Service Oriented Architectures Using DoDAF[1]

Huei-Wan Ang, Fatma Dandashi, Michael McFarren
The Mitre Corporation

The MITRE Corp.
7515 Colshire Dr.
McLean, VA 22102
hwang(at)mitre.org, dandashi(at)mitre.org, mmcfarren(at)mitre.org

This paper describes how the Department of Defense Architecture Framework (DoDAF) can be used for describing a Service-Oriented Architecture (SOA). With acceptance of Network-Centric Warfare (NCW) and "net-centricity", the Department of Defense (DoD) is moving towards operational and systems designs based on dynamic producer/consumer models consistent with Service Oriented Architecture (SOA). Service Oriented Architecture (SOA) is an architectural approach to application integration that enables flexible connectivity of applications or resources implemented as services. Such services have well-defined, platform-independent interfaces that hide the underlying technical complexity of the environment (encapsulation), are self-contained (loosely coupled), and reusable. DoDAF does not prescribe any particular methodology or process (SOA or otherwise) for creating the actual architecture model, but only the elements and relationships that any given methodology would use. Tailoring of the DoDAF elements and relationships, which DoDAF allows, is necessary to meet the SOA paradigm shift. Further, creating DoDAF architecture descriptions that are service orientated supports globalization and the integration of geographically dispersed organizations (Net-centricity).

1 Introduction

A Service Oriented Architecture (SOA) approach is one where application design and development is based on the concept of services. The principal objective of the DoD Architecture Framework (DoDAF) [1] is to ensure that architecture descriptions can be compared and related across organizational boundaries, by defining a particular set of architectural elements and relationships used for describing architectures. This difference allows DoDAF to effectively describe a SOA. However, some tailoring is required to better support SOA design patterns within DoDAF. In this paper, we describe an approach to reducing software systems complexity through the use of

---

[1] This work was partially supported by the US Space Situation Awareness Integration Office and the AF Warfighting Integration and Chief Information Office.

reusable and composable services[2] and describing the resulting SOA with tailored DoDAF. Our premise is that using DoDAF to describe a SOA enables leveraging the existing body of knowledge and architecture artifacts within DOD. Further, tailoring DoDAF for SOA enables architects to more effectively describe SOA as an alignment of services (in SV) to operational activities (in OV) and to identify common functionality as a set of re-usable services. The last section shows sample tailored SV-1, SV-4, and SV-10c products for an example scenario that is described in section 3.2.

2 What is a Service?

The term *service* has been defined by several industry and standards consortia such as IBM, OASIS, etc., with slightly different variations. In general, the current body of knowledge is consistent in applying the term to a certain kind of **software** application with the following characteristics. A service provides well defined, self-contained functionality that is loosely coupled from other functionality/services. The functionality is well encapsulated (i.e., complexity of the implementation is hidden from potential consumers except for the information required by the consumers to determine whether a given service is appropriate for their needs; that information is exposed in the service interface [2]). The semantics of a service should be documented, either directly or indirectly, by its description (also referred to as a service's standard interface(s) or set of messages [3]). Services have course granularity, they tend to use a small number of operations with relatively large and complex messages which are exchanged between the provider and consumers. A service is location transparent (i.e., consumers do not need to be aware of the physical location of a hosting server); protocol independent (messages are sent in a platform-neutral, standardized format delivered through the interface); and stateless (i.e., the service remains in the same state after each execution request from a consumer). Services tend to be oriented toward use over a network, though this is not an absolute requirement. [3]

3 What is SOA?

SOA is a form of distributed systems architecture based on services (as defined above) where a consumer does not need to know the internal structure of a provider, including features such as its implementation language, process structure, and even database structure [3]. In SOA, the focus is on the sequence of operational activities or business process. The operational process is then mapped to a systems architecture description with specific applications that support the operational process cast as services. Thus, the architecture supports a business process via a set of independent, reusable, but collaborative services. The service integration happens dynamically, via service composition (the execution of several of these independent

