SoaML and UPIA Model Integration for Secure Distributed SOA Clouds

R. William Maule
Information Sciences Department
Naval Postgraduate School
Monterey, CA, USA
rwmaule@nps.edu

***Abstract*—The U.S. Department of Defense (DoD) specifies the Department of Defense Architecture Framework (DoDAF) for military architecture models. Industry typically uses different modeling tools, including the Unified Modeling Language. There are many modeling techniques for Service Oriented Architecture (SOA). DoDAF supports SOA through DoDAF version 2.0. Industry supports SOA through several toolsets including UML Profile-based Integrated Architecture (UPIA) and the SOA modeling language (SoaML). This paper presents background information on SOA modeling techniques and some methods to transition between DoD and commercial tools. Variables and metrics for analysis in SOA models are presented. UPIA is advanced as a means to transition between DoDAF 2.0 and SoaML. SoaML is rendered within UPIA models to help address potential future DoD initiatives for implementation of SOA within cloud architectures. This paper advances methods for industry-to-DoDAF model integration for secure distributed SOA clouds and provides background on defense and industry modeling techniques for enterprise-class SOA.**

Keywords-SOA; modeling; services; DoDAF, UPIA, UML, SoaML, SOA-RM, SOA-RA

I. INTRODUCTION

Architecture development for secure distributed SOA clouds presents unique challenges, especially when communications may be intermittent with significant outages, the content within each cloud node changes very rapidly, the nodes must be federated, and the content synchronized in real-time. This paper does not attempt to solve this complex and multi-faceted problem but rather presents work on a preliminary architecture and model integration methodology that may be part of a future solution.

SOA presents the potential to be an all-encompassing architecture for next-generation information management. The extremely large number of components active in a distributed enterprise SOA often means that models cannot be directly implemented in computer code. The Department of Defense Architecture Framework (DoDAF) version 2.0 provides a framework to enact operational workflows, integrated systems, and ubiquitous services for a comprehensive SOA. UML Profile-based Integrated Architecture (UPIA), together with SOA Modeling Language (SoaML), provide a framework for the integration of components and services, with provisions for code generation and real-time monitor. Together these frameworks and their supporting tools and methodologies provide a solution for modeling secure distributed SOA. The following discussion provides an overview of the major modeling frameworks for enterprise SOA, followed by specific examples pertinent to Secure Distributed SOA clouds. The discussion begins with background on the major methodologies.

II. OASIS SOA-RM

In the Department of Navy (DoN), the OASIS SOA Reference Model (SOA-RM) is being advanced as a conceptual reference for desired SOA capabilities and practices [1]. SOA-RM provides a reference model for a SOA "market" and supporting services [2]. Models describe the type of transactions that must take place for SOA services to be "traded" with minimum risk. Also described is the administrative structure required to support necessary transactions. OASIS SOA Reference Architecture (SOA-RA) extends this research with a focus on operational models as a basis for systems engineering [3]. SoaML provides SOA-specific tooling and a means to instantiate OASIS-RM [4, 5].

III. DODAF 1.0

Department of Defense Architectural Framework (DoDAF) is used extensively throughout the DOD as the basis for modeling and systems engineering. DoDAF uses the IEEE 1471 definition of an architecture description to define a standard approach to describing, presenting, and integrating a DoD architecture that can be used with a service oriented approach for capability-based planning [6]. DoDAF version 1.5 "views" contain information about systems from a particular perspective–which are referred to as "products". Products are classified according to their specific architectural attributes to include: All Views (AV), Operational Views (OV), Systems Views (SV), and Technical Standards Views (TV). Certain aspects of a system's architecture may be best described by all three views. Relationships between architectural elements of the operational enterprise often require multiple views within each category, increasing in detail with each view.

In expressing these views the core elements of the models are nodes, needlines, services, and information exchanges. Nodes can be physical or logical, including both systems and humans. Needlines show relationships

978-0-7695-4756-5/12 $26.00 © 2012 IEEE
DOI 10.1109/SERVICES.2012.27

and dependencies between nodes to help characterize the system-of-systems relationships which fulfill missions. Services represent operational functionality, with information exchanges documenting the data types that traverse that functionality. This service element is expanded in DoDAF 2.0 to include service architecture.

IV. DoDAF 2.0

DoDAF-2 adds additional perspectives to assist in the transition to service architecture: Capability Viewpoints (CV) to articulate capability requirements and deployed capabilities; Data and Information Viewpoints (DIV) to articulate data relationships and alignment structures; Project Viewpoints (PV) to describe relationships between operational and capability requirements and detail dependencies among capability and operational requirements and services design; and Services Viewpoints (SvcV) to present design solutions with Performers, Activities, Services, and their Exchanges [7].

