Web Services Provide the Power to Integrate

A Web-Services-Based Framework for Integration of Power System Applications

Jun Zhu

1540-7977/03/$17.00©2003 IEEE

POWER SYSTEM APPLICATIONS ARE TYPICALLY built to automate business processes in electric utilities. Traditionally, these applications were designed as a discrete business function. The users of such an application fell within a single functional area in a utility organization. However, as business evolves, electric utility operations may involve multiple business processes across several functional areas. For example, today's operation of an electrical distribution system requires information from many sources, including circuit descriptions from the GIS, real-time measurements from the SCADA, distribution analysis from the DMS, customer information from the CIS, and even weather forecasts from some public services. With the advent of electric power deregulation, electric utility organizations also require integration of business processes beyond corporate boundaries. For example, to properly oversee safe and reliable operation of the transmission grid, independent system operators (ISOs) and regional transmission organizations (RTOs) must maintain operational power system models that span multiple service areas and regularly exchange power system models with their member utilities. These changing business conditions have resulted in a requirement for integration of heterogeneous legacy power system applications inside and outside an electric utility organization.

Web services, emerging as the next generation of integration technology, provide an attractive alternative for enterprise application integration. This article presents a Web-services-based integration framework to address some general integration issues common to electric utilities. As a case study, the Web-services-based integration framework is then applied to design an integrated utility information system.

Integration of Legacy Power System Applications: A Challenging Task

Seamless integration of legacy power system application is a challenging task. These legacy applications were developed when there were no open standards. Vendors developed and deployed these legacy systems using their proprietary technologies that are not interoperable with each other.

To address these integration issues common to electric utilities, a variety of industrial groups were formed to define a set of specifications that would facilitate the integration of heterogeneous power system applications. For example, the Common Information Model (CIM) specification was defined to facilitate the data exchange and promote the application interoperability in energy control centers. Another initiative is the Utility Integration Bus (UIB). The UIB provides a common information-bus-based integration model for data exchange and communication between utility information systems. It simplifies the integration of loosely coupled applications by replacing the traditional point-to-point proprietary interfaces with a universal API, which provides access to all participating applications.

While these specifications and initiatives successfully address some of the strategic integration issues unique to electrical utilities, implementation of these good integration strategies remains a challenging task. For example, traditional enterprise application integration (EAI) approaches normally involve developing middleware applications to communicate the noninteroperable applications using the message broker technology. Developing and deploying a compatible object request broker (ORB) with CORBA or DCOM is a fairly complex issue, a task requiring special expertise. Quite often, achievement of a good integration strategy is significantly constrained by many implementation restrictions in the traditional EAI solutions, such as application interoperability and implementation complexity, etc.

Web services have emerged as the next generation of integration technology. Based on open standards, the Web services technology allows any piece of software to communicate with each other in a standardized XML messaging system. It eliminates many of the interoperability issues that the traditional

Acronyms

AMR	Automatic meter reading
API	Application programming interface
ASP	Active server page
CIM	Common information model
CIS	Customer information system
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DMS	Distribution management system
DOM	Document Object Model
EAI	Enterprise application integration
EMS	Energy management system
GIS	Geographical information system
HTTP	HyperText Transfer Protocol
ISO	Independent system operator
J2EE	Java 2 Enterprise Edition
ORB	Object request broker
RDF	Resource Description Framework
RTO	Regional transmission organization
SAX	Simple API for XML
SCADA	Supervisory control and data acquisition
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
UI	User interface
UIB	Utility integration bus
WMS	Work management system
WSDL	Web Services Description Language
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations

```
<SOAP:Envelope>
  <SOAP:Body xmlns:m="http://www.ieee.org/pes-services">
    <m:GetLineImpedance>
      <m:ConductorSize>4/0</m:ConductorSize>
      <m:Length>500</m:Length>
      ...
    </m:GetLineImpedance>
  </SOAP:Body>
</SOAP:Envelope>
```

figure 1. A SOAP message requesting to model a transmission line.

```
<SOAP:Envelope>
  <SOAP:Body xmlns:m="http://www.ieee.org/pes-services">
    <m:GetLineImpedanceResponse>
      <m:PositiveResistance>0.5862</m:PositiveResistance>
      <m:PositiveReactance>0.8657</m:PositiveReactance>
      ...
    </m:GetLineImpedanceResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

figure 2. A SOAP response message containing the line modeling results.

EAI solutions have difficulty resolving. The underlying transport protocol behind Web services is based on XML over HTTP. With minimal programming, the Web services technology enables you to easily and rapidly wrap the legacy enterprise applications and expose their functionality through a uniform and widely accessible interface over the Internet.

What Are Web Services?

Software services are applications that expose message-based interfaces suitable for being accessed by other applications. As a new type of software service, Web services are modular, self-describing, and self-contained applications that can be published, located, and dynamically invoked across the Web.

The Web-services technology is built on the foundation of open standards and common infrastructure. The Web-services framework is divided into three areas—communication protocols, service descriptions, and service discovery, each specified by an open standard.