[2] While the scope of this paper is software services, the DoDAF tailored products and underlying metamodel are equally applicable to human functions (e.g., as in outsourcing business services)

services in an orchestrated manner). SOA is an "architectural discipline that centers on the notion that IT assets are described and exposed as Services. These Services can then be composed in a loosely-coupled fashion into higher-level business processes,…" [4]

3.1 Advantages of SOA

SOA offers several advantages. 1) An operational process orchestrates simple services (owned by the process) into complex services. 2) Operational performance can be improved through closed feedback loops from runtime behavior back to the operational process. 3) Services allow the exchange of information and data between: a) different computers, from different vendors, b) different programs, from different functional areas, or different members of a Community of Interest (CoI), and 4) SOA supports globalization and the integration of geographically dispersed organizations (net centricity) through service orchestration of distributed services owned and executed across ownership boundaries. [5]

3.2 Space Weather Impact Analysis Sample Scenario

Throughout section 4, some tailored DoDAF products supporting the following scenario are provided as an example. Space weather information from the Space Weather Service is required to provide Space Situation Awareness (SSA) for Space Command and Control (C2). Space weather information is also required to support theater operation planning and commercial satellite launch planning. A Space C2 Operator uses the Space C2 Application to view a User Defined Operational Picture (UDOP). In providing the UDOP display, the Space C2 Application uses the previously defined UDOP criteria to request the Space UDOP from the Space UDOP Service, which in turn sends a request to the Space Weather Service to provide a space weather impact report for a specified period of time. The Space Weather Service queries the Weather DataBase (DB). The WeatherDB returns the queried space weather impact report present at that moment in the database. If the space weather information requested is not in the database, the Space Weather Service sends a request to the Space Weather Impact Analyzer to make and provide a space weather impact prediction. The Space Weather Service returns a space weather impact report as the response to the Space UDOP Service, which integrates the report with all other SSA information and returns the requested Space UDOP to the Space C2 application. The Space C2 application displays the Space UDOP. The example architecture provided here was developed using UML 2.0[3]. Due to paper length limits, only three key products are described to illustrate the tailoring. The example provided does not represent a complete architecture model and each example product does not necessarily include the entire potential content of a product, but includes enough detail to properly demonstrate the tailoring of a product. In addition, the content of the example is based on information provided by the Space Situation Awareness Integration Office (SSAIO) but does not represent actual or planned operational processes or systems from Space Command.

[3] The example was developed using IBM/Rational's Software Architect

4 SOA in DoDAF

All Views (AV): There are some overarching aspects of an architecture description that relate to all three views. These overarching aspects are captured in the AV products. Two products are defined in the All-View section, Overview and Summary Information (AV-1) and Integrated Dictionary (AV-2). These two products do not need to be tailored for SOA.

Operational View (OV): OV describes the capabilities, operational activities, and information exchanges required to conduct operations. No tailoring of the Operational View (OV) is required to support SOA. However, the OV is important to SOA because it shows the operational or mission justification for application investment, and identifies specific operational activities that can be automated via services in a SOA. One of the promises of SOA is that it allows for better alignment of applications with operational processes. In order to accomplish this alignment, SOA requires that the operational processes be both well defined, and defined at a granular enough level to be able to map directly to the services in the SV. One of the tenets of DoDAF is to rigorously define the OV products to better enable this alignment.

Systems View: The SV consists of a set of graphical and textual products that describe human functions and machine processes (services) and their interconnections in support of DoD operational activities. The relationship between architecture data elements across the SVs to the OVs can be exemplified as follows: humans and automated systems, or human functions and services, are grouped into nodes and fielded to fulfill OV requirements and execute operational activities. Some tailoring of the SV is required to support SOA.

