Service-Oriented Architecture and the C4ISR Framework

Dr. Yun-Tung Lau
Science Applications International Corporation

This article presents an architecture modeling approach for formulating service-oriented architectures such as those being developed on the global information grid. The approach uses object-oriented techniques to supplement the traditional Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance Framework.

The global information grid (GIG) is the globally interconnected, secured, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, policy makers, and support personnel [1]. The GIG supports all U.S. Department of Defense (DoD), national security, and related intelligence community missions and functions. It provides capabilities from all operating locations and interfaces to coalition, allied, and non-DoD users and systems.

The GIG as a transformational vision aims at achieving information superiority in a network-centric environment. It enables various systems to interoperate with each other. For the warfighters, it brings *power to the edge* through a Task, Post, Process, Use process. For the business and intelligence communities, it provides the infrastructure for effective information gathering and collaborative operation.

This transformation from a centralized, sequential thinking and a static one-to-one interfacing paradigm to a distributed, parallel information sharing and dynamic collaboration approach requires a fundamental shift in the way systems are built. Specifically, it lends itself to a service-oriented architecture (SOA) on a ubiquitous network carrying information on demand.

In an SOA, a set of loosely coupled services works together seamlessly and securely over a network to provide functionalities to end users. These services have well-defined interface contracts. Supported by service management tools at the enterprise level, they are published, discovered, mediated, and consumed in an orderly fashion.

The service-oriented approach is inherently dynamic. It allows fast formation of expedient communities of interest (COI) to handle highly volatile situations and changing mission requirements. It also supports the stable operation of longstanding or institutional COIs. The SOAs are flexible because each service encapsulates the underlying platforms and technologies that support it. The services provided at the enterprise level are therefore agnostic to those specific platforms and technologies.

The Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework [2], along with its three standard views and common products, has been widely used in building C4ISR systems. The framework was originally developed in 1996 by the DoD to provide guidance for describing architectures. Version 2 was officially mandated in 1998 as the DoD Architectural Framework and is being superceded by the DoD Architecture Framework (DoDAF). Standard techniques employed by C4ISR include point-to-point interfacing, static connectivity, and data-flow analysis. These are more suited to the traditional sequential processing, system-oriented, and one-to-one integration paradigm.

"In an SOA [service-oriented architecture], a set of loosely coupled services works together seamlessly and securely over a network to provide functionalities to end users. These services have well-defined interface contracts."

With the ongoing efforts to transform the DoD into a network-centric, service-oriented environment, the following questions are often raised:

- Does the C4ISR framework apply to such a service-oriented architecture?
- How is it supplemented with other techniques to fully describe such architectures?
- What are the set of products that describe the essence of a service-oriented architecture?

Whereas full answers to these questions will require extensive discussion, this article describes a pragmatic approach that naturally fits the service-oriented environment. This approach utilizes object-oriented design and analysis techniques to supplement the standard C4ISR framework for developing SOAs. As the DoD moves toward a network-centric environment supported by SOAs, this approach provides a timely and rigorous methodology for specifying future enterprise architectures, and has been recently applied to the architecture development of Net-Centric Enterprise Services (NCES) with satisfactory results.

The NCES is a collaborative environment that supports vertical/horizontal interoperability between DoD business and warfighting domains, as well as the national intelligence domain [3]. The NCES provides the core enterprise services that support various standing and expedient COIs on the GIG.

Using this approach, the use cases for the NCES core enterprise services were developed. The corresponding operational and systems views were constructed as part of an integrated architecture product. Activities in the use cases were also mapped to the Net-Centric Operations and Warfare Reference Model [4].

In the following sections, I first describe the approach for formulating an SOA. Using an enterprise messaging system as an example, I then discuss the corresponding architecture views that embody the approach in the C4ISR framework[1].

Formulating an SOA

In formulating an SOA, you start with operation. Here the focus is how end

Architecture Views and Products

The Department of Defense (DoD) Architecture Framework (AF) (which will supercede the Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance Architecture Framework) provides the guiding principles for modeling and designing architectures in the DoD environment (Version One is available at <www.aitcnet.org/dodfw>). Architecture is described by three views:

- The Operational View (OV): Describes the tasks, activities, operational elements, and information exchanges required to accomplish missions.
- The Systems View (SV): Describes systems and interconnections supporting operational functions.
- The Technical View (TV): Includes technical standards, implementation conventions, rules, and criteria that guide systems implementation.