Researchers have additionally proposed enhancements to DoDAF to help sequence modeling products [8], validate consistency in those products [9], enhance event specification capabilities [10] , and introduce simulation-based testing [11]. Integrations with Activity Based Methodology (ABM), System Modeling Language (SysML), and Business Process Modeling Notation (BPMN) have been achieved [12]. The research herein extends this research to include integrated UPIA, SoaML and DoDAF models for the development of federated SOA clouds.

V. UPIA

Unified Profile for DoDAF and MODAF (UPDM) is an integration methodology for DoDAF and MODAF models and modeling techniques [13]. UML Profile-based Integrated Architecture (UPIA) extends UPDM to provide a standard approach for modeling enterprise architectures to support both DoDAF 2.0 and the UK Ministry of Defence Architecture Framework (MODAF). UPIA integrates with Model-Driven Software Development (MDSD) methodology to enable the generation of operational code from the models [14].

UPIA models use specialized elements to depict enterprise and system of systems architectures. The UPIA SOA Design viewpoint can be employed to design high-level structures for service-oriented architectures and include service specifications, service ports, and service consumers and providers at both operational and system levels of abstraction. UPIA is DoDAF 2.0 XML-complaint and can import and export DoDAF 2.0 Physical Exchange Specification (PES) architectural data. Most of the concepts defined in the DoDAF 2.0 meta-model (DM2) can be modeled in UPIA. UPIA includes utilities for model validation. Models advanced in this paper will implement UPIA to bridge UML and DoDAF and therein provide a framework for a comprehensive architecture for distributed SOA-based cloud nodes. Presented is a means for distributed SOA cloud modeling, from high-level operational and systems models to low-level code generation.

VI. SoaML

SoaML is an open source specification project from the Object Management Group (OMG) to describe a UML profile and metamodel for services within a service-oriented architecture. Existing models and metamodels (e.g., TOGAF [15]) for describing system architectures were considered insufficient to describe SOA in a precise and standardized way. UML was too general and needed clarification and standardization specific to SOA. Additionally, a means was required to operationalize SOA as advanced in the OASIS Reference Model (OASIS-RM). SoaML was adopted as a means to provide SOA-specific tooling and to instantiate OASIS-RM.

Studies have found that SoaML adds value to UML for large-scale SOA deployment [16]. Extensions have been added to provide support for multi-agent systems [17], mobile applications [18], service requirement variability [19], security-aware invocations [20, 21], and pattern propagation between design and performance models [22]. While designed to bridge IT and business models, SoaML is somewhat vague regarding implementation methodology to other business-level languages, such as Business Process Modeling Notation (BPMN) [23] and is an area addressed in the research herein.

Models in the following sections continue to extend UML methodology for SOA through model integration examples. SoaML is modeled through UPIA to address anticipated future DoN requirements for secure distributed SOA clouds. UPIA is the means presented in this paper to associate SoaML models to DoDAF models. The author has taken liberties in the application of SoaML and UPIA models to DoDAF 2.0 in order to model test systems and services active in the SOA Evaluation Architecture (SEA).

VII. SEA Model Components

SEA is a laboratory and field architecture to baseline performance variables in SOA components and identify issues in node-to-node services. The lab environment for the tests herein consists of two (2) physical locations geographically separated with communication through the Internet, wireless, and multiple intervening networks.

Within these two (2) physical locations are six (6) host servers supporting thirty (30) virtual machines – with SOA components on those machines. Implementation includes three (3) different virtualization approaches, and several different SOA configurations from multiple vendors including BEA, Oracle, IBM and open source. Models to be realized through SoaML will align with SEA test capabilities, generically modeled in Figure 1.

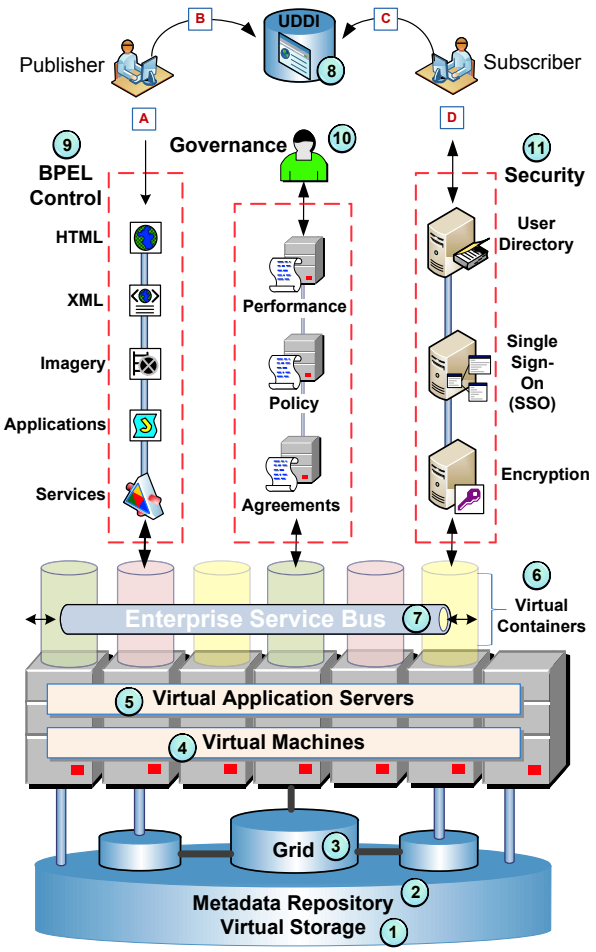

Figure 1. SEA operational framework model components.