Web services are invoked using Simple Object Access Protocol (SOAP), a standard XML-based protocol for messaging on the Internet. Encoded in XML, SOAP provides a way to communicate between applications developed with different programming languages and running on different operating systems. At its core, a SOAP message has a very simple structure: an XML envelope element used to wrap the user-defined messages. Figure 1 shows such a SOAP envelope, where a message requesting to model a transmission line is contained. When the designated Web service receives the message, it calls the specified method with the input arguments and returns the output in a separate SOAP response message shown in Figure 2.

The interface exposed to the external Web services, such as the `GetLineImpedance` method in the above example, is described in the XML-based Web Services Description Language (WSDL). WSDL is used to describe the Web service, specifying its location and publishing its operations (i.e., methods). A WSDL document details the published function calls and provides users with a point of contact. In other words, WSDL provides a standard way to describe the Web service interfaces in enough detail to allow a user to build a client application talking to the described Web service.

Universal Description, Discovery, and Integration (UDDI) is a standard-based specification for service description and discovery. The UDDI specifications offer Web-service users or consumers a unified and systematic way to find service providers through a centralized registry of service. The registry of service is an automated online "phone directory," through which the registered Web services advertise their business services. Registry access is accomplished using a standard SOAP API for both querying and updating.

Combining these open standards, the Web-services technology provides a standardized way of publishing, locating, and invoking the business services. With minimal programming, a SOAP interface can be layered on top of a new or existing piece of code to allow it to be accessed by other applications over the Internet. Once a Web service is deployed to a portal, other applications can discover and invoke its services. These client application locates its services using its registered UDDI and determines the interface definitions of the services using its published WSDL.

Web services provide a distributed computing technology for integrating the business services on the Internet using open standards and XML encoding. The use of standard XML protocols makes Web services platform-, language-, and vendor-independent, thus an ideal candidate for use in enterprise application integration. Built on the foundation of

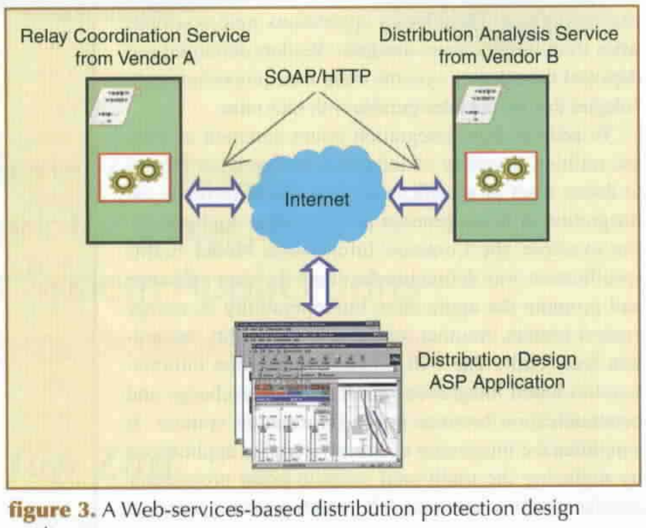
Relay Coordination Service from Vendor A
Distribution Analysis Service from Vendor B
SOAP/HTTP
Internet
Distribution Design ASP Application

figure 3. A Web-services-based distribution protection design application.

open standards, Web services eliminate the interoperability issues of the traditional EAI solutions. In addition, the technology provides a wealth of new opportunities because of its standardized and Internet-friendly method for integrating function calls. Although the Web-services technology is still in the realm of the early adopters, it represents an attractive alternative in the application integration space and has gained wide industry support in a very short period of time.

Potential Applications in the Power Industry

Web-services technology does offer a fundamentally different way to deliver business functions. It uses the open standards to expose business functions internally and to customers, suppliers, distributors, and trading partners. It opens new doors for many businesses and could fundamentally change the existing business models in many industries.

One of the benefits that the Web-services technology may offer to electric utilities is the great flexibility in choosing their planning and design tools. Most of today's power system planning and design software tools are monolithic and hard to customize. Electric utilities may end up maintaining multiple software tools that are not interoperable with each other. As adoption of Web services accelerates, some of the software product vendors will also position themselves as service providers. They will provide their expertise engineering services over the Internet using the Web-services technology. This will provide great flexibility to their utility clients. An electric utility may simply develop an integration application, such as an ASP application, to combine the selectively licensed services from various vendors. Under this scenario, as illustrated in Figure 3, a utility protection design engineer may choose Vendor A's relay coordination service to design the distribution protection system and then use Vendor B's distribution fault analysis service to verify the design. The customization business logic can also be easily added. In other words, the Web services technology allows electric utilities to build their customized planning and design tools by assembling their favorite engineering building blocks offered on the Internet.

