

SERVICE-ORIENTED ARCHITECTURE

Issue Date: January 2007

Abstract—Service-oriented architecture creates a framework wherein applications can use standardized Web services to share data. Using standardized Web services eliminates the need for proprietary, custom-developed middleware to specifically address how particular applications speak to one another. Eliminating middleware significantly reduces the cost and complexity of enterprise networks. Telecommunications service providers can build on the gains realized in enterprise architectures. Carriers benefit from deploying these architectures, both in managing their networks and customers and in deploying new services and applications.

Key Words—application integration, distributed architecture, IMS, OSS/BSS, service-oriented architecture, SOA, Web services

INTRODUCTION

Called “the next big thing in software,” service-oriented architecture (SOA) promises to change the way enterprises do business, reducing the cost and complexity associated with integrating software applications. The primary driver for change revolves around integration—specifically, a methodology for ease of communications among applications. As enterprises scale, the number of applications required to accomplish basic business processes scales as well; to accomplish a business function, the need to share data among applications becomes apparent. Getting applications to talk to one another is a problem that has plagued information services departments for decades. Cottage industries have been built around this issue, with millions of hours in productivity and huge dollar amounts in services and software spent trying to address it. Building on the innovations used to create the Internet’s communications structure, SOA promises to change the approach to enterprise integration, potentially eliminating the headaches involved in the inter-working of applications [1].

The framework for SOA extends well beyond enterprise computing; in fact, the SOA model can be applied in telecommunications provider networks to both increase the speed of launch for new services and reduce the required network infrastructure.

BACKGROUND

Fundamental to SOA is the *loose integration* of services and applications. What does this mean? To better understand loose integration as it pertains to an enterprise network, a basic understanding of distributed computing and *tight integration* of applications is helpful.

As enterprises began to automate their operations, multiple computing platforms were introduced. For example, an enterprise may have initially purchased a mainframe and used specialized software to perform accounting and financial tasks. That same enterprise may have used other software and systems to perform scientific computing, graphic design, document publishing, payroll processing, inventory, etc. At some point, the enterprise recognizes the benefits of integrating the software platforms. Consider, for instance, a business transaction involving the enterprise’s purchasing system and accounting system, each running on a different computing platform. When a bill is generated by the purchasing system, the outstanding amount needs to be passed to the accounting system as a receivable item. But first, the message must be translated, since the two platforms use two differently written software packages. A separate piece of software is written to convert the information flowing from the purchasing system into data readable by the accounting system

Brian Coombe
bcoombe@bechtel.com

An enterprise may face the choice of waiting to roll out new software or spending thousands of dollars to update interface software to address changes in a new release.

ABBREVIATIONS, ACRONYMS, AND TERMS

AS	accounting system	MRF	multimedia resource function
BGCF	breakout gateway control function	MRFC	MRF control
BSC	base station controller	MRFP	MRF processor
BSS	billing support system	ORB	object request broker
BTS	base transceiver station	OSS	operation support system
CORBA®	Common Object Request Broker Architecture	P-CSCF	proxy CSCF
CSCF	call session control function	PDF	policy decision function
ETSI	European Telecommunications Standardization Institute	PLMN	public land mobile network
GGSN	gateway GPRS support node	PSTN	public switched telephone network
GPRS	general packet radio service	RAN	radio access network
HLR	home location register	RMI	remote method invocation
HSS	home subscriber service	RNC	radio network controller
HTTP	hypertext transport protocol	SCIM	service capability interaction manager
I-CSCF	interrogating CSCF	S-CSCF	serving CSCF
IMS	IP multimedia subsystem	SGSN	serving GPRS support node
IOOP	inter-ORB object protocol	SIP	session initiation protocol
IP	Internet Protocol	SOA	service-oriented architecture
IPSec	IP security	SOAP	simple object access protocol
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector	UDDI	Universal Description, Discovery, and Integration
MGCF	media gateway control function	WLAN	wireless local area network
MGW	media gateway	W3C®	World Wide Web Consortium
		WSDL	Web services description language
		XML	extensible markup language

(see **Figure 1**). An additional process within this custom interface software, or a separate piece of software, is also required to convert the

information flowing in the opposite direction. This deployment scenario is known as tight integration of applications [2].

