A Unified Service Description for the Global Information Grid

Yun-Tung Lau, Ph.D.
Science Applications International Corporation

This article presents a unified approach to service description for enterprise services on the Global Information Grid (GIG). The approach introduces the concept of service module. It also identifies the links between various standards and frameworks of service description through mappings of metadata. These linkages provide end-to-end traceability for enterprise services across the architecture and design levels, thereby facilitating the development of service-oriented architectures (SOAs).

As a Department of Defense (DoD) transformational initiative, the GIG will provide a set of globally interconnected, secure end-to-end information capabilities to support operational missions conducted by various communities of interest (COIs) in the warfighting, business, and intelligence mission areas [1]. These capabilities will be fulfilled by GIG enterprise services, which are self-contained, stateless functions with well-defined interfaces that allow discovery and use of the services [2]. Such enterprise services resemble subroutines or functions in traditional computer programming except that they can be invoked by other computer programs over a network, and they are typically at a higher (mission operation) level.

SOAs are promising architecture paradigms for building GIG enterprise services. In an SOA, a set of loosely coupled services works together seamlessly and securely over a network to provide functionalities to end users [3]. As shown in Figure 1, the service provider registers information about a service interface at a service registry (step 1 in Figure 1). Service consumers can find the service from the registry (step 2) and then invoke the service through the service interface (step 3).

A typical SOA has many service consumers and service providers. The service registry may consist of a federation of registries or repositories across an enterprise. An example of an SOA on the GIG is Net-Centric Enterprise Services (NCES) [4], which provide a set of core enterprise services, including security service, service discovery, machine-to-machine messaging, and mediation for data transformation. Other applications and services on the GIG can utilize these general purpose core services to perform common functions. For COIs, enterprise services may be developed within an SOA. For example, in the command and control area, services such as blue (friendly) force location and target management services can be part of the upcoming Net-Enabled Command Capability (NECC) SOA [5].

A service description describes the way a service consumer interacts with the service provider, including the format of the request/response (messages), preconditions and post conditions, security information, quality of service (QoS) levels, etc. Some of this information is packaged into machine-readable interface contracts (e.g. Web Service Definition Language [WSDL] files). Others are entered into service registries for discovery (e.g. a Universal Description, Discovery, and Integration [UDDI] registry). Consequently, service descriptions play a central role in an SOA. They are key assets of an enterprise and should be part of the shared knowledge in the enterprise.

However, as various industry and DoD standards and frameworks of service description emerge over time, each framework tends to address a specific need without linking itself to the overall SOA engineering life cycle. This article identifies the links between existing service description standards and DoD frameworks, thereby establishing an end-to-end picture of a service and its role in an enterprise.

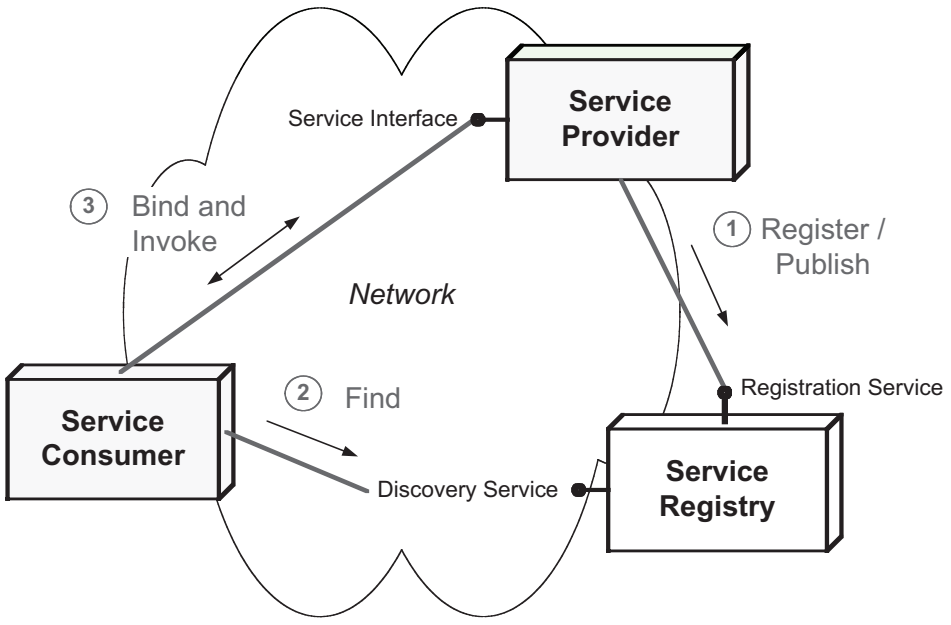
The following sections describe the artifacts for service definition and develop the relationships and mappings among them. I also provide a complete object model for an overall service description.