The impact of the Web services technology on the area of energy management and marketing could be more significant. The Web services technology could result in a new e-business model that largely simplifies and accelerates the delivery of the vendor's proprietary technologies and fosters collaboration between heterogeneous real-time applications in a standardized way. Under this new business model, the electric utilities will focus on information consumption, rather than system maintenance and upgrading. Instead of spending their IT budgets chasing vendors' changing proprietary technologies, electric utilities will concentrate on consuming the energy management and marketing services subscribed from the selected system and integration vendors, leaving the costly maintenance and upgrading tasks to those who develop the systems. Figure 4 illustrates such a scenario, where the system vendors and the integration vendors are working together to run the Web-services-based utility's real-time information systems from their local sites.

While some of the above initiatives may be too ambitious, especially considering the Web-services technology itself is still at its early evolving stage, one area that the Web-services technology can have immediate impact on is the integration of existing legacy power system applications. Because Web services can be easily applied as a wrapping technology around existing legacy power system applications, new solutions can be deployed quickly and recomposed to address the changing business conditions. The remainder of this article will focus on investigation of this application.

A Web-Services-Based Integration Framework

Integration of power systems applications is typically achieved through function integration and data exchange. The former involves one application programmatically invoking code that lies in another application, while the latter describes

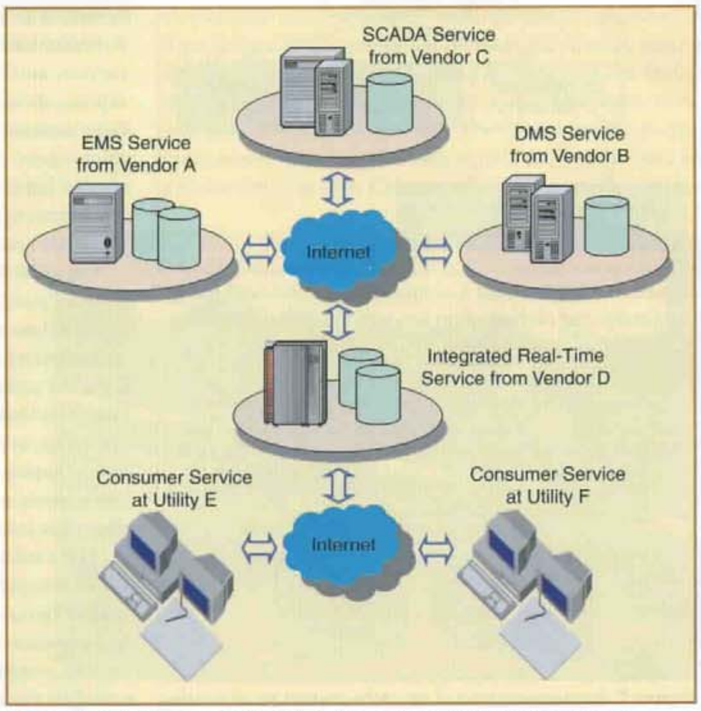
SCADA Service from Vendor C
EMS Service from Vendor A
DMS Service from Vendor B
Internet
Integrated Real-Time Service from Vendor D
Consumer Service at Utility E
Consumer Service at Utility F
Internet

figure 4. A Web-services-based utility real-time information system.

the use of tools that move data from one application to others. When properly integrated, an application may expose its data and functionality within the context of another without actually duplicating the application itself, thus resulting in a seamlessly integrated system. To address these common integration issues, a Web-services-based integration framework is presented. The framework provides the integration model and the implementation methodology. It also provides some guidelines on how to reengineer legacy power systems applications for Web-services-based integration.

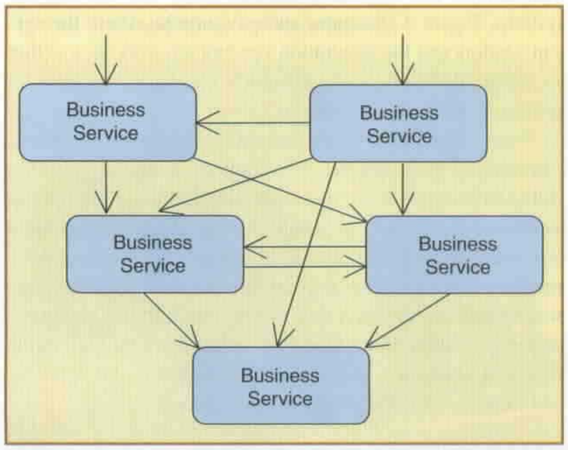

figure 5. An unmanageable function integration model allowing direct communications between business services.

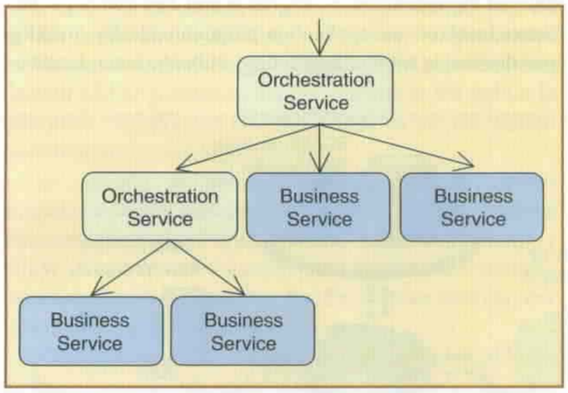

figure 6. A hierarchical function integration model using the centralized orchestration services to control the interaction of business functions.