SV-1 Systems Interface Description: SV-1 shows where services are desired by describing logical or physical nodes that group services and human functions. Services and human functions may be logically grouped by where they are advertised (e.g., all services published in a particular registry) or by some broad functional category (warfighting, logistics, financial management, HR, medical, etc.); alternatively, services and human functions can be physically grouped, for example all services and/or and human functions that reside on an aircraft carrier. SV-1 also shows logical communication channels between the nodes, systems, or humans. Figure 1 is an SV-1 that shows systems nodes (the outermost box, including those denoted as *External*), systems, servers and/or applications that provide or consume certain services. Inter-nodal communication channels shown here reflect the fact that the services interact across nodes. A provided service is denoted by a part with a port that binds to a defined interface (the service specification) from SV-4. The provided services shown grouped into nodes are the same ones that are defined in SV-4 diagrams. A consumer port denotes that the consumer (system or application) requires the use of a service through its defined interface. A consumer port is shown on SV-1 applications or services to indicate they consume services provided by others. To reduce SV-1 clutter, not all consumer ports are shown on Figure 1.

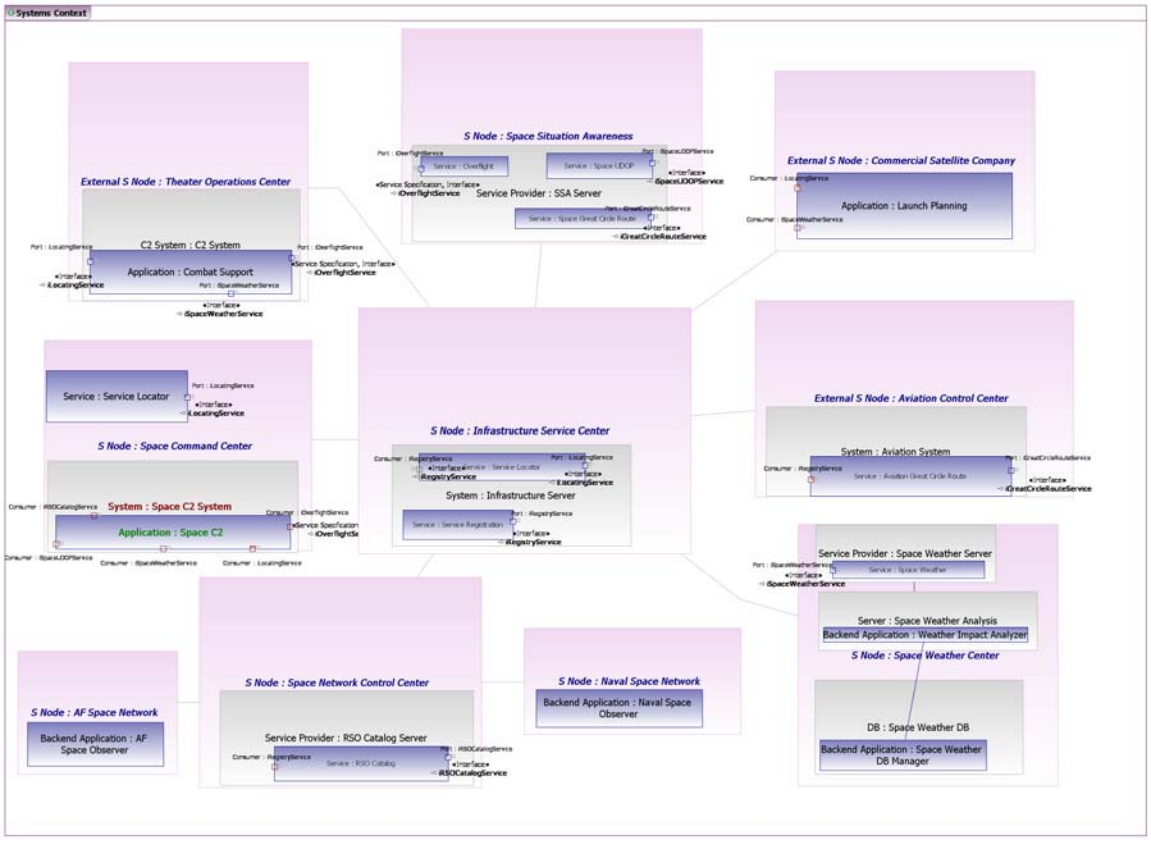

Figure 1: SV-1 Showing Groupings Of Services By Nodes

SV-4 Systems Functionality Description: The Systems Functionality Description documents specifications of human functions, services, and their known product (data or material) flows, human and machine interactions. While SV-1 shows providers and consumers of services and how they might be structurally designed, SV-4 for SOA can be tailored to show the service interface (or the service description/specification), the service realization, and dependencies as demonstrated in Figure 2. Service interfaces in SV-4 (denoted in UML using the Circle notation) are the same ones shown as ports in SV-1 through which consumers utilize the service (the interface is the exposed part of the service). The service realization (this is the black box that is not exposed to consumers) is denoted in SV-4 using UML class notation. Operations defined on the service class (not shown in Figure 2, but show up as activities in SV-10c) realize the functionality described in the service interface. Service consumers (not always services themselves, e.g., the Space Launch Planning Application) are also shown in both SV-1 and SV-4.