Service Definition Artifacts

Because of the central role played by service descriptions in an SOA, they are needed practically in all phases throughout an SOA engineering life cycle. At the enterprise architecture level, the DoD Architecture Framework (DoDAF) [6] is used for programs of record across the DoD. DoDAF provides the guiding principles for modeling and designing architectures in the following 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

Figure 1: *Basic Interactions in a Service-Oriented Architecture*



Artifacts	Representative Usages
DoDAF SV-6	Portfolio management of services. Enterprise architecture and high-level system design.
SST	Documentation of services for the GIG.
WSDL	Definition of services readable by Web service engines.
UDDI Registry	Discovery of service and provider information.
ebXML Registry	Life-cycle management and discovery of services.

Table 1: *Artifacts for Service Definition*

guide systems implementation.

Each view has a set of products. Among the SV products, the Systems Data Exchange Matrix (SV-6) specifies the characteristics of data exchange between systems. The characteristics are captured in tabular form and include data description, producer and consumer, performance attributes, security information, etc. For SOAs, similar characteristics can describe data exchange between service consumers and service providers. One may therefore apply the SV-6 product to service descriptions at the architecture level. In this case, the producer in the SV-6 matrix represents the service provider and the consumer represents the service consumer.

At the design and implementation level, the Service Specification Template (SST) has been proposed as part of the GIG Net-Centric Implementation Document series [7]. The SST identifies a set of elements (grouped by categories and subcategories) that describe a GIG enterprise service. These elements indicate what the service does, how to access the service, the security mechanisms or restrictions for the service, relevant performance information, etc. The SST is intended to aid in the specification, implementation, documentation, and discovery of services across the GIG.

For implementation and deployment, several industry standards are widely used: WSDL for machine-readable interface contracts [8] and UDDI [9] and Electronic Business eXtensible Markup Language (ebXML) registries [10] for discovery of services. They contain different aspects of information about the services. Table 1 gives a summary of the above artifacts for service definition.

Unified Service Definition

These artifacts were developed separately for the uses shown in Table 1. To gain a deeper understanding of the relative roles they play in building an SOA, one must establish an end-to-end linkage across them. We can achieve this in two steps. First, we introduce the concept of a service module in order to facilitate the transition from architecture to design of an SOA. Second, we identify the mappings of metadata between the artifacts.

Service Module

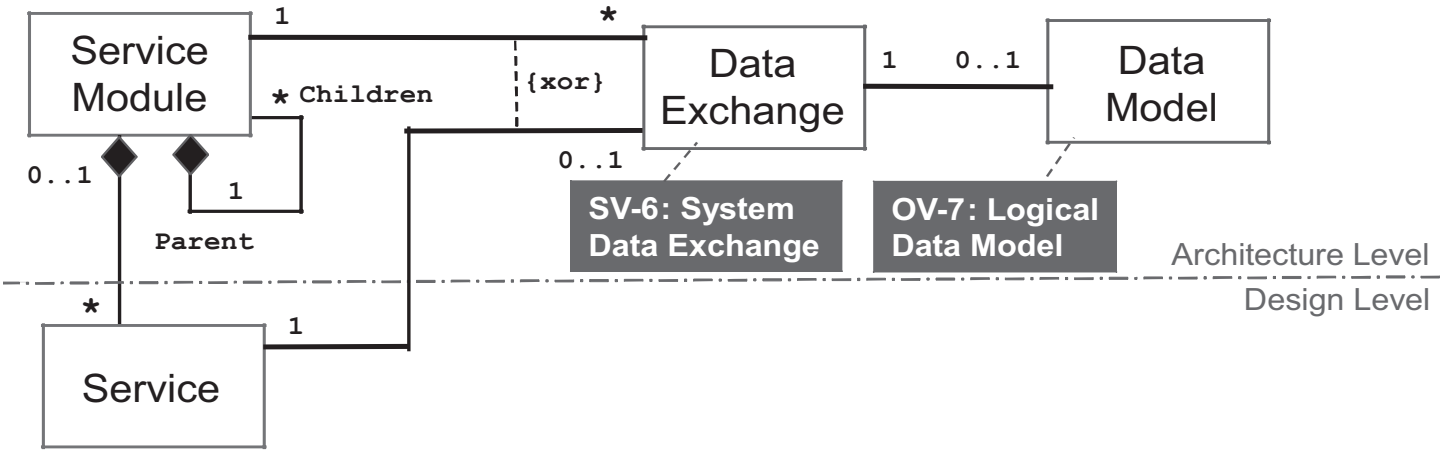
A key activity in the early stage of SOA development is identifying the services. These services, identified at the enterprise architecture level, are often (though not always) *service modules*, which handle operational processes in a certain mission area. Each service module may contain multiple concrete services which are implemented in the design and development phases and invoked by consumers after deployment.