At the storage tier reside the databases and metadata repository for the SOA. Grid computing techniques are addressed at the database layer for service management and node synchronization. Virtual machines host the J2EE application servers, which in turn host the primary SOA components. The components and their modeling variables are detailed in Table I.

Modeling variables in the storage tier depend on the intended use of the services – small and optimized for special functions at one extreme, or large and capable of backbone and archival operations at the other extreme. There are modeling variables associated with media selection (iSCSI, Fibre Channel, etc.), and there are associated attributes for throughput, latency and capacity.

The metadata repository, which is basically a database with XML schema, inherits the characteristics of a traditional database. All of the traditional metrics for database analysis are applied to the metadata repository, plus metrics specific to access and virtualization.

TABLE I. SEA FRAMEWORK MODEL VARIABLES

#	Component	Model Variables
1	Storage	Virtualization, and virtual interfaces
2	Repository	Databases, clusters, virtualization
3	Grid	Database and application management, process and agent options
4	Virtualization	Virtual machine management, systems management, performance options
5	Servers	Java Virtual Machine performance, J2EE and .NET configuration options
6	Containers	Interoperability variables, deployment processes, Quality of Service options
7	Service Bus	QoS, interoperability, throughput, latency; deployment options
8	Registry	QoS, object permissions, throughput, latency, service performance
9	Orchestrate	Performance of process monitors and procedure calls; latency, throughput
10	Governance	Service agreements, policy enactment; latency, throughput
11	Security	Authentication and authorization latency and throughput, encryption

Grid computing, in the context of current commercial service offerings, tends to focus on either the database and storage tiers, or on the application and process tiers. In SEA, a grid agent remains resident on the host and gathers information from specified processes.

Virtualization can and does occur throughout the SOA, but in Figure 1 is presented as number 4 at the level of the physical host. SEA includes VMware, Xen and Hyper-V virtualization technologies. Fortunately, many of the more comprehensive modeling environments allow the selection of specific virtualization boundaries within their frameworks. Comprehensive virtualization models address virtual machine creation, cloning, failover, distributed deployment, and template creation.

The application server is the middle tier host for services and applications. The application server and its virtual machine instances comprise what is typically referred to as “middleware” (Table II) and are mutually supporting. If middleware tiers in a SOA experience performance problems then services will start to fail.

Servers in the SEA environment use Java 2 Enterprise Edition (J2EE) to support currently tested services. In addition to basic server functionality, server performance variables model Java Virtual Machines (JVMs) and J2EE containers–with metrics for throughput of SOA processes.

TABLE II. SOA MIDDLE TIER MODEL VARIABLES

Component	Model Attributes
Server	Instance that runs in its own Java Virtual Machine (JVM) and has its own configuration
Cluster	Multiple servers run simultaneously to increase performance and scalability
Virtual Hosts	Set of host names to which server instances respond. Host names map to the IP address
Machine	Logical representation of the computer that hosts one or more instances (servers)
Work Managers	Request classes and thread constraints: J2EE modules, EJBs, RMI provide metrics
Startup / Shutdown	Java classes/programs which provide system-wide services for applications

Java "containers" work within the application server. They are interfaced with hardware; applications or services in Java are deployed to the server/container where they execute and are accessed by users. This process is critical to any service architecture model and the functions of the components become variables for models.

Containers are the primary means of service deployment and containers provide metrics for the processes they support. These metrics serve as attributes within the models.

The Enterprise Service Bus (ESB), as the primary application-layer router in the SOA, is at the core of a SOA model. Distributed architectures need to model ESB interfaces since interoperability and throughput (latency) for ESB exchanges will be a significant issue, especially in the DoD where security imposes overhead, and in DoN where there are communication challenges.

Model variables for the Registry include service invocation, and metadata management processes, with metrics to address transport Quality of Service (QoS). Process flows can be assigned to each step of the registration process, addressing: WSDL services that have been published to the UDDI, reference information for service providers, endpoints and interfaces through which programs connect to services, policies associated with the Registry or individual services, and XML translations.