Each view has a set of products. Some are listed in the table below (the product numbers do not imply the order of developing them). In addition to those listed, there are 12 products and two *All Views* products omitted for brevity.

Views	Products and Descriptions
Operational	OV-1 (high-level operational concept graphic)
	OV-2 (operational node connectivity description)
	OV-3 (operational information exchange matrix)
	OV-5 (operational activities model)
	OV-6c (operational event-trace description)
Systems	SV-1 (systems interface description)
	SV-2 (systems communications description)
	SV-4 (systems functionality description)
	SV-5 (operational activity to systems function traceability matrix)
	SV-6 (system data exchange matrix)
	SV-11 (physical schema)
Technical	TV-1 (technical standards profile)

users, systems, or applications use services. Use cases in Unified Modeling Language (UML) [5] describe the external behavior of a service as seen or utilized by an actor (user, system, or application). The boundary of the service in a use case is clearly delineated. The interaction of the actor with the service is described without revealing the internal details of the service. Use case is, therefore, a natural tool for describing operational activities in an SOA.

Figure 1: *Formulating an SOA*

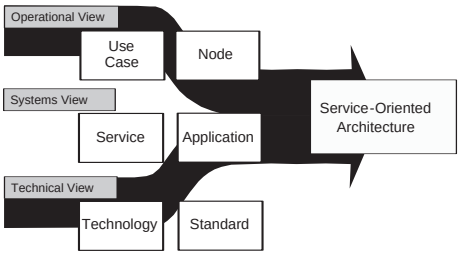
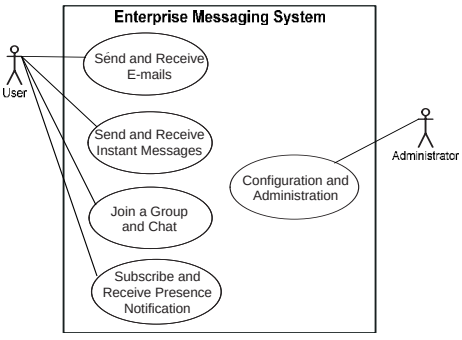


Figure 2: *Use Cases as Part of the Activity Model (OV-5) for the Enterprise Messaging System*



Based on the operational concept, the scope of services, and the high-level requirements, one may identify a set of high-level critical use cases. These are the use cases that the architecture must support to meet the minimal requirements. Use cases are not requirements. Nevertheless, they illustrate what functions architecture provides and highlight the requirements. Therefore, use cases are the first step in formulating an SOA (see Figure 1).

In each use case, typically two or more nodes interact with each other by exchanging information. If a node is a service consumer, then it is an actor in the use case for that service. If it is a provider, then it is a component providing that service. Traditionally, a node in the C4ISR framework represents a role, an organization, an operational facility, etc. For an SOA, its scope is expanded to include shared resources and services. Hence a service node interacts with consumer nodes to provide services. Use case and node are, therefore, the primary objects in describing the operational aspects of an SOA, as shown in Figure 1.

Once the operational aspects are identified, the next action is to find the solution that satisfies the operational requirements. In an SOA, each service provides a set of well-defined functions useful to its users, or consumers. An example is chat service, which allows users to perform online chat. A set of services may interact in an orderly manner to provide a complete set of mission functions. In this way they form a mission application, or simply application. Application is not just a simple collection of services, but an integral set of logically connected services. Application and service form the primary objects that describe the systems aspects of an SOA. They are, in practice, the software that needs to be built by developers.

Finally, standards and technologies are the primary objects that constitute the technical foundation in implementing an SOA. Figure 1 summarizes the approach for formulating an SOA, along with the primary objects. Discussed next are the corresponding architecture views that embody the approach in the C4ISR framework[1].

Operational View

For a concrete example, let us consider an enterprise messaging system, which encompasses e-mail, instant messaging (IM), chat, and presence services. The critical use cases are send and receive e-mails and instant messages, participate in a chat session, subscribe to and receive presence notifications, etc. They are shown in the use case diagram in Figure 2. In addition, the administrator configures and administers the services.