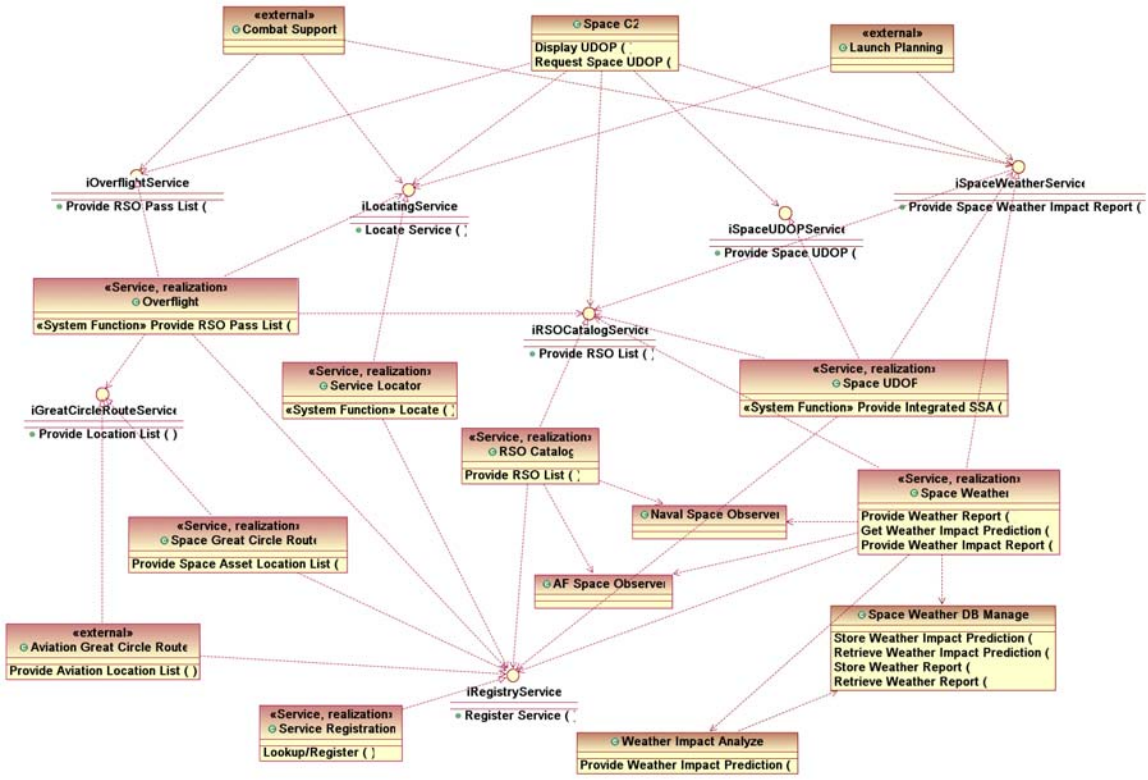

Figure 2: SV-4 Showing Service Dependencies

Systems Functionality Sequence and Timing Descriptions – (SV-10): The SV-10 generally mirrors the OV-6, and describes the dynamic behavior of humans and systems, and of human functions and services. The Systems Event-Trace Description (SV-10c) describes the sequence (control) flows for human functions and services. With SOA, these service sequence flows can be used to specify service interactions, or they can be used to specify the sequence of flows between human functions, application services, and infrastructure services. Figure 3 is a UML Activity diagram (like a flowchart with swimlanes) that details the process flow among interacting services. The swimlanes correspond to the UML classes that