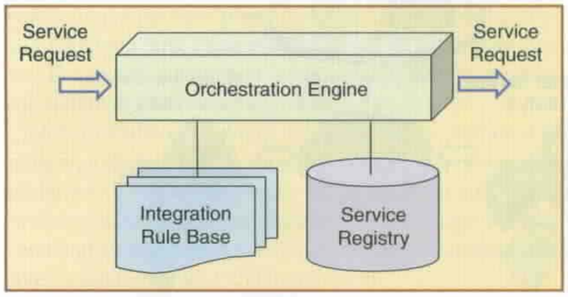

figure 7. Implementation of an orchestration service using the knowledge engineering approach.

Integration of Business Functions Using a Knowledge-Based Orchestration Service

Web-services-based function integration involves calling the business functions in other applications through SOAP messaging. When you create an integrated system by allowing participating business Web services to communicate with each other, it is very easy for the communications between them to get out of control, as illustrated in Figure 5. Because each Web service has to know the interfaces of the Web services that it calls, a large system can end up containing a multitude of dependencies. This can also result in the creation of multiple entry points into a system, thus further reducing manageability. In this scenario, the integration process is implicitly achieved through message passing between all of the participating business services, making it difficult to change the integration logic in line with your organizational needs. This also makes it harder to track where any problems lie in the system.

A logical solution to the problem presented above is to separate the integration procedures into a dedicated Web service referred to as orchestration service. An orchestration service controls the step-by-step actions of integration, moving the integration process from one state to another. At each step, it calls a participating business service and then moves on to the next step, which may require the use of a different business service. In this way, the orchestration service unifies its participating business functions and exposes them through a single entry point, therefore simplifying external communication. An orchestration service can also expose itself as a participating business service of another higher-level orchestration service. This could result in a hierarchy of a well-organized service network, as illustrated in Figure 6.

Using orchestration service to control the interaction of business functions and to provide one central service significantly enhances the manageability and maintainability within an integrated system. Instead of communicating with each involved participant, a business service only deals with its host orchestration service. Adding a new business function or upgrading an existing business function to the integrated system is largely simplified. In addition, the proposed function integration model significantly reduces the number of dependencies.

There are many ways to implement the orchestration service. A straightforward approach is hardcoding all of the integration business logic and the business function calls in the orchestration service. Such an implementation is quick and easy to accomplish. However, as more business services are added to the integration framework or the integration business logic changes, tremendous maintenance effort must be involved to put each piece together and make them work collaboratively.

A more flexible and robust way to implement an orchestration service is using the knowledge-engineering solution. Instead of intertwining the integration business logic with the orchestration process, we separate them by creating two knowledge bases: a service registry and an integration rule base, as shown in Figure 7. The service registry is a private UDDI registry designed to store the information of participating business services. The integration rule base is built to store the integration business logic, which specifies how to respond to each type of request. By separating such integration meta-data from the integration process, a generic orchestration engine can be built to drive the process. In response to each received request, the orchestration engine searches the integration rule base for the matched integration rule and then executes its specified integration steps. The integration rule contains all of the knowledge needed to handle a certain type of request, including a sequence of integration steps and exception handling steps, etc. However, it doesn't bind a certain business function call to an integration step. Instead, the orchestration engine relies on the information stored in the service registry and online Web discovery to determine which participating business service should be contacted and what function calls should be invoked. Based on these discoveries, the orchestration engine prepares the input, invokes the discovered functions, and waits for the response or moves on to the next step depending on the specified synchronization mode.

A major benefit of the knowledge-based implementation is its great flexibility. When a new type of request needs to be supported or existing integration rules change, only the integration rule base needs to be modified. No programming activities and redeployment are needed. When a new business service joins the community, only the registration process is needed. The orchestration service will automatically discover its services and invoke them as needed.

CIM/XML for Data Exchange

The Web services exchange information in XML, a universal format for structured documents. To support XML document processing, a variety of specification and standards have emerged, such as eXtensible Stylesheet Language (XSL), Document Object Model (DOM), Simple API for XML (SAX), Resource Description Framework (RDF), etc. Due to wide industry support, the XML-formatted documents are much more searchable, integratable, reusable, and manageable. In fact, converting proprietary documents to XML is the most economical way to add intelligence to your documents and to make them immediately consumable over the Internet.

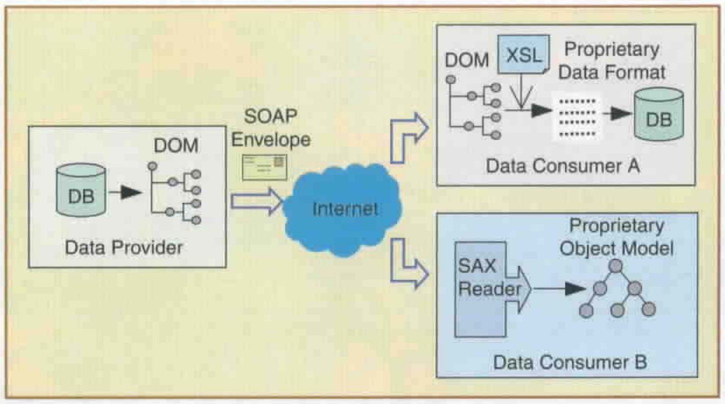

figure 8. XML-based data exchange.