For each use case, you may describe a sequence of events or activities. These activities may be presented in a hierarchy, as in the standard activity model operational view (OV-5). Here, however, the use cases provide a natural grouping of those related activities. Additionally, a use case highlights the actors and system/service boundary, allowing you to delineate roles and nodes easily. Hence, include use case as part of OV-5 and consider it an essential product for an SOA.

For an SOA, the use-case diagrams (such as Figure 2) often identify the nodes. These nodes are roles, organizations, shared resources, or service nodes. You can further draw the connections (i.e., the need lines) between the nodes, thereby forming the operational node connectivity description (OV-2). An example is given in Figure 3. Except for the use of UML deployment diagram

notations, this figure is the same as the standard OV-2.

Finally, a description of each connection in Figure 3 gives the operational information exchange matrix (OV-3). Each row in the matrix describes the provider and consumer nodes, the information exchange, the mode of exchange (e.g., synchronous), the security aspect, etc. This again is the same as the standard OV-3.

The high-level operational concept graphics (OV-1) still applies to an SOA. This, together with OV-5, OV-2, and OV-3, encompasses the concepts of operation, the use cases from user's viewpoint, the connectivity between operational nodes, and their information exchanges. They therefore characterize the essential operational aspects of an SOA. Furthermore, since operational nodes include shared resources and services, dynamic and collaborative operational activities are properly captured.

To analyze more details in the use cases, you may use the Integration Definition for Function Modeling process diagrams [6] to depict the activities (including inputs, controls, outputs, and mechanisms). This will also be part of OV-5. Alternatively, you can use the UML sequence diagrams (OV-6c) to describe the details.

Systems View

As discussed earlier, application and service are the primary objects in Systems View (SV). In an SOA, one is more concerned with the logical interaction between service providers and consumers. Rather than relying on static connections, users in an SOA may select different services under different use cases. Consequently, system interface description (SV-1) in the form of logical architecture diagrams is usually appropriate. Logical architecture diagrams show the connectivity between service provider nodes (or components) and consumer nodes. They also specify the types of interface or communication protocols.

For efficient software management, closely related use cases utilizing similar services are usually grouped together and supported by an application. Thus you may draw a functional decomposition diagram as systems functionality description (SV-4). The diagram will identify the applications. Each application supports one or more use cases and may have a corresponding logical architecture diagram as SV-1.

Going back to the sample enterprise messaging system, the basic services are e-mail, IM, chat, and presence services. E-mail service is a familiar form of asynchronous messaging and may be provided through either a thick or thin client. The other three services emphasize synchronous interaction and therefore are best provided through a single application to the users. The corresponding systems functionality description (SV-4) is shown in Figure 4.

The SV-1 picture for the e-mail service is shown in Figure 5. Here again, notations similar to the UML deployment diagrams are used. The boxes represent nodes that are connected by channels of data exchange. A service node has a well-defined service interface (indicated by a protruded match head) and supports data exchange in certain protocols. In Figure 5, the protocols and interfaces are HyperText Markup Language (HTML), HyperText Transfer Protocol (Secured) (HTTPS), Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP), and the mail access application programming interface called Post Office Protocol v.3 (POP3).

Services are often organized into layers, with the lowest layer containing core services, and the upper layers containing value-added and composite services. Services in the upper layers use those in the lower ones to perform specific functions. You may use service layer diagrams such as SV-1 to show the dependencies of such service stacks. An example for the enterprise messaging system is shown in Figure 6 (see page 14), which includes the synchronous messaging services and storage and security core services. Note that a service consumer (such as IM and Chat Client) may dynamically connect to one of many chat servers that provide chat service. In this sense, the connectivity is not static.

There are situations such as in security service or network management service in which a system communications description (SV-2) is more suitable than SV-1. This is because such services are naturally associated with physical systems and network elements.