Orchestration components are modeled through service interfaces, coordination processes, and management capabilities. These include governance interfaces [such as Business Process Management (BPM)] and tools to build and control processes and web service interactions, such as Business Process Execution Language (BPEL). Attributes to be modeled include interface functions of the BPEL, capabilities of the Governance system, web service policy rules and enforcement processes, and BPEL procedure calls to remote services.

Grid agents can be modeled as end-to-end performance monitors for components on distributed nodes. Governance adds to this functionality with support for security policies, service agreements, and runtime governance of BPEL flows. Additional variables in Governance include J2EE container interfaces, JVM representations, Java Messaging Service (JMS) linkages, and connectors to Business Activity Monitoring (BAM) systems. Potentially, output from Governance software can appear on BAM dashboards, in which case dashboard variables would include interfaces to analytic and decision support components. Specific components and attributes to be modeled would include:

Repository and Registry Variables – These variables give visibility into assets and relationships:

- *Collection*—Processes through which the system can automatically populate both the Repository and the Registry with existing services and support new asset creation through synchronization with developer tools
- *Categorization*—Processes through which the system organizes and advertises services and procedures
- *Management*—Processes to discover, map and persist new dependencies and support service impact analysis

Lifecycle Variables – Automation of repeatable processes for registration, life cycle management, with metrics for:

- *Workflows*—Based on standards and best practices
- *Policies*—To ascertain trust in service quality, including QoS, performance, and security
- *Standards*—to validate architecture compliance

Analytic Variables – To track and report on reuse throughout the entire service lifecycle:

- *Reuse*—Statistics from development through runtime, consumption and compliance
- *Compliance*—Performance, quality, standards
- *Return-on-Investment*—Decision support, feedback processes sufficient for cost analysis/projection

Web Services Policy (WS-Policy), and specifications WS-PolicyAssertions and WS-PolicyAttachment, can define conditions under which a service is to be provided and provide metrics based on performance of the service. WS-Policy metrics help assess the processing of data by services and composite applications. Service Level Agreements (SLAs) define capabilities and QoS expected from services.

Security is too significant an issue to properly address in this paper, only high level comments will be advanced. Security is mandatory in DOD infrastructure, applications, and services, not only for transmission but also for user authentication and authorization of specific data items in order to implement "need to know" filters. Models would assess data encryption through hundreds of processes, with user authentication and authorization at multiple points

within a distributed SOA. For example, a composite application may integrate web services from multiple sources, each source having different security requirements and different classes of authorization. "Need to know" may change at any time and for any individual or partner organization. Policies may change and certain web services or data types that were previously allowed may no longer be permitted. High-level control and monitor processes would help ensure compliance. Enablers of security processes to be modeled would include:

- *Content Security—XML Encryption, XML Signature*
- *Message-Level Security—WS-Security*
- *Secure Message Delivery—WS-Addressing, WS-ReliableMessaging, WSReliableMessagingPolicy*
- *Metadata—WS-Policy, WS-PolicyAssertions, WS-PolicyAttachment, WSSecurityPolicy*
- *Trust Management—SAML, WS-Trust, WS-SecureConversation, WSFederation*
- *Public Key Infrastructure—PKCS, PKIX, XKMS.*

VIII. SEA Model Scenario

A test scenario was developed for the SEA models. The scenario assumes a range of DoN services to diverse participants, including non-governmental agencies (NGA), medical facilities and staff, and civil authorities. To each of these entities the DoN provides support services in the event of an emergency or natural disaster. In the test scenario, DoN SOA services are interfaced with non-military services and through provided publish and subscribe interfaces the civilian agencies are able to request DoN logistics and personnel support to provide local assistance. Enablers for this process include SOA services within ships providing rescue services; services between ships and the nearest shore Network Operations Center (NOC) for enterprise communications; and services to all coalition, non-governmental and civilian participants.

The first industry-to-DoDAF model integration methodology to be discussed is a BPMN to OV-5b model characterization (Figure 1). The OV-5b Operational Activity Model describes activities (or tasks), including information flows between activities. Representations address inputs and outputs, constraints, and activities.