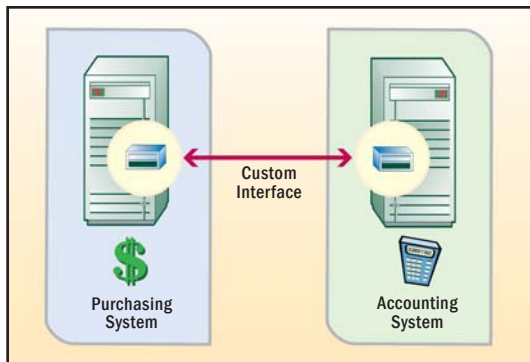


Figure 1. Separately Developed Custom Interface Allows Applications To Pass Information

For tightly integrated networks, developing custom interface software can prove to be a challenge. Developers must understand the inner workings of both applications; extensive testing is also required to ensure that all message and data formats can be passed successfully. Data structures in commercial software packages are tightly controlled as part of the intellectual property of the developer, further adding to the challenge. Network operations and maintenance in this tight integration scenario can also be troubling. Each application needing

communication requires a separate piece of software to be executed and maintained, adding to software and hardware cost, as seen in **Figure 2**. Each new release of software must undergo detailed evaluation to ensure that any changes to data structures are understood. For any changes identified, the software interfaces must be modified; the process of executing, testing, and troubleshooting issues slows the deployment of new software and adds cost for the enterprise. An enterprise may face the choice of waiting to roll out new software or spending thousands of dollars to update interface software to address changes in a new release [3].

The complexity of tightly integrated applications only increases as more applications are added. For example, turning to the business transaction discussed earlier, the enterprise may wish to introduce a third software application, such as an inventory system, into the existing purchasing and accounting systems. Integrating this application with both existing systems will require two additional software interfaces to be specified, developed, deployed, and maintained. A firm can quickly expend a great amount of both financial and human resources on developing and maintaining all of these software interfaces [4].

Fortunately for those enterprises seeking to use and leverage the model of integrated software applications, the development of the Internet and the World Wide Web has led to better solutions with the emergence of SOA. Using a concept known as *Web services*, SOA is a model for loose integration of disparate applications.

SOA EXPLAINED

SOA focuses on the perspective that software applications are services that support a particular business process. Fundamental to the concept of SOA are standards that determine how these services are built, maintained, and used in the network. By defining a standard and requiring adherence to it, SOA helps to address the issues outlined earlier in describing a tightly integrated network.

The SOA concept revolves primarily around the use of Web services to facilitate communications among applications. Web services generally use a language based on extensible markup language (XML) to both standardize data formats and exchange information. The language is referred to as Web services description language (WSDL). Both XML and WSDL are World Wide Web Consortium (W3C®) standards, allowing the SOA framework to use data formats put in place by others.

Using WSDL, system architects create a method for communications among applications that is fundamentally independent of the underlying platforms and programming languages. The method created is then advertised to the network as a “service,” a step referred to as “exposing” or “revealing” the application in an SOA framework [5].

Simple object access protocol (SOAP) is generally used by computers to implement SOA. SOAP specifies how a computer using hypertext transport protocol (HTTP) and XML can execute a program call to another computer; it also specifies how the program on the other computer

Since Web services are platform-independent and application-transparent, even a sweeping change, such as wholesale replacement of an application, can be done while remaining invisible to the users and the architecture.