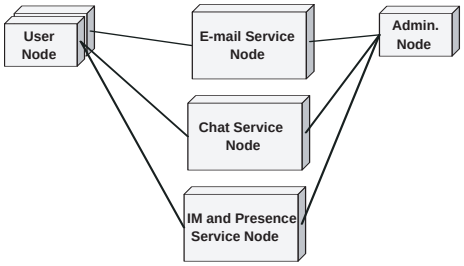


Figure 3: *Operational Node Connectivity Description (OV-2) for the Enterprise Messaging System*

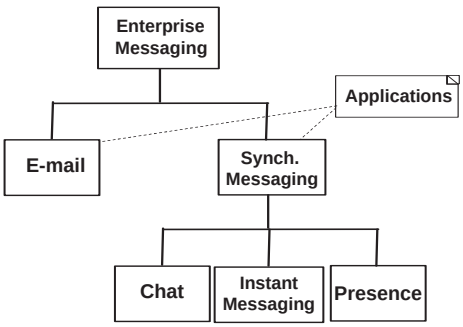
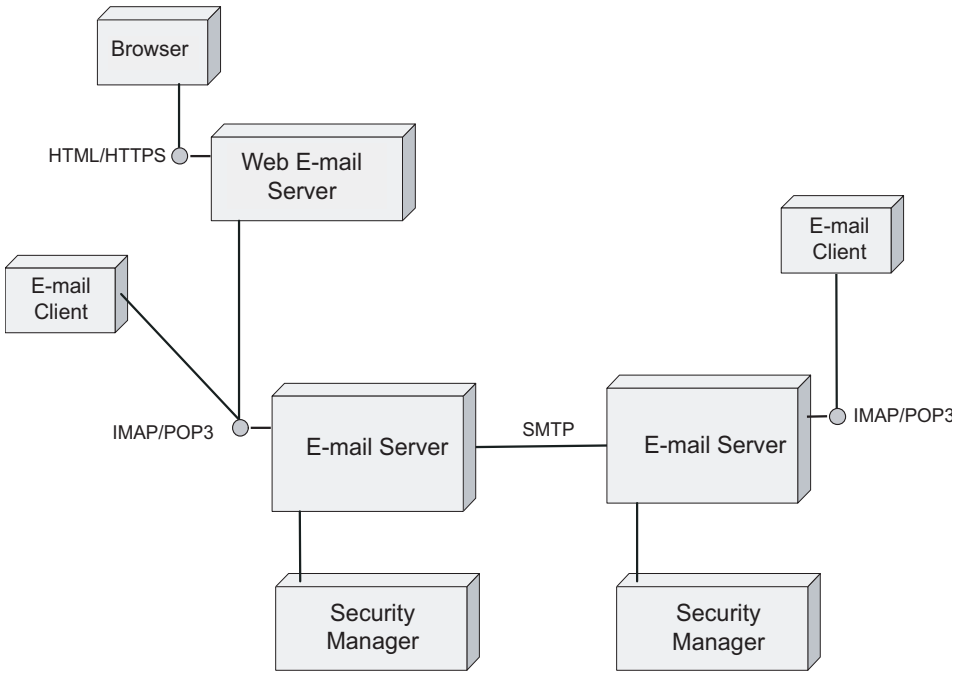


Figure 4: *Systems Functionality Description (SV-4) for the Enterprise Messaging System*

Figure 5: *Logical Architecture Diagram (SV-1) for the E-mail Service in the Enterprise Messaging System*



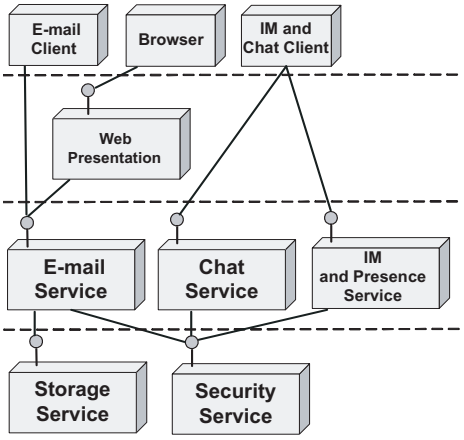


Figure 6: *The Enterprise Messaging System Represented as a Service Stack (SV-1)*

For an SOA, SV-4 and SV-1 (or SV-2), capture the functional breakdown and the logical or physical structures that support those functions.

Traditionally, system data exchange matrix (SV-6) provides detailed data exchange information between system nodes. Such a matrix depicts static data exchange connections. In contrast, data exchange in an SOA is specified by service contracts and a list of consumers of the services. Services can be dynamically published through a service broker. Consumers may then dynamically discover, subscribe, and consume the services. Hence service contracts play the role of SV-6 and the service broker facilitates connection between consumers and providers.