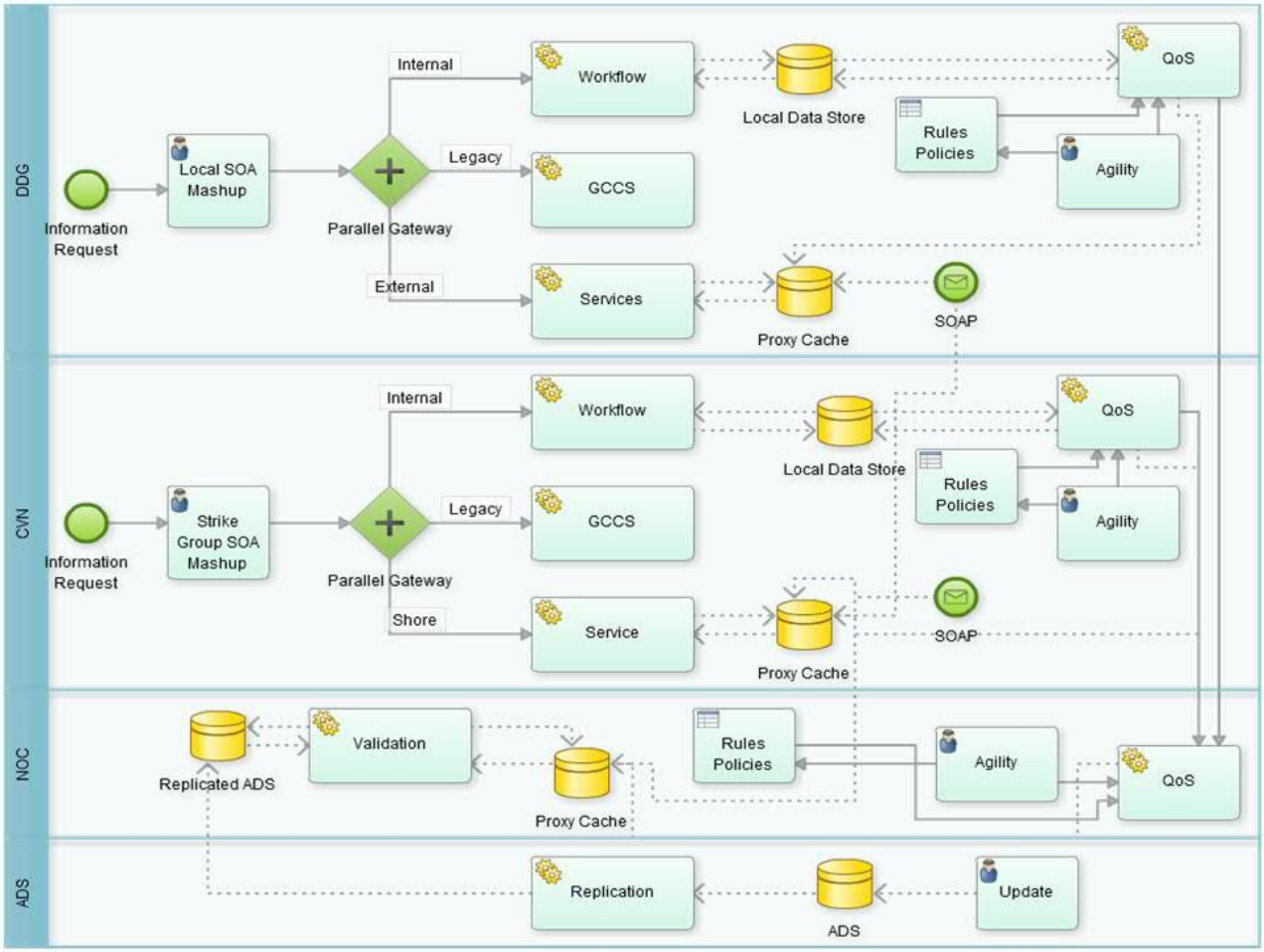

Figure 1. BPMN OV-5b activity model.

In the current context, the activity diagram will be used to describe operations conducted through the SOA to achieve a mission or goal. Historically, activity models used in the DoD originated from Integration Definition for Activity Modeling (IDEF) Standards, and also the Federal Information Processing Standard (FIPS) published by the National Institutes for Standards & Technology (NIST). More recently, architecture development using structured methods has included those utilizing Business Process Modeling Notation (BPMN), developed by the Business Process Management Initiative, with specifications from the Object Management Group (OMG).

BPMN is a graphical representation for specifying processes and provides a graphical notation based on a flowcharting technique very similar to activity diagrams as depicted in Unified Modeling Language (UML). The objective of BPMN is to support process management for both technical users and business users by providing a notation that is intuitive to users yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphics of the notation to the underlying constructs of the execution language(s).

The model in figure 1 shows process flow across SOA nodes. NOCs, CVN nodes, and Tactical units (DDG) are modeled in swim lanes. Operational activities are modeled in each command, with systems and services represented as resources or capabilities. The model assumes unit-specific portals and mashups for SOA services, and widget-based dashboards customized for each unit.

An Authoritative Data Source (ADS) produces mission-critical information that is consumed by tactical users. Continuing the previous example, in this instance a Shore Command issues a rescue mission that is based on procedures in an ADS. This mission is published to ship commands, which in turn publish tactical implementations of that mission to deployed units. Deployed units publish status information back to the command, which in turn synthesize that information for publication back to shore. In the national disaster scenario, the DoN Fleet nearest the disaster coordinates responses with local hospitals, emergency teams, and civil authorities.