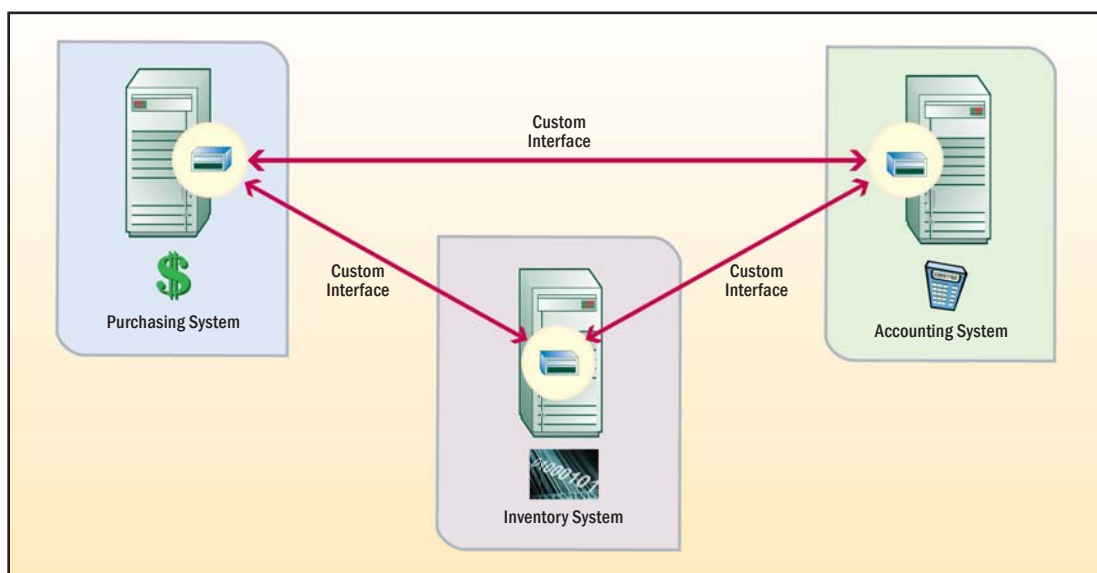


Figure 2. Integrating a New System Requires Further Custom Interface Development

One of the greatest challenges for service providers in launching a new application centers on supporting that application; specifically, how to provision, configure, manage, troubleshoot, and bill for that service.

can format, package, and transmit a response to the system calling it. HTTP and XML are present on all computing platforms that use the Web, regardless of manufacturer, and SOAP takes advantage of their presence. An added benefit is that interoperability in the presence of firewalls is further assured by using HTTP and XML. While a firewall may be configured to block certain types of traffic, based on port usage, protocols, or remote calls, almost all firewalls are configured to allow standard HTTP traffic. By using HTTP to transport XML, SOAP is most likely to successfully penetrate corporate firewalls and other perimeter security devices. While SOAP is the dominant methodology for communications over Web services, other techniques exist, including remote method invocation (RMI), using Java™; and inter-object request broker (ORB) object protocol (IOOP), a part of the Common Object Request Broker Architecture (CORBA®) [6].

Once SOA is deployed in an enterprise, developers need only maintain the applications that use the SOA, and not their method for communication. Again returning to the business example discussed earlier, if the purchasing system was to be updated, a developer would simply make sure that the updates used Web services properly for communication. Since Web services are platform-independent and application-transparent, even a sweeping change, such as wholesale replacement of an application, can be done while remaining invisible to the users and the architecture, as long as the new platform continues to communicate with Web services [6].

The fundamental concepts of SOA have existed for years, though described in different terms, such as “distributed architecture,” “modular programming,” and “event-oriented design.” Why did the SOA push prove successful, while

preceding concepts did not? With the rise of the Internet and the World Wide Web, almost every computer and operating system today ships with the ability to access the Web using standard protocols. Thus, the architects of the SOA concept are able to leverage and build on these standards to ensure interoperability among platforms, as seen in **Figure 3**.

Businesses can use WSDL to enter themselves into the Universal Description, Discovery, and Integration (UDDI) registry, a list of businesses worldwide that use Web services. The UDDI registry contains information on the business, as well as details on how data is implemented in the firm’s Web services. The registry streamlines the communications exchange among businesses, allowing applications from different businesses to interact and fostering e-commerce. Thus, SOA not only improves integration within a company’s network, it can also improve integration among any number of company networks [7].

SOA AND TELECOMMUNICATIONS

As outlined, SOA can fundamentally change how businesses integrate their applications and can streamline both enterprise communications and e-commerce. Can SOA also be used to improve the business of a telecommunications service provider? The business problems for service providers parallel those of large enterprises, and service providers can benefit from the lessons learned by these enterprises.

A large area of focus and expense for a service provider is the operation support system and billing support system (OSS/BSS). A typical OSS/BSS supports inventory management, provisioning, trouble tickets/resolution, lawful intercept and compliance, customer relationship management, order management, and network services. Various applications accomplish one or more of the many functions supported by the OSS/BSS, but there is no “one-size-fits-all,” single-supplier OSS/BSS implementation. Instead, organizations generally rely on a hodgepodge of vendor-provided and home-grown solutions, each performing part of a function, a single function, or several. Service providers also have to deal with legacy systems, some of which are built on foundations that are decades old. Integration of these systems is a great source of operational expenditures for carriers.

