research platform to overcome the challenges that currently inhibit development of tiny sensing devices.

As intranets grew in size and complexity, webmasters were faced with increasing content and user management challenges. A consolidated view of company information was judged insufficient. Users wanted personalization and customization. *EIPs (enterprise information portals)* also became common after the public portals. EIP solutions can also include workflow management, collaboration between work groups, and policy-managed content publication. Most can allow internal and external access to specific corporate information using secure authentication or single sign-on.

Java Specification Request (JSR168) standards emerged around 2001. JSR168 standards allow the interoperability of *portlets* across different portal platforms. These standards allow portal developers, administrators, and consumers to integrate standards-based portals and portlets across a variety of vendor solutions. The concept of content aggregation seems to continue gaining momentum, and portal solutions will likely continue to evolve significantly over the next few years. The Gartner Group predicts generation 8 portals to expand on the business mashups concept of delivering a variety of information, tools, applications, and access points through a single mechanism. This technology should also be considered in WoT applications; for example, the ^{ez}M2M middleware platform developed by the author used the JSR168 standard portlet technology, mostly for Intranet IoT applications. As an example, an IoT platform with EIP portal and dashboard support can be found at http://iobridge.com/.

On the other hand, there is a need for a set of *ontologies* to marry sensor data and sensing information with meaning. The W3C's working group on semantic sensor networks is currently developing some definitive examples using RDF metadata model and related technologies discussed in the previous chapter. Additionally, real-time extension of the semantic sensor web concept is being developed, called Sensor Wiki. The motivation behind this concept is to allow real-time browsing of the physical world consistent with the situational awareness goal. Understanding the physical world via a myriad of sensors is now possible.

In a sensor Wiki, one or more sensors contribute real-time information as Wiki pages with suitable themes and formats useful to prospective Sensor Wiki users. Sensor Wiki users can look up information about objects, events, or places of interest interactively. They can also add intelligent interpretations of what they observe, use sensor tasking to add to the content to improve accuracy, or even develop the overall scene to offer situation assessment on a proactive basis. Others might want to record such sensor streams and related information as part of a larger objective such as future planning, training, or simply record keeping for historical purposes, and make it available to a specific community or an individual.

On a more practical basis, when enormous amount of data are collected in a IoT system, data mining can be conducted to acquire business intelligence (BI) and help decision support. Data mining deals with finding patterns in data that are by user definition, interesting, and valid. It is an interdisciplinary area involving databases, machine learning, pattern recognition, statistics, visualization, and others. Decision support focuses on developing systems to help decision-makers solve problems. Decision support provides a selection of data analysis; simulation; visualization; modeling techniques; and software tools such as decision support systems, group decision support and mediation systems, expert systems, databases, and data warehouses.

BI technologies provide historical, current, and predictive views of business operations. Common functions of BI technologies are extract, transform, and load (http://ckbooks.com/computers/data-warehousing/extract-transform-load-etl/) as well as reporting, online analytical processing, analytics, data mining, process mining, complex event processing, business performance management, benchmarking, text mining, predictive analytics, and so on.

For example, in many SCADA applications, BI is widely used. SCADA software vendors provide a number of relevant products such as CitectSCADA Reports, Wonderware Intelligence, Acumence Plant Analytics Server, and others. Also, a BI analytics of data from a fleet management system in China's truck/bus industry (partner of the author's current company) reveals that fuel usage can differ as much as 15 percent due to driver behavior for the same truck, route, and mileage.

7.4 Challenges of IoT Information Security

The security issue of the IoT is always an issue of concern, just like the security issue of all ICT systems. In the context of the IoT, because most of the "Things" (devices, assets, equipment, facilities, etc.) are owned by certain entities, the *ownership* characteristics make the security concern of IoT systems even more significant, often more tricky to deal with than the existing documents processing dominant ICT systems. For example, the whereabouts and size of an RFID-tagged and tracked nuclear warhead on the move could be exposed on the Internet if the security system is hacked. Privacy, such as the location of an object or a person, is one of the most concerning issues. Also, the sheer number of devices to be managed will add complexity to the existing security measurements. The Internet of Things will no doubt present new security challenges in cryptographic security, credentialing, and identity management.

Some technologically disadvantaged countries are worried about the security issues more than others. In China, for example, experts are evaluating potential new security threats brought in by the broad implementation of the Internet of Things across national borders. They are worried about the connected IoT systems such as the power grid, transportation (railways, airways, and roads) systems, water supply system, oil and gas pipelines, and so forth being compromised by third parties and losing *information sovereignty*, considering that developed countries such as the United States have programs such as the U.S. Army Signal Command (ASC), which provides support for the war-fighting commanders to win the information war. ASC directs the activities of some 15,000 soldiers and civilians in more than a dozen nations around the world. The USANETCOM (USA Network Enterprise Technology Command) [135] makes the continental U.S.-centered army capable of executing a force-projection mission through its integrated, worldwide theater tactical information assets. In theaters outside the continental United States, the USANETCOM provides the total spectrum of information services through centralized operation and maintenance of European, Southwest Asian, Pacific, and Central American strategic, theater, tactical, and sustaining-base information systems. On the other hand, developed countries are accusing underdeveloped countries of hacking into their systems to steal information and technologies.