The second industry-to-DoDAF model integration methodology to be discussed is a SoaML subsystem modeled to an SvcV-2 Services Resource Flow Description. The DoDAF-2 SvcV-2 specifies resource flows between services, and may also list the protocol stacks used in connections. The model is used to give a precise specification of a connection between services. This may be an existing connection or a specification of a connection that is to be made in the future.

A complementary capability is the SoaML subsystem to represent independent, behavioral components in a system. In SoaML a system can be characterized as a hierarchy of subsystems—represented in class, component, and use-case diagrams. Behaviors of the subsystem are indicated through interfaces, services between interfaces, and resource flows that support the interfaces. Attributes, operations, interfaces, and realizations provide the internal structure of the subsystem.

The model begins with the concept of SOA in the integrated enterprise and service delivery as a core capability for Fleet services to civil authorities. Within the comprehensive SOA node, represented as a CVN or LHD node in Figure 2, the primary subsystems are the enabling SOA components: the repository, application server, enterprise service bus, database server, registry, and security servers. Specific resource flows are not indicated at this high level since each component communicates with each of the others, and for most transactions. The direction and hierarchy of the exchanges would depend on the specifics of the transaction. Modeling of specific resource flows, for specific transactions, would be a logical extension. NGA and Maritime Domain Awareness (MDA) operations would extend the capabilities of the CVN/LHD nodes through interaction with different participants.

The third and final industry-to-DoDAF model integration methodology presented is a SoaML to SvcV-10b Services State Transition Description. SvcV-10b is a graphical method of describing a resource (or function) and its response to various events when changing state. The diagram represents the sets of events to which the

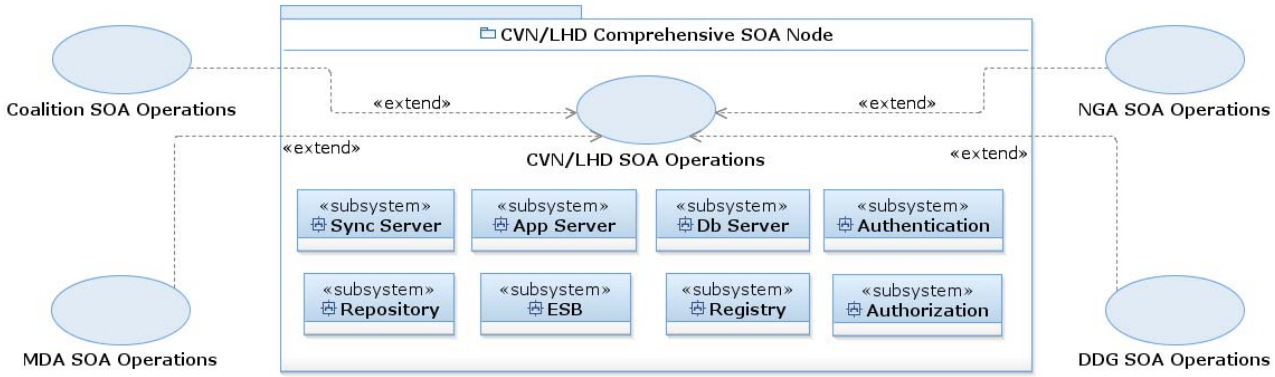

Figure 2. SoaML SvcV-2 resource flow description.

resources in the activities respond (by taking an action to move to a new state) as a function of its current state. Each transition specifies an event and an action. A SvcV-10b can be used to describe the explicit sequencing of service functions. SvcV-10b is based on the statechart diagram. A state machine is defined as a specification that describes all possible behaviors of some dynamic view element. Behavior is viewed as a traversal of a graph of specific states interconnected by one or more joined transition arcs that are triggered by the dispatching of a series of event instances. During this traversal, the state machine executes a series of actions associated with various elements of the state machine [24].

UML state machines refer to two kinds of state machines: behavioral state machines and protocol state machines [25]. Behavioral state machines can be used to model the behavior of individual entities (e.g., class instances). Protocol state machines are used to express usage protocols and can be used to specify the legal usage scenarios of classifiers, interfaces, and ports [26]. SEA does not use a formal statechart for the SvcV-10b but rather a SoaML model intended to aggregate service component capabilities and interfaces into comprehensive service architecture (Figure 3). State change in this instance refers to a change in SOA information flows due to changes in publication topics, subscriptions, data elements/attributes, or security for any of these processes. The contract between the publisher and subscriber is validated by security with interfaces for state change.

State change is a facet of the contract process, for contract enablement as well as security review. Ovals in the diagram are UML collaboration use cases for participant interaction, which is where the change occurs. Connection boxes with handles identify service interface ports and protocols. The service contract ensures QoS. Collaboration shows how the participants work together to provide services. Participants collectively verify that underlying constraints are honored.