Figure 8 illustrates how straightforward and flexible an XML-formatted proprietary document is extracted from the data provider, transferred over the Internet, and consumed by two different client applications. Consumer A restores the received XML data in a DOM object, which is later transformed to its proprietary data format by applying a predefined XSL style sheet. Consumer B, however, uses a different data consumption strategy. Instead of restoring the complete DOM object in memory, it uses a SAX reader to scan and filter the received XML document. Only the useful information is accumulated and used to populate its proprietary object model.

Seamless integration of vendors' proprietary information requires an industry-wide standard for describing power system resources, their attributes, and relationships. The CIM and its extensions were defined for this purpose. By encoding the CIM schema with the syntax and vocabularies of RDF, an XML-based knowledge representation language, a standard XML-based resource definition can be created to describe the power system resources. Specified by this common resource definition, the vendors' proprietary information can be converted as an XML document. All of the tags used to mark up the power system resources in this document are supplied by the CIM/RDF schema. The resulting format is called CIM/XML. Documents in this standard-based format can be readily parsed and consumed by the foreign systems as illustrated in Figure 9.

Reengineering Legacy Applications for Web-Services-Based Integration

Many of the business services in the integrated applications are provided by the enterprise legacy applications. Enterprises have made significant investments in their legacy applications. These applications have been functioning reliably and maturely with many years of maintenance efforts. They will continue to be useful in the new integrated environment. In order to bring these legacy applications into the new integrated environment, a reengineering process is needed to expose their business services over the Internet.

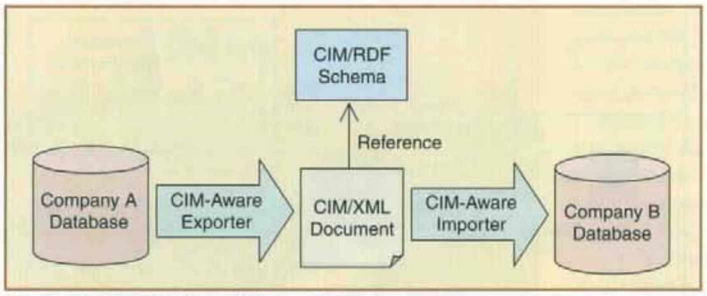

figure 9. CIM/XML-based data exchange.

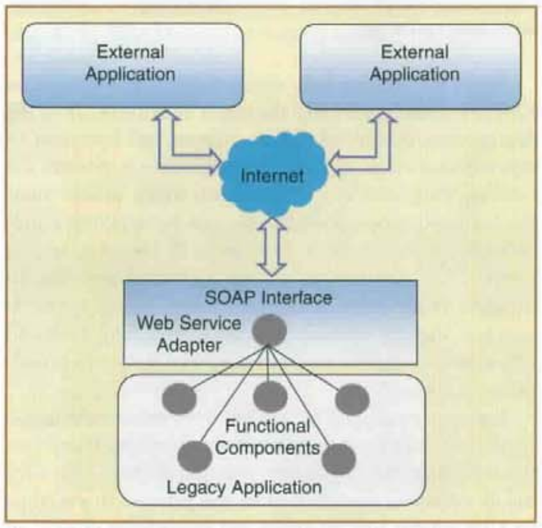

figure 10. Wrapping legacy applications for Web exposure.

Reengineering legacy applications for Web-services-based integration is typically accomplished by building a Web-service adapter that exposes the legacy application's functionality through the SOAP protocol. A Web-service adapter is a wrapper program that talks to the legacy application using its native protocol at one end and talks to the rest of the world in SOAP at the other end, as illustrated in Figure 10.

Design of an Integrated Utility Information System Based on Web Services

The Web-services-based integration framework is a generic integration infrastructure. In order to demonstrate how it can be effectively applied for integration of power system applications, a case study is conducted to design an integrated utility information system capable of providing a wide range of services.

Objective and Requirements

The objective of this case study is to design an integrated utility information system seamlessly unifying the legacy power system applications and providing an integrated environment to support various business functions within a utility. Specifically, the following requirements have been identified:

- ✔ The designed system should provide an open architecture, allowing both the existing legacy applications and the new leading-edge applications to be rapidly and seamlessly integrated into the system.
- ✔ Consolidating the functions of the participating applications, the designed system should be capable of providing integrated business services to various types of utility users, ranging from operators in the control centers to crew on the field.
- ✔ The design should successfully address the key design issues, including availability, performance, and scalability, etc. These key design issues are critical to mission control utility information systems.

Service-Based Architecture

The designed system architecture is shown in Figure 11. As shown in this figure, the designed system is service-oriented. All components in the system are services. Each service encapsulates behavior and publishes a messaging API to other collaborating services across the Web. The services in the system are classified into four categories: business services, utility services, consumer services, and orchestration services.