For instance, the emerging Web service paradigm uses Web Services Description Language to describe service contracts. Simple Object Access Protocol is used as the transport mechanism. And Universal Description, Discovery, and Integration may be implemented as part of the service broker.

In addition to SV-6, other SVs may also capture other supplementary properties of a service. For example, physical schemas (SV-11) may be used to describe data schemas in a service contract, and system performance parameters for quality of service or service level agreement. The operational activity to systems function traceability matrix (SV-5), on the other hand, shows how the applications satisfy the requirement by supporting the use cases.

Technical View

The Technical View (TV) in an SOA is the same as traditional C4ISR architectures, with standards and technologies as key elements. Here technical standards profile (TV-1) is essential because it references the key technical standards and technologies employed by the SOA. The Joint Technical Architecture [7] provides primary references for these standards and technologies.

Summary

Using UML techniques to supplement the traditional C4ISR framework, I have elucidated an approach for formulating an SOA. On the operational side, it starts with use cases, which involve the interaction of two or more operational or service nodes. Mission functions are provided through applications, which are implemented by a set of services. The corresponding C4ISR architecture products are also discussed along with an example.

In the appendices[1], I also present the complete UML model for architectural products in an SOA and its mapping to the Federal Enterprise AF. It provides a solid modeling foundation for the above approach.

When applied to NCES, this approach was very effective. The use cases in the nine core enterprise services of NCES drove the development of the architectural products. They also provided a natural link to the NCES requirements and direct connection to the end users. After developing the high-level architecture products, detailed events for the use cases were analyzed, service interfaces or contracts were defined, and metrics for service performance were established.

Some topics for future investigation on this approach include how to capture SOA products in a Core Architecture Data Model database, which is included in the DoDAF; how to specify and manage service contracts in an SOA to ensure interoperability across the enterprise; and how to evaluate compatibility or compliance between different SOAs.◆

References

1. U.S. Joint Forces Command. Global Information Grid Capstone Requirements Document. JROCM 134-01. Norfolk, VA: USJFCOM, Aug. 2001 <https://jdl.jwfc.jfcom.mil>.
2. C4ISR Architecture Working Group. C4ISR Architecture Framework Vers. 2.0. Washington, D.C.: Department of Defense, 18 Dec. 1997 <www.fas.org/irp/program/core/fw.pdf>. The next version is the DoD Architecture Framework Vers. 1.0. 15 Aug. 2003 <www. aitcnet.org/dodfw>.
3. Global Information Grid Enterprise Services. Initial Capabilities Document for Global Information Grid Enterprise Services. Arlington, VA: GIG ES, 9 Sept. 2003 <http://ges.dod.mil>.
4. Net-Centric Operations and Warfare Reference Model, Draft Vers. 1.0. 20 Oct. 2003 <https://disain.disa.mil/ncow. html>.
5. Object Management Group, Inc. Unified Modeling Language (UML), Vers. 1.5. Needham, MA: OMG, Mar. 2003 <www.omg.org/technology/documents/formal/uml.htm>.
6. National Institute of Standards and Technology. Integration Definition for Function Modeling (IDEF0). Federal Information Processing Standards Publication 183. Gaithersburg, MD: NIST, Dec. 1993.
7. JTA Development Group. Joint Technical Architecture Vers. 5.1. Washington, D.C.: U.S. Department of Defense, 12 Sept. 2003 <www.jtaonline.disa.mil>.

Notes

1. Additional details on this and other developments can be found in this article's online version at <www.stsc.hill.af.mil/crosstalk>. In the PDF version, click on the appendices link.

About the Author

Yun-Tung Lau, Ph.D. is assistant vice president of Technology at Science Applications International Corporation. He has been involved in large-scale software architecture, design, and development for 14 years. Lau has served as chief architect for many software and enterprise architecture projects, from scientific computing, to electronic commerce, to command and control systems. He has published many articles in professional journals and has written "The Art of Objects: Object-Oriented Design and Architecture." Originally trained as a theoretical physicist at Massachusetts Institute of Technology, Lau also has a Master of Technology Management.

Science Applications International Corporation
5107 Leesburg Pike STE 2000
McLean, VA 22041
Phone: (703) 824-5817
Fax: (703) 824-5836
E-mail: yun-tung.lau@saic.com