IX. CONCLUSION

Enterprise-class distributed SOA presents special challenges for modelers. Unlike client/server, SOA may employ dozens of servers, with near-continuous communications between hundreds of components – which may be widely distributed. Network conditions become a factor. In the case of composite applications, which may aggregate different services from different locations, mechanisms to ensure data accuracy and timeliness are not well evolved. Modeling systems, integrated with test and measurement systems, have an important role to play.

DoDAF modeling is robust from the organizational perspective down to the high level systems perspective, and with DoDAF-2 down to the capabilities. Absent is the ability to run simulations or to generate operational computer code from the models. Industry modeling tools and frameworks are more comprehensive with solutions for the entire software development lifecycle: from concept to requirements analysis, to code generation, to deployment, monitoring and maintenance. However they do not address the intricacies of the highly complex DoD processes through which large organizations interface, partner, and then disconnect as the project completes. The natural disaster scenario developed throughout this paper is an example. Sub-contracts and acquisitions are another. In these organizational interfaces the DoDAF excels. Note that only three (3) of the fifty-two (52) DoDAF models were presented.

In sum, this paper presented a methodology to integrate DoDAF and industry modeling techniques to support development of an enterprise SOA that consists of multiple independent SOA nodes or clouds, each of which can operate independently to provide human assistance and disaster relief (HADR) to civilian agencies. When implemented, the intent is that the models are integrated with operational systems such that information flows can be monitored across the lifecycle, and when change is required the models can be updated and change will

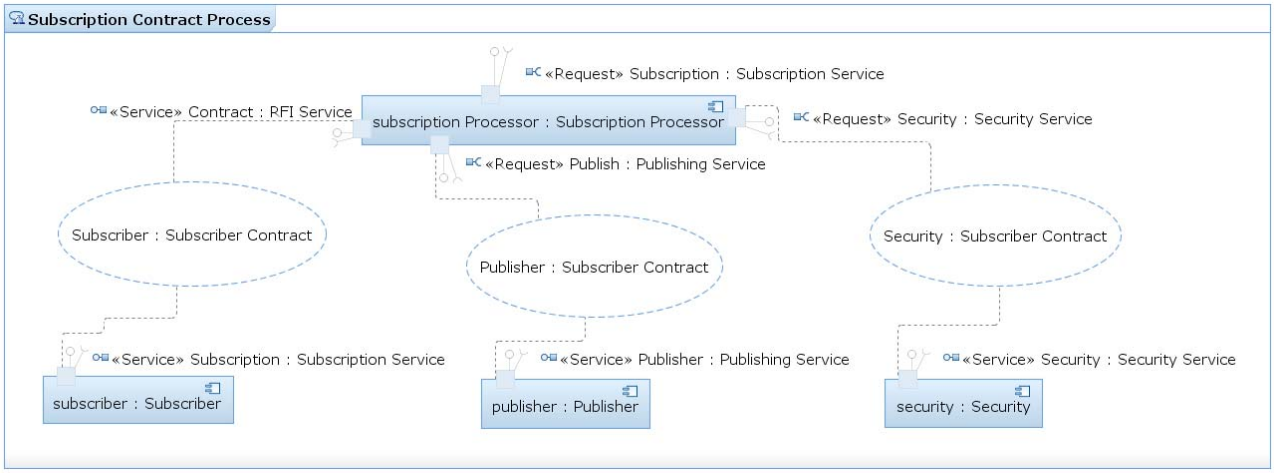

Figure 3. SoaML SvcV-10b service state diagram.

cascade from the models out across the cloud nodes to the SOA components.

ACKNOWLEDGEMENT

The author wishes to acknowledge OPNAV N81 for their support for this research. The model integration methodology herein is unique to the author and is not meant to imply DoD or DoN policy or practice.

REFERENCES

[1] W. Lewis. SOA Characteristics versus DOD Echelon, Washington DC: Pentagon, OPNAV N2/N6, April 2011.

[2] OASIS. Reference Model for Service Oriented Architecture 1.0, August 2006, URL: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[3] OASIS. Reference Architecture for Service Oriented Architecture Version 1.0, April 2008, URL: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf

[4] C. Casanave. Enterprise Service Oriented Architecture Using the OMG SoaML Standard, December 2009. URL: http://www.omg.org/news/whitepapers/EnterpriseSoaML.pdf

[5] OMG. Service oriented architecture Modeling Language (SoaML), URL: http://www.omg.org/spec/SoaML/

[6] A. Huei-Wan, F. Dandashi, M. McFarren, "Tailoring DoDAF for Service-Oriented Architectures," Proc. IEEE Military Communications Conference (MILCOM 2006), IEEE Press, Sep. 2006, pp. 1-8, doi: 10.1109/MILCOM.2006.302044.