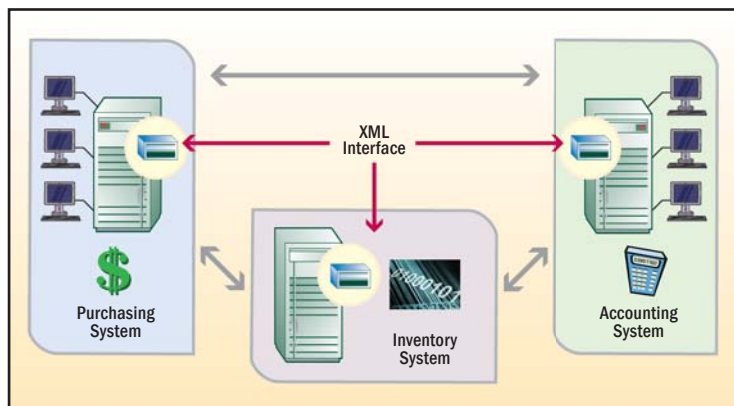


Figure 3. Web-Based XML or WSDL Ensures Interoperability Among Platforms

Significant opportunities exist for integrating the applications that make up the OSS/BSS. Most of the data, such as subscriber information or network infrastructure topology, is used by multiple applications. To date, primarily point-to-point, proprietary systems have been developed and used to integrate the communications among applications. These systems are fraught with the issues outlined previously.

Carriers can work to expose each piece of the OSS/BSS with Web services and to build an SOA that links the pieces together, as shown in **Figure 4**. By doing so, a service provider can both extend the life of an existing system—particularly in the case of a homegrown or closed, proprietary solution—and simplify the introduction of new systems and applications. Holistically, as the pieces of the OSS/BSS work better together, the subscriber experience should be improved—including faster time-to-service and quicker, easier problem resolution. Thus, a service provider can reduce customer churn by deploying Web services to unify and integrate its support systems.

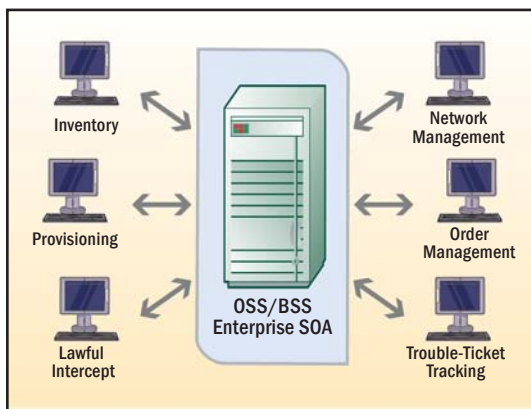


Figure 4. Using SOA To Integrate OSS/BSS Systems

One of the greatest challenges for service providers in launching a new application centers on supporting that application; specifically, how to provision, configure, manage, troubleshoot, and bill for that service. Without integration, each piece of the OSS/BSS must be separately addressed to ensure that the new service functions properly—often resulting in significant duplications of effort. With integration, applications can build on and leverage already defined frameworks from other parts of the OSS/BSS. Service providers can therefore reduce the time-to-market for new services by deploying an SOA framework [8].

Standards bodies worldwide are exploring and defining the benefits of using an SOA framework in network management. The European Telecommunications Standards Institute (ETSI) has released its architecture document TS 188 001, which outlines the architecture of a next-generation network’s operational support system. The International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) has produced Recommendation M.3060, “Principles for the Management of Next Generation Networks.” Both of these documents outline how a network management system functions and the benefits offered by SOA.

Consolidation is the norm in the telecommunications industry; the seven Baby Bells that resulted from the Modified Final Judgment in 1983 have consolidated into three today. Mergers and acquisitions have also dotted the landscape of cable system operators, wireless carriers, and competitive local exchange carriers. When a merger is executed, each firm brings a patchwork network of systems and operational functions that must be integrated, along with the customers and networks of the firms involved.

Before SOA existed, a newly formed entity would have to painstakingly inventory each system to understand its functionality, path forward, and interfaces with other business processes before deciding to either (a) build a custom front end to integrate the pieces, or (b) merge two platforms. Often, the same process was implemented multiple times within each pre-merger entity, further complicating the integration project.