As an example, a task management service module may contain two concrete services: retrieval and administration. The task retrieval service is consumed by general users assigned to perform the tasks. The task administration service, on the other hand, is used by administrative users who set up and maintain the tasks. An enterprise architecture artifact, such as the SV-6, may capture information about the task management service module only, or it may also contain information about the two concrete services.

Figure 2 gives the relationship of service and service module in Unified Modeling Language (UML) notations [11]. It shows that a service module may contain multiple child modules, as indicated by the asterisk next to the label *Children*. A service module may be related to multiple rows in the SV-6 matrix (each row corresponding to a data exchange object in Figure 2). These rows represent data exchanges of concrete services under that service module. Alternatively, one may roll up the information from the concrete services under a module to a single row in the SV-6 matrix. Also, the notation 0..1 in Figure 2 indicates zero or one instance of an object, whereas 1 means exactly one instance. For example, a service may be associated with zero or one data exchange, whereas a data exchange is always associated with one service (under SOA).

Depending on the level of details conveyed by an enterprise architecture, one may provide data exchange information at the concrete service level. In this case, a row in the SV-6 matrix contains information about an individual concrete service. The label {xor} in Figure 2 indicates that a row in SV-6 may be associated with either a concrete service or a service module, but not both. In what follows, concrete services at the design level are simply

Figure 2: *Relationship of Service and Service Module*



called services.

Figure 2 also shows that a data exchange (a row in the SV-6 matrix) may be associated with zero or one logical data model (OV-7). Identifying data models at the architecture level helps promote sharing of data across services, which is a key tenet of SOAs.

Mapping of Metadata

Service definition is about information that describes a service. In other words, it contains metadata about a service. One can group those metadata into categories, such as security information, service level information, etc. Different artifacts for service definition focus on different categories of metadata. By mapping the metadata across the artifacts, one establishes the linkages between the artifacts.

Table 2 gives the mappings of metadata across the artifacts for service definition. An empty cell indicates that there are no corresponding metadata for that artifact. For example, SV-6 does not carry version information, which is needed for design and implementation. Note that if an entry in the SV-6 represents a service module, then there is no corresponding mapping to the other artifacts. This is because those other artifacts are below the architecture level.

In the SV-6 matrix, the parent-child relationship can be indicated by a dot-delimited System Interface Identifier in the form of x.y.z…, where x, y, z are integers. For example, the following shows a Security Service Module and a partial list of services under it:

- 1.7 Security Service Module.
- 1.7.1 Certificate Validation Service.
- 1.7.2 Policy Decision Service.
- 1.7.3 Policy Retrieval Service.

Artifacts at the design level usually do not carry information on such a parent-child relationship.

Table 2 shows, other than architecture level information, the SST provides rather comprehensive information about a service. The information needed for invoking a service is mapped to WSDL, whereas the information for discovery of services is mapped to a UDDI or ebXML registry.

UDDI uses tModel (which basically contains name-value pairs) to facilitate searching by attribute values. The mapping strategy in this case is to link the elements in SST to a UDDI tModel. For example, an *InformationSecurityMarking* element under the Service Information/Security category maps to an *InformationSecurityMarking* tModel. For an ebXML registry, *ClassificationScheme* and *ClassificationNode* are the equivalent of a tModel. One may therefore construct the mapping similarly.