However, there is no fundamental difference between IoT security and the traditional ICT system security. People have been putting hard-earned money (the most important assets) in banks and access accounts and do transfers via the Internet without much problem. The security measurements of the current ICT systems have eight dimensions [237]: access control, authentication, nonrepudiation, data confidentiality, communication security, data integrity, availability, and privacy. These technologies still apply to IoT systems and cover most, if not all, of the IoT security concerns and requirements, especially at the early stages of IoT development.

Some of the specific security issues that are more concerned with IoT application scenarios include the following (as summarized by the Association for Automatic Identification and Mobility, taking the RFID scenario as an example but applicable to all IoT systems):

- Skimming: Data are read directly from the tag without the knowledge or acknowledgment of the tag or device holder.
- Eavesdropping or sniffing (also called “man-in-the-middle” reader): Unauthorized listening/intercepting.
- Data tampering: Unauthorized erasing of data to render the device useless, or changing the data.
- Spoofing: Duplicates device data and transmits it to a receiver to mimic a legitimate source.
- Cloning: Duplicates data of one device to another device.
- Malicious code: Insertion of an executable code/virus to corrupt the enterprise systems.
- Denial of access/service: Occurs when multiple devices or specially designed devices are used to overwhelm a receiver’s capacity, rendering the system inoperative.

- Killing: Physical or electronic destruction of the device deprives downstream users of its data.
- Jamming: The use of an electronic device to disrupt the receiver's function.
- Shielding: The use of mechanical means to prevent reading of a tag or device.

Due to the above issues and the capacity of devices and diverse networking conditions, a few challenges face the development of IoT in addition to traditional ICT security issues, especially at the advanced stages of IoT development:

- The 10 security issues listed above and the sheer number of devices involved will make the design and deployments of security solutions more complex.
- The heterogeneous, multihop, distributed networking environments make the passing and translation of security credentials and the end-to-end security functionalities a very difficult mission across the four categories of networks, that is, the long- and short-range wireless, and the long and short wired networks categorized in the previous chapters of this book.
- These cryptographic suites were designed with the expectation that significant resources (e.g., processor speed and memory) would be available. The differences of sizes, limited storage capacities, and constrained processing power of the devices also make the processing of public key infrastructure (PKI) encryption, decryption, and key management hard to be consistent along the entire data flow.
- The joining and leaving (bootstrapping) of devices into the IoT systems and the grouping of the mobile devices over dynamic networks also add complexity to the authentication and authorization process.

Some of the following security protocols are discussed as candidate solutions in the 6LoWPAN and CoRE IETF working groups [136].

- The Internet Key Exchange (IKEv2)/IPsec and MOBIKE (Mobility and Multi-homing IKEv2)
- The Host Identity Protocol (HIP) and a HIP variant for lossy low-power networks called Diet HIP
- Transport layer security (TLS) and its datagram-oriented variant DTLS secure transport-layer connections
- The Extensible Authentication Protocol (EAP)
- The Protocol for Carrying Authentication for Network Access (PANA)

Secure Middleware for Embedded Peer-to-Peer systems (SMEPP) is an EU/ICT project that built a security middleware framework for IoT applications [238].

KoolSpan's TrustChip® (http://www.koolspan.com/) is a fully hardened, self-contained security engine that aims to provide an end-to-end security solution over resource-constrained, heterogeneous networks.

7.5 Summary

Compared with data format standardization, the standardization of a unified IoT system architecture is more feasible and doable, especially the back end multitiered platform middleware architecture. This is one of the important conclusions drawn in this chapter.

Many projects worldwide but mostly in Europe have created a number of architectural specifications for IoT that cover one or more of the four pillar segments, some with reference implementation prototypes. Some companies such as

Tridium worldwide but mostly in the United States have also announced platform middleware products such as the Niagara Framework and ArchestrA of Wonderware for generic IoT applications. The aforementioned conclusion was drawn based on the investigation and analysis of those efforts and cases.

This chapter pointed out that the unified IoT architecture should be based on the existing multitiered middleware architecture, especially the JavaEE three-tiered application server architecture with support of related technologies such as SOA, ESB/EAI, OSGi, and others without reinventing the wheel. Other relevant technologies such as BI, information security, data formats, and more were also discussed in this chapter.

We will be talking about cloud computing and its synergy with IoT in the next two chapters.