With SOA, Web services can expose multiple processes, allowing for seamless interaction of data among them. As processes are consolidated, the Web services used to communicate among them continue to function, removing the underlying infrastructure from the user experience. Deploying SOA allows the newly formed firm to slowly and carefully integrate the pre-merger entity systems, easing transitions and removing the urgency felt while evaluating the path forward.

Verizon®, the nationwide local carrier formed by the merger of the regional carrier Bell Atlantic and the independent consortium GTE, chose the SOA route when integrating the two networks. Verizon identified 500 key business functions and built Web services around them. Known as the IT Workbench, Verizon’s SOA even goes as far as having developers “advertise” the applications they develop on a Web site, encouraging other applications to use them [9].

The extensibility of applications using Web services makes them ideal for integration with third parties; using Web services, a network provider can open its network to allow third parties to offer content and applications.

The SOA concept can go well beyond operations support for carriers; in fact, an SOA model can be used by a service provider in deploying new applications and services. The extensibility of applications using Web services makes them ideal for integration with third parties; using Web services, a network provider can open its network to allow third parties to offer content and applications.

Using SOA to leverage the network to provide new services turns the SOA model on its side; rather than just offering a way to save operational costs, SOA becomes a revenue generator for the operator. The SOA model also streamlines development time and costs for the third-party developer. Standard information valuable to the developer, such as the user's handset type, is now readily available, saving the developer the time of writing the code to discover this type of information.

T-Mobile® has taken this approach, sharing information about its network applications and resources via the SOA framework. Third parties using T-Mobile's network can streamline the development of their applications, making use of existing infrastructure and services wherever possible. T-Mobile provides external parties with a Web repository of deployed applications and services, and how the services are implemented. This approach allows T-Mobile to collect higher proportions of dollars in revenue-sharing schemes with the third parties because the third parties' costs to deploy applications are greatly reduced by the SOA [10].

As wireless and other service providers begin to deploy integrated applications using an Internet

Protocol (IP) multimedia subsystem (IMS), as illustrated in **Figure 5**, comparisons are being made between the IMS and SOA. Is an IMS an SOA? As outlined earlier, SOA is not a particular product or standard, but a framework for a type of architecture. Much like other types of SOAs, an IMS provides standardized, reusable services that are separate from the underlying network; however, the IMS adds the dimensions of call setup and control (via session initiation protocol [SIP]) and security (via IPSec). Security itself is not specified in the SOA framework; however, operators must consider security as a fundamental piece of their architecture [11].

Thus, an IMS can be viewed as a special case of SOA, designed in particular to support telecommunications applications and services. The IMS may also become just a portion of the SOA in an evolved network – another piece of the underlying architecture used to deploy services to end users.

CONCLUSIONS

Thanks to the innovations of the Internet and the World Wide Web, a new framework—SOA—has emerged that allows for the integration of applications residing on various platforms and networks and even within various firms. This framework holds great potential that is only beginning to be realized. Service providers can follow the example of enterprises in deploying SOA to reduce operational expenses and enhance the integration of business processes. In addition, service providers can use SOA principles to accelerate new service offerings and increase revenues. Service providers are just beginning to realize the benefits of SOA, and the future looks bright for this architecture in the field of telecommunications. ■

TRADEMARKS

CORBA is a registered trademark of Object Management Group, Inc., in the United States and/or other countries.

Java is a trademark of Sun Microsystems, Inc., in the United States and other countries.

T-Mobile is a federally registered trademark of Deutsche Telekom AG.

The **Verizon** name is a registered trademark of Verizon Trademark Services LLC or its affiliates in the United States and/or other countries.

Service providers can follow the example of enterprises in deploying SOA to reduce operational expenses and enhance the integration of business processes.