[7] URL:http://en.wikipedia.org/wiki/Department_of_Defense_ Architecture_Framework#DoDAF_V2.0_Viewpoints

[8] A.E. Thal, J.D. Havlicek, S.J. Chambal, J.W. Osgood, "Sequencing the development order of architecture products: An Application to DoDAF," Proc. 2010 43rd Hawaii International Conference on System Sciences System Sciences (HICSS), IEEE Press, Jan. 2010, pp. 1-10, ISBN 978-0-7695-3869-3.

[9] J. Zhiping, W. Hongda, H. Ming, B. Guangyu, "The consistency validation method of the DODAF described Models based on meta-data model," Proc. International Conference on E -Business and E - Government (ICEE), IEEE Press, Aug. 2011, pp. 1-6, doi: 10.1109/ICEBEG.2011.5882710.

[10] B.P. Zeigler, S. Mittal, "Enhancing DoDAF with a DEVS-based system lifecycle development process," Proc. IEEE International Conference on Systems, Man and Cybernetics. Vol. 4, IEEE Press, Oct. 2005, pp. 3244-3251, doi: 10.1109/ICSMC.2005.1571646.

[11] E. Baumgarten, S.J. Silverman, "Dynamic DoDAF power tools," Proc. IEEE Military Communications Conference (MILCOM 2008), IEEE Press, Jun. 2008, pp. 1-6, doi: 10.1109/MILCOM.2008.4753127.

[12] P. Xing, Y. Baoshi, H. Jianmi, "Modeling and simulation for SoS based on the DoDAF framework," Proc. 9th International Conference on Reliability, Maintainability and Safety (ICRMS), Jun. 2011, pp. 1283-1287, doi: 10.1109/ICRMS.2011.5979468.

[13] M. Hause, "The Unified Profile for DoDAF/MODAF (UPDM) enabling systems of systems on many levels," Proc. 4th Annual IEEE Systems Conference, IEEE Press, Apr. 2010, pp. 426-431, doi: 10.1109/SYSTEMS.2010.5482450.

[14] IBM. UML Profile-Based Integrated Architecture (UPIA) DoDAF 2.01 Mapping Reference, June 2010, URL: http://www.ibm.com/developerworks/wikis/download/attachments/133497801/UPIA_DoDAF2.01_Reference_v8.0.pdf

[15] The Open Group. The Open Group Architecture Framework (TOGAF), URL: http://pubs.opengroup.org/architecture/togaf8-doc/arch/

[16] I. Todoran, Z. Hussain, N. Gromov, "SOA integration modeling: An evaluation of how SoaML completes UML modeling," Proc. 15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), IEEE Press, Aug. 2011, pp. 57-66, doi: 10.1109/EDOCW.2011.48.

[17] S. Jacobi, C. Hahn, D. Raber, "Integration of multiagent systems and Service Oriented Architectures in the steel industry," Proc. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Vol. 2, IEEE Press, Aug. 2010, pp. 479-482, doi: 10.1109/WI-IAT.2010.218.

[18] N. Ali, M.A. Babar, "Modeling Service Oriented Architectures of mobile applications by extending SoaML with ambients," Proc. 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009), Aug. 2009, pp. 442-449, doi: 10.1109/SEAA.2009.25.

[19] M. Abu-Matar, H. Gomaa, "Feature based variability for Service Oriented Architectures," Proc. 9th Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE Press, Jun. 2011, pp. 302-309, doi: 10.1109/WICSA.2011.47.

[20] B. Hoisl, S. Sobernig, "Integrity and confidentiality annotations for service interfaces in SoaML models," Proc. Sixth International Conference on Availability, Reliability and Security (ARES), Aug. 2011, pp. 673-679, doi: 10.1109/ARES.2011.105.

[21] G.B. Ayed, S. Ghernaouti-Helie, "Digital identity management within networked information systems: From vertical silos view into horizontal user-supremacy processes management," Proc. 14th International Conference on Network-Based Information Systems (NBiS), Jan. 2011, pp. 98-103, doi: 10.1109/NBiS.2011.24.

[22] N. Mani, D.C. Petriu, M. Woodside, "Studying the impact of design patterns on the performance analysis of Service Oriented Architecture," Proc. 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Aug. 2011, pp. 12-19, doi: 10.1109/SEAA.2011.13.

[23] A. Sadovykh, P. Desfray, B. Elvesaeter, A. Berre, E. Landre, "Enterprise architecture modeling with SoaML using BMM and BPMN - MDA approach in practice," Proc. 6th Central and Eastern European Software Engineering Conference (CEE-SECR), Nov. 2010, pp. 79-85, doi: 10.1109/CEE-SECR.2010.5783155.

[24] URL: http://cio-nii.defense.gov/sites/dodaf20/services-10b.html

[25] URL: http://en.wikipedia.org/wiki/UML_statechart

[26] OMG. OMG Unified Modeling Language (OMG UML), Superstructure Version 2.2, Feb. 2009, URL: http://www.omg.org/spec/UML/2.2/Superstructure/PDF/