The business services are designed to address business requirements within one or more functional areas in a utility organization. For example, the SCADA service is designed to access real-time data and execute operation steps, and the GIS service provides automated mapping and facility management functions for distribution system operations. These services are built through a reengineering process: laying a SOAP interface on top of the existing legacy applications to allow them to be accessed over the Internet.

The utility Web services are developed to provide the common infrastructure supports for the integrated utility information system. These services are used to secure, automate, and facilitate the data exchange and function integration. For example, the authorize/authenticate service is built to secure the data and service access. The subscribe/publish service is designed to automate the data exchange and synchronize the replicated data in various data repositories. The CIM/XML merge service is used to merge the related CIM/XML fragments.

The consumer services initiate the service requests and consume the responses. These services are normally user interface (UI) services, though the external services can also be treated as consumer services. They perform tasks such as requesting information from a user or passing information to a user. Some UI consumer services are built to separate the

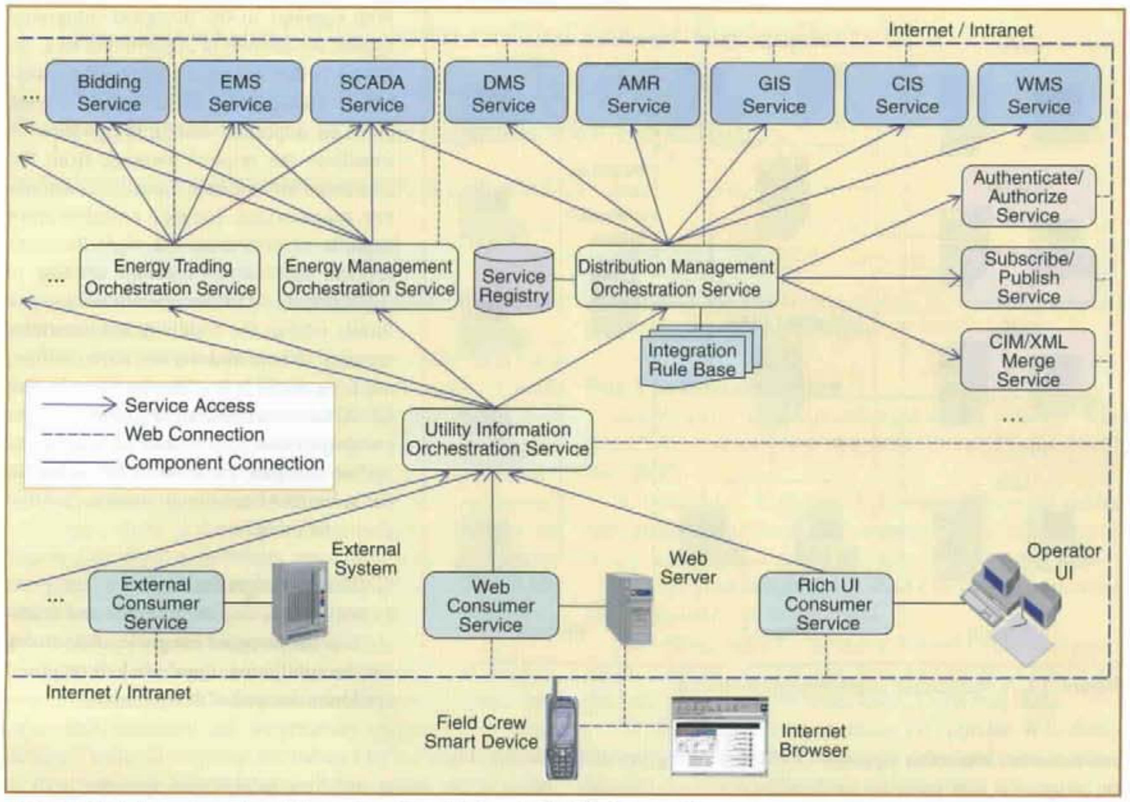

figure 11. Architecture of a Web-services-based integrated utility information system.

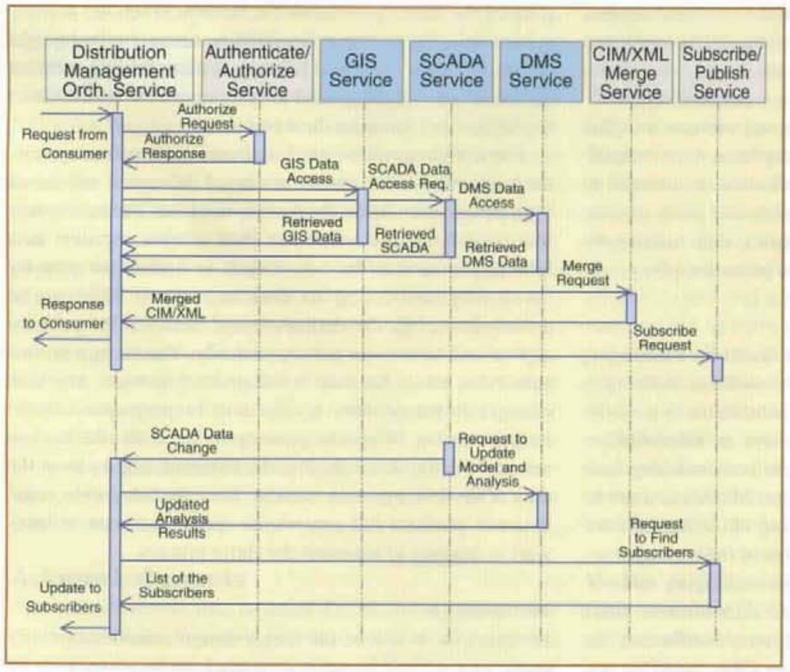

figure 12. Sequence diagram of distribution display call-up.