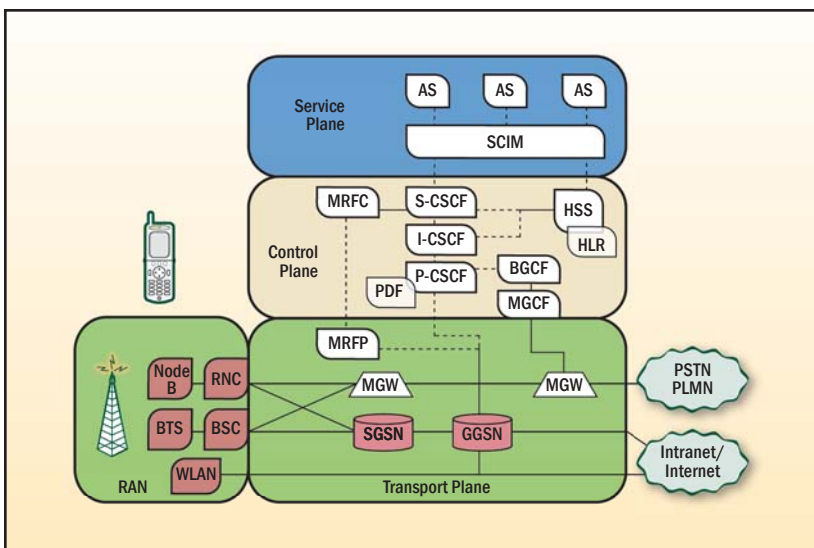


Figure 5. IMS Presents an SOA Framework

W3C is a registered trademark (registered in numerous countries) of the World Wide Web Consortium; marks of W3C are registered and held by its host institutions MIT, ERCIM, and Keio.

REFERENCES

- [1] "Service-Oriented Architecture (SOA)" (http://www-306.ibm.com/software/solutions/soa/overview.html?S_TACT=106AJ04W&S_CMP=campaign).
- [2] E. Pulier and H. Taylor, *Understanding Enterprise SOA*, Manning Publications Co., 2005.
- [3] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, 2005.
- [4] D. Linthicum, "Application Integration: Addressing the Issues" (<http://webservices.sys-con.com/read/44011.htm>).
- [5] "Will SOAP Wash Away Those Business Integration Issues?" (http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci883699,00.html).
- [6] "Publishing and Finding Web Services Using UDDI" (<http://edocs.bea.com/wls/docs92/webserv/uddi.html>).
- [7] A. Skonnard, "SOAP: The Simple Object Access Protocol," MSDN site (<http://www.microsoft.com/mind/0100/soap/soap.asp>).
- [8] "OSS/BSS Integration" (<http://www.capeclear.com/solutions/ossbss.shtml>).
- [9] L. Erlanger, "Verizon Goes Back to the Workbench," *InfoWorld* magazine (http://www.infoworld.com/article/05/11/07/45FEsoacaseverizon_1.html).
- [10] C. Koch, "How SOA Really Works," *CIO* magazine (http://www.cio.com/blog_view.html?CID=10591).
- [11] C. Boulton, "When SOAs and Telcos Collide," *Enterprise* magazine (<http://www.internetnews.com/ent-news/article.php/3606016>).

BIOGRAPHY



Brian Coombe joined Bechtel Telecommunications in 2003. Currently, as program manager of the Strategic Infrastructure Group, a pivotal unit of the Bechtel Federal Telecoms organization, Brian manages a program that involves telecommunications systems and critical infrastructure modeling, simulation, analysis, and testing. He evaluates government telecommunications markets, formulates requirements for telecommunications and water infrastructure work, and develops the Strategic Infrastructure Group's scope.

As Bechtel's technical lead for all radio frequency issues, Brian draws on his extensive knowledge of wireless and fiber optic networks. In his first position with the company, he engineered configurations to allow for capacity expansion of the AT&T Wireless GSM network in New York as part of a nationwide buildout contract. Later, he was the lead engineer for planning, designing, and documenting a fiber-to-the-premises network serving more than 20,000 homes. He is the Bechtel Telecommunications Laboratory's resident expert for optical network planning, evaluation, and modeling.

Before joining Bechtel, Brian was a systems engineer at Tellabs®, where he launched the company's dense wavelength-division multiplexing services and managed network design and testing. He developed solutions to complex network issues involving echo cancellation, optical networking, Ethernet, TCP/IP, transmission, and routing applications.

Brian is currently completing work toward an MS in Telecommunications Systems Engineering at the University of Maryland. He earned a BS with honors in Electrical Engineering at The Pennsylvania State University.

Brian is a member of the Institute of Electrical and Electronics Engineers; INSA; AFCEA; and Eta Kappa Nu, the national electrical engineering honor society. He has published technical articles in the *Bechtel Telecommunications Technical Journal*, and his tutorial on Micro-Electrometrical Systems and Optical Networking was presented by the International Engineering Consortium.