	DoDAF SV-6	SST	WSDL	UDDI Registry	ebXML Registry
Service Name	√	√	√	√	√
Version		√	√	√	√
Parent/Child Relation	√				
Transaction Information	√				
Data Standard	√				
Namespace		√	√	√ [1]	√ [2]
Operations		√	√	√ [1]	√ [2]
Access Point		√	√	√ [1]	√ [2]
Security Information	√	√		√ [1]	√ [2]
Service Level Information	√	√		√ [1]	√ [2]
Schedule Information		√		√ [1]	√ [2]
Contact Information		√		√ [1]	√ [2]

Note 1: Each of these categories corresponds to a UDDI tModel, which basically contains name-value pairs.
Note 2: Each of these categories corresponds to an ebXML ClassificationScheme.

Table 2: *Mapping of Metadata by Category*

Detailed UML Model and Mapping

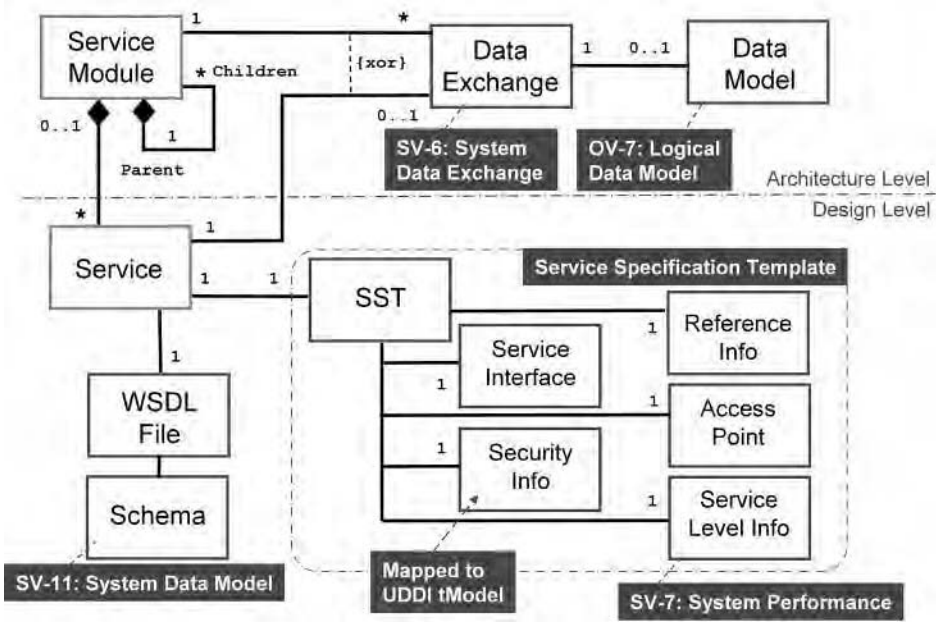
The material in this section is intended for SOA practitioners who would like to find out details of the mappings described in this article. They may further use the Web examples[3] as references for building service definitions in an SOA.

Figure 3 presents a full UML model for Service Module and Service. In addition to the relationship given in Figure 2, it shows the linkages from Service to SST, WSDL, and other related artifacts at the design level. Here the UML notation 0..1 indicates zero or one instance of an object. An asterisk represents zero or more instances, whereas 1 means exactly one instance. The label {xor} indicates that a Data Exchange object (a row in SV-6) may be associated with either a concrete service or a service module, but not both.

The mappings at the field or XML element level between these artifacts are given in the spreadsheet USD_Mapping _and_Example.xls[3].

For the SST, an earlier version (v. 2.0) of the document defines an XML schema, which is called the Service Definition Framework (SDF). The sample XML data is based on that SDF schema and is in the

Figure 3: *UML Model for Service Module and Service*



file CES_Security_CVS(SDF). They may be useful as references for building service definitions.

Finally, in the spreadsheet, we use the following XPath notations in identifying elements in XML data for the mappings:

1. /A/B/C: Element C under element B, which is under the root element A.
2. D/E[@x]: Attribute x of element E under element D.