were defined in Figure 2 and which define the service realizations. Activities appearing in the swimlanes correspond to operations (System Functions) defined on these classes. Data flows and data stores can also be modeled on an Activity diagram, and Figure 3 illustrates two of those (not all flows have been modeled in this example product to reduce diagram clutter).

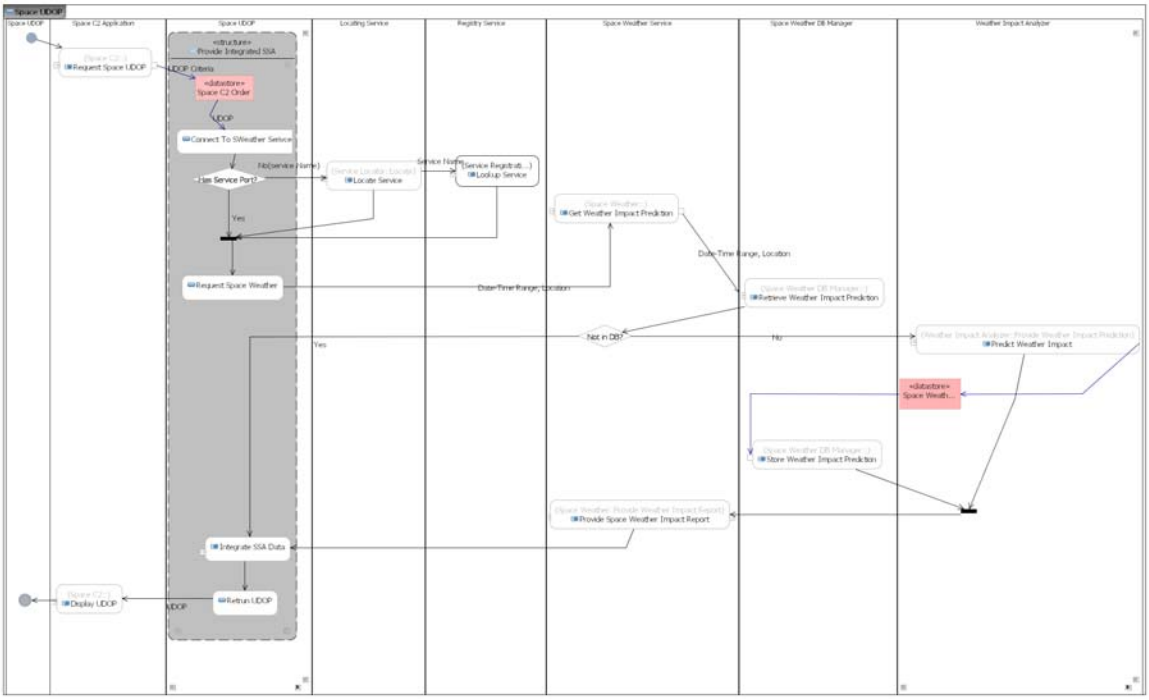

Figure 3: SV-10c Showing An Activity Diagram

Figure 4 demonstrates a particular scenario of the process flow illustrated by Figure 3. It includes the planner (Space C2 Operator) who sends a request for space UDOP to the SSA server. The messages that are exchanged between the various entities correspond to the service interface operations that were illustrated in Figure 2.