specific user interface implementation from the more general dialog flow and backend interactions. For example, a consumer service can be created to feed Web applications in a utility. Through the Internet server, this consumer service can power both Internet browser clients and the smart devices such as those used by the field crew. Other UI consumer services may be designed to support rich or thick UI applications. These services may cache the data for performance reasons. For example, a consumer service that feeds a distribution operator UI may cache the geographical information and network description data for the displayed distribution feeders.

The orchestration services discover the service resources and orchestrate the service interactions to achieve the just-in-time integration of applications. Each orches-

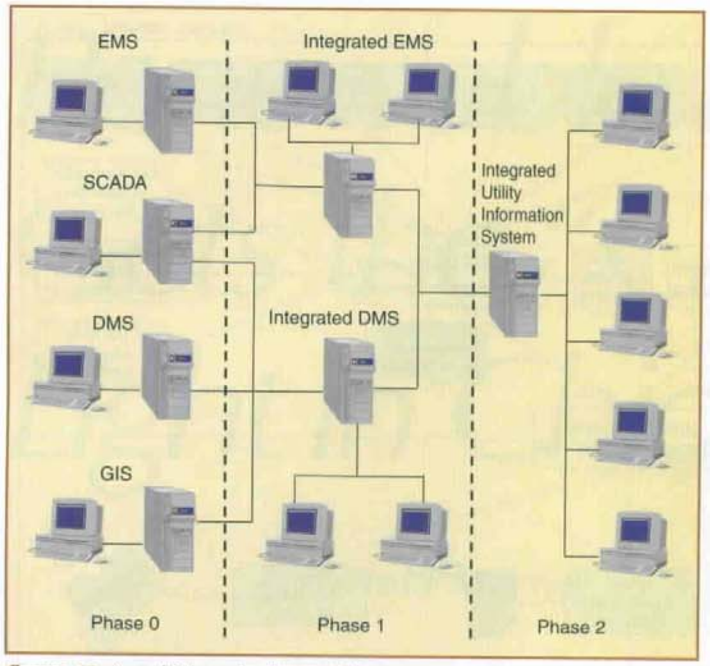

figure 13. A multiphase implementation strategy.

tration service maintains a private UDDI service registry and an integration rule base. An orchestration service can also serve as a business service of another orchestration service. For example, the distribution-management-orchestration service unifies all of the distribution management related services and presents them as a single business service in the integrated system. This further promotes service decoupling and allows us to manage the business services in an organized way.

The proposed service-based architecture presents a logical evolution from the isolated monolithic systems to an integrated system of services in a utility organization. It allows us to reuse the legacy power system applications and yields a loosely coupled but seamlessly integrated open system meeting the constantly changing requirements in the power industry.

Messaging and Collaboration

In a service-based system, services interact by exchanging messages. The most familiar form of message delivery is request/response: the sender opens a connection to a receiving port, posts the message, and receives an acknowledgement possibly combined with a response body resulting from the processing of the request message. Messages must be self-sufficient, containing or referencing all of the information necessary to understand the contents of the message.

A use case is used to illustrate the messaging and collaboration mechanism in the designed Web-services-based system. The use case involves calling up a distribution display that combines information from GIS, SCADA, and DMS. The sequence diagram in Figure 12 shows how the Web services in the designed integrated system collaborate to achieve this task. As shown in the sequence diagram, the distribution management orchestration service plays an important role in the process. It translates the request message from the consumer service into input for a discovery mechanism, locates collaborators capable of providing the right business service, orchestrates message sending to collaborators to invoke their services, and finally returns the results to the consumer service. When underlying data change, such as switch status changes in the SCADA, the orchestration service invokes corresponding business services for update and then broadcasts the update to the subscribed consumer services for display refreshing.

Critical Design Issues

In this section, we will examine and evaluate how the proposed integration framework can be configured, tuned, and recomposed to address the critical design issues.

Availability

Most of the utility real-time information systems, such as SCADA, EMS, and DMS, etc., require high availability. Traditionally, this is achieved by dedicating a redundant machine running the same applications for backup. When the primary system fails, the backup system will be automatically brought online, replacing the failed primary system. Special software and hardware may be needed to synchronize the status, detect the failure, and automate the transition.