   For example, the XPath expression

/SDF/ServiceAccessPointInformation/ ServiceAccessPoint/operation

corresponds to "getStatus" in the XML data below:

```
<SDF>
   ...
   <ServiceAccessPointInformation>
        ...
        <ServiceAccessPoint>
            <operation>getStatus
             </operation>
            <binding>SOAP/HTTP
             </binding>
            <port>http://decc2.dod.
             mil/CES/Security/CVS
             </port>
            <POCIndex>Jane Smith
             </POCIndex>
            <SupplementalInformation>
             OCONUS
              </Supplemental
             Information>
        </ServiceAccessPoint>
   </ServiceAccessPointInformation>
</SDF>
```

Similarly, the XPath expression

/definitions/binding/operation[@name]

points to the operation name "getStatus" in the XML data below:

```
<definitions>
  ...
  <binding name="CertificateValidation
   ServiceSOAPBinding" ... >
   <soap:binding style="document"trans
    port="http://schemas.xmlsoap.org/
    soap/http"/>
   <operation name="getStatus">
   ...
    </operation>
  </binding>
</definitions>
```

Closing Remarks

The introduction of service module enables a unified approach for service definition across the architecture and design levels. When performing a top-down SOA design, one may start with one or more services in a module and later refine them into more services. The service module remains the same during this refinement, therefore allowing the design to evolve without affecting artifacts at the architecture level.

On the other hand, in a bottom-up approach, one can map design level information in the SST to WSDL, UDDI, and ebXML, as shown in Table 2. As one refines the individual services, one may further group related services into service module at the architecture level.

The unified service description thus lends flexibility to the system engineering process and provides end-to-end traceability for enterprise services in the GIG. Even though one may use different tools for the different standards and frameworks in Table 2, those tools can in principle be integrated or linked together to provide a complete picture of the services.◆

References[1]

1. "Global Information Grid Capstone Requirements Document." JROCM 134-01. U.S. Joint Forces Command, 2001 <https://jdl.jwfc.jfcom.mil>.
2. United States. Assistant Secretary of Defense, Networks and Information Integration. "Net-Centric Operations and Warfare (NCOW) Reference Model." v1.1. Washington: DoD, 2005 <https://standmgt.disa.mil/restricted/ncow.html>.
3. Erl, Thomas. "Service-Oriented Architecture (SOA): Concepts, Technology, and Design." Prentice Hall, 2005.
4. United States. Defense Information Systems Agency (DISA). Initial Capabilities Document for Global Information Grid Enterprise Services." DISA, 2004 <https://gesportal.dod.mil>.
5. "Net-Enabled Command Capability Milestone A Acquisition Decision Memorandum, Assistant Secretary of Defense, Networks and Information Integration." Washington: DoD, 2006 <https://gesportal.dod.mil/sites/necc>.
6. United States. Assistant Secretary of Defense. "DoD Architecture Framework." Ver. 1. Washington: DoD, 2007.
7. United States. Assistant Secretary of Defense. "Global Information Grid Net-Centric Implementation Document: Service Specification Template (S300)." Ver. 4.0. Washington: DoD, 2006 <https://gesportal.dod.mil>[2].
8. Christensen, Erik, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. "Web Services Description Language." 1.1 World Wide Web Consortium, 2001 <www.w3.org/TR/wsdl>.
9. "Universal Description, Discovery, and Integration (UDDI)." v3.0.2. Organization for the Advancement of Structured Information Standards, 2004 <http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf>.
10. ebXML. Organization for the Advancement of Structured Information Standards <www.ebxml.org/specs>.
11. International Organization for Standardization/International Electrotechnical Commission "Unified Modeling Language." 19501. International Organization for Standardization. <www.iso.org>.

Notes

1. Some Web sites quoted here require a user account for access. Online application forms can be found on the sites. Some require government sponsorship.
2. An XML schema was included in version 2.0 of this document, titled Service Definition Framework. The sample XML are based on that schema.
3. The example files are available for download in the online version of this article.

About the Author

Yun-Tung Lau, Ph.D. is Vice President of Technology at Science Applications International Corporation (SAIC). He has been involved in large-scale software architecture, design, and development for 18 years. Lau has served as chief architect for many software and enterprise architecture projects, from scientific computing and electronic commerce to command and control systems. He holds a master of technology management. Lau has published many articles in professional journals, and authored the book *The Art of Objects: Object-Oriented Design and Architecture.*

SAIC
5113 Leesburg Pike
STE 200
McLean, VA 22041
Phone: (703) 824-5817
Fax: (703) 824-5836
E-mail: yun-tung.lau@saic.com