The SV products provided in this paper collectively demonstrate how a service-oriented systems architecture may be described with limited tailoring to DODAF SV products. The example provided here was developed using UML because this modeling language defines constructs that are semantically rich enough to describe a SOA using DODAF.

The Technical View provides the technical implementation standards upon which engineering specifications are based, and common building blocks are established. The Technical Standards Profile collects the various standards rules that implement and sometimes constrain the choices that can be made in the design and implementation of an architecture description.

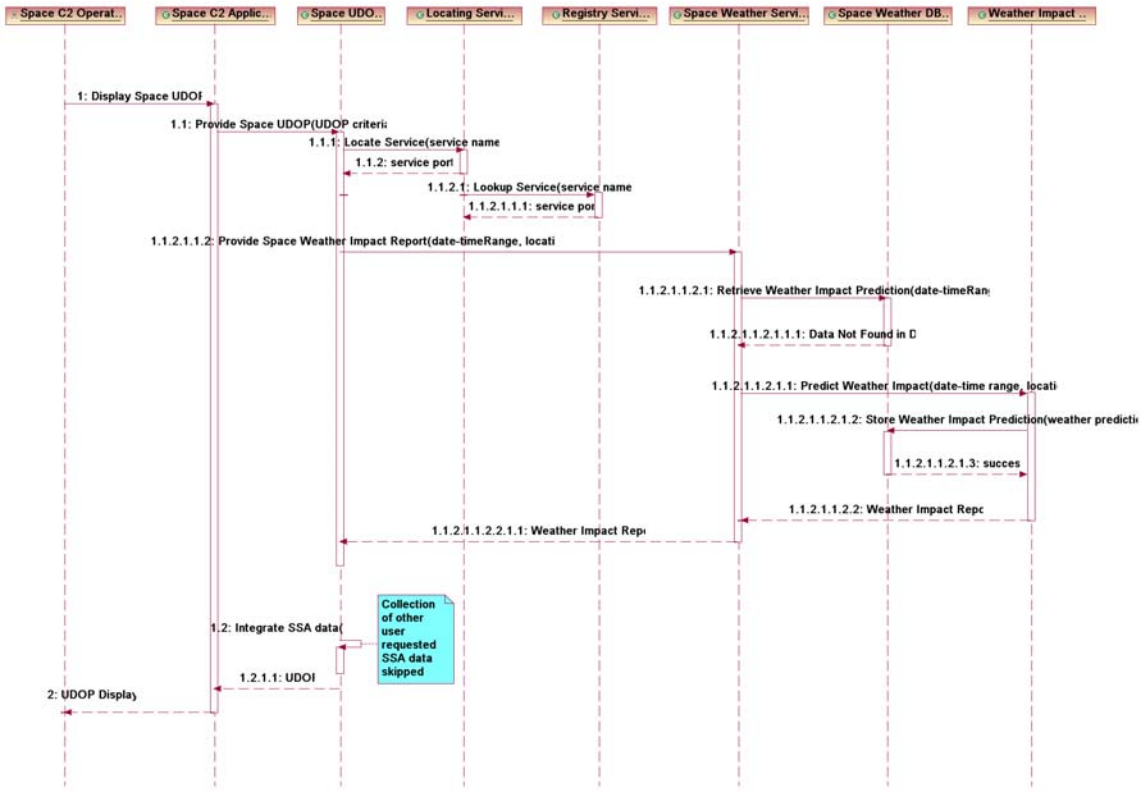

Figure 4: SV-10c Showing A Scenario with A Sequence Diagram

The Technical View does not require tailoring. It is used to capture standards that are utilized (or are to be utilized) in specifying service interfaces/descriptions (e.g., WSDL, OWL), service messaging protocols, service implementation languages, etc.

Bibliography

[1] Department of Defense Architecture Framework V1.0

[2] Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/home/index.php

[3] World Wide Web Consortium, http://www.w3.org/

[4] http://www.zapthink.com/

[5] "Patterns: Service-Oriented Architectures and Web Services," http://www.redbooks.ibm.com/abstracts/sg246303.html?Open