For a Web-services-based real-time information system, the high availability can be achieved through a redundant Web service running on the backup machine. During the normal conditions, the redundant Web service registers as a hibernating service. Its only task is to imitate the primary service by replicating its data and status. This can be achieved through the data exchange between the primary service and backup service. Specifically, The backup service subscribes all of the data in the primary service. Any data changes in the primary service will be propagated to the backup service. When the primary service fails, the backup service will be discovered and awakened, taking over the duty of the failed primary service. Since the integration infrastructure provides full support, no special software or hardware is required to automate the above process.

Performance

Performance is one of the major design considerations for many time-critical power system applications, especially for the real-time applications. Considerable attention should be paid to the performance when designing an integrated utility information system using the Web-services technology.

Web-service applications are fundamentally based on XML and HTTP. Both could generate some overheads to your applications. An XML-wrapped data structure could bloat much more in size than its proprietary format. Message passing on the Internet is slower than on the local network. While today's Web-services technology may not deliver the same performance that you can get from traditional monolithic applications, there are many performance enhancement methodologies you can apply to achieve the same or even better performance for your Web-service applications. Among them, caching is the most frequently used technique. By caching infrequently changing data and meta-data, such as integration rules, service WSDL descriptions, data transform stylesheets, etc., you can minimize the frequency and volume of data exchange. Besides caching, there are many other techniques, such as threading, asynchronization, and SOAP attachment, etc., that can be applied to improve the performance for your Web-service applications.

Scalability

The integrated utility information system definitely has a larger user base. As more power system applications are added to the system, the user base will continue to grow at an accelerating speed. The scalability of the integrated system could be a major concern. Fortunately, the designed integrated utility information system is highly configurable. As the user base grows, new business Web services and hardware resources can be easily added to the system, meeting the growing service demands. This is normally accomplished by adding the Web services on the new hardware, modifying the integration knowledge base, and maybe adding some customization steps or help to balance loads. In this way, the newly added Web services and hardware resources will relieve the old services from heavy loading, therefore improving the system responsiveness and performance.

Adaptability

One of the major features of the design integrated utility information system is its adaptability to the changing environment. The orchestration service relies on online service discovery for request routing. When new services are added, or the old services are replaced with the new upgraded services, the integrated system can smartly adapt to the new environment. For example, when a new SCADA Web service from a different vendor replaces the existing one, the orchestration service will discover the type of service it offers and the syntax of its function calls. When a SCADA service is requested, it will automatically route the service request message to the new SCADA service in the dialect it understands. Due to the intelligent adaptability of the integrated system, addition of a new service or upgrading of an existing service is absolutely seamless and effortless.

Implementation Strategies

Compared with other integration approaches, the Web-services-based integration is much easier to accomplish. This is partially due to wide industry support. J2EE and .NET Framework have emerged as two mainstream platforms for Web-service implementation. Both platforms provide a rich library of run times, programming models, components, and tools for Web-services development and deployment. These supports have significantly simplified the Web-services application development and have led to stable, portable, and leading-edge applications.

The implementation of the designed integrated system will be more evolutionary than revolutionary. Since the Web-services-based integration mainly involves wrapping the legacy applications for Web exposure, it will not trigger a major or radical disruption to the existing utility information systems. The integration will be an evolutionary multiphase process. Figure 13 illustrates such an implementation strategy. As integration process progresses, more legacy applications will participate in the integrated system. On the other hand, the legacy systems can continue to operate as standalone until the integrated system becomes stable.

For Further Reading

M. Clark, *Web Services Business Strategies and Architectures.* Chicago, IL: Expert, 2002.

Ethan Cerami, *Web Services Essentials.* Sebastopol, CA: O'Reilly, 2002.

"Web services architecture requirements, W3C Working Draft 19," Aug. 2002. Available: http://www.w3.org/TR/ wsa-reqs

"Common Information Model (CIM) for the Control Center Application." Available: ftp://ftp.kemaconsulting.com/epriapi

"Utility Integration Bus (UIB) white paper," ver. 2.1, http://www.sisconet.com/downloads/uibintro.pdf.

G. Robinson, "Key standards for utility enterprise application Integration (EAI)." Available: http://scottneumann.com:5714/Presentations/StandardsUtilityEAI.doc

A. deVos, S. Widergren, and J. Zhu, "XML for CIM Model Exchange," presented at 22nd International Conference on Power Industry Computer Applications, Sydney, Australia, May 2001.

Biography

Jun Zhu received his B.S. degree from Huazhong University of Science and Technology (HUST), Wuhan, China; his M.E. degree from Nanjing Automation Research Inst. (NARI), Nanjing, China; and his Ph.D. degree from Clemson University, Clemson, South Carolina. He is currently with ALSTOM Energy Automation & Information (EAI), working on the development of distribution management systems. Previously, he developed distribution analysis software for Power Technologies, Inc. (PTI). Dr. Zhu is a corresponding member of IEC TC57 WG 14. He can be reached at jun.zhu@esca.com.