WBB CONSULTING
Solutions and Support for a Changing World

Services in DoDAF V2.0:

A Methodology for Making Services Modelable and Relevant in DoDAF V2.0

Submitted to the ***a|EA Journal*** for Publication

Lawrence P. McCaskill
Sr. Manager / Chief Enterprise Architect
WBB Consulting
11790 Sunrise Valley Drive
Reston, VA 20191
703-448-6081 x127 Fax: 703-821-6955
lmccaskill@wbbinc.com

Andy D. Rogers
Manager, Enterprise Solutions
WBB Consulting
11790 Sunrise Valley Drive
Reston, VA 20191
703-448-6081 x313
arogers@wbbinc.com

Services in DoDAF V2.0:
A Methodology for Making Services Modelable and Relevant in DoDAF V2.0

ABSTRACT

This white paper advocates for a complete restructuring of the Services Viewpoint and Views within the DoDAF V2.0. It introduces the concept of overloading of the term "Service" within DoDAF V2.0, and provides a means of clarifying what is meant in the DoDAF regarding Services via the introduction of the term "Commoditized Service" into the DoDAF vernacular. The Commoditized Service is a manifestation of Service Oriented Architectures (SOA) at a higher level of abstraction than Web Services; it is not in-and-of-itself a Performer - the definition of Service in DoDAF V2.0 states a Service requires a Performer [as a Mechanism] to execute. The Commoditized Service (as well as the Web Service), requires a Service Level Agreement to declare available functionality for the Service. This paper introduces the concept of the SLA as a means of "information hiding" for the Commoditized Service, which allows for the manifestation of 3 concepts: 1) Capabilities as Systems that are bought or developed outright that are internal or functionally specific applications, with no intention of offering underlying capabilities as a Service 2) Outsourcing Capability (or parts of a Capability) to Commoditized Services 3) Building the Service "in house" with the intention of offering it as a Commoditized Service. Each has different requirements regarding development of DoDAF artifacts; each case is discussed in detail as to the artifacts required for their manifestation. Finally, the paper proposes a means of management of the underlying data associated with Commoditized Services. As such, what is presented is a logical construct for accommodating and modeling Commoditized Services, Web Services, Systems, Organizations, and People, providing value added to the architect and the organizations they support.

Services in DoDAF V2.0:
A Framework for Making Services Modelable and Relevant in DoDAF V2.0

SERVICES AND DODAF V2.0 – OVERLOADING OF SERVICE AS A TERM

In its attempt to address Service Oriented Architecture concepts, the DoDAF V2.0 has adopted the OASIS definition for use within DoDAF V2.0:

> *SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations (http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf).*

In attempting to utilize the SOA paradigm, the DoD Architecture Framework V2.0 defines service in the following manner:

> ***Service**: A mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. The mechanism is a Performer. The capabilities accessed are Resources -- Information, Data, Materiel, Performers, and Geopolitical Extents.*

As "Jedi Knight practitioners" of DoDAF, with over 100 Solutions and several Enterprise architectures on the books, as well as an excellent understanding of Services and Service Oriented Concepts, upon reading this definition of Service, we collectively said to ourselves: "What in the heck does that mean…?" However, upon re-reading the definition in the context of the overall DoDAF V2.0, we came to the conclusion that the DoDAF V2.0 and its related DoDAF metamodel (DM2) are out of synchronization regarding their respective definitions of Service; the DM2 defines Service as a subtype of Performer. Via the definition above, one can see a Service is not a Performer. Utilizing DoDAF terminology, it is a standardized means for providing access (i.e., a mechanism) to invoke a Performer (which must be a real person, organization or system located somewhere with access to resources) to conduct one or more Activities. Defining a Service includes describing standardized inputs and expected, measurable outputs and results that must be included as part of invoking the performer to action and obtaining a "reusable" capability.

Delving deeper, it became clear that the authors of the DoDAF were overloading[1] the term "Service." How did we come to this conclusion? In the introduction to DoDAF V2.0 in Volume 1, it states:

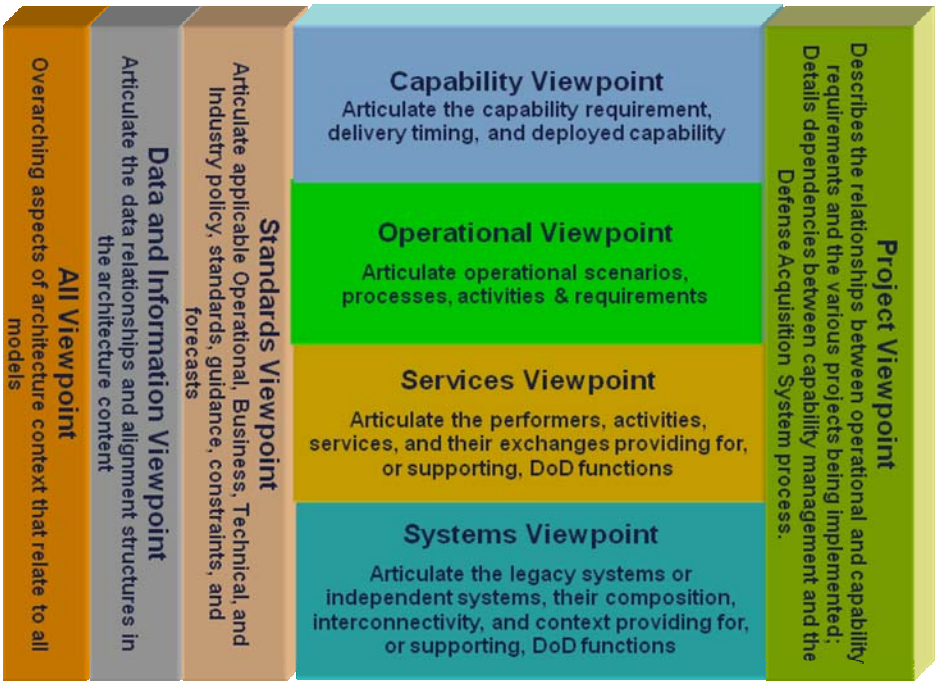
[1]Note: "overloading" is a software engineering term; it means that one is allowed to use the same name for two different pieces of code (called "methods" in object oriented languages), and have the program choose the code to execute based on the method's input/output parameters. Stated differently: it allows one to use the same term to infer different things, based on the context associated with the term's use.

> *DoDAF V2.0 expands previous framework development efforts to capture architecture information about net-centricity, support Departmental net-centric strategies, and describe* ***service-oriented****…* (emphasis ours) *…solutions that facilitate the creation and maintenance of a net-centric environment…* ***Over time, the Department's emphasis on Service Oriented Environment and Cloud Computing may result in the elimination of the Systems Viewpoint...*** (emphasis theirs; cooler heads have prevailed, and this last bit verbiage has been removed from subsequent versions of the DoDAF V2.0 on the wiki as well as the public site).

So… Service = Web Service… the current accepted means of implementing Service Oriented Architecture, right? Not so fast… when they describe Service in Volume 2, they describe it as:

> *… A Service, from a software service to a business service such as Search and Rescue…*

Hmm… they've used this quite often during discussions of presentations of DoDAF V2.0… Combat Search and Rescue (CSAR) as a "Service." OK, that's a stretch (where's my service level agreement [SLA] for that?) but we'll play along to see where this leads. One could state that there is an implied SLA that the DoD executes with aircrews (if the aircraft is downed, we'll make every attempt to get you out of harm's way, and have assets allocated to do so), but it isn't normally identified as a "Service" in the SOA sense. Nonetheless, via the definition, as well as the descriptions of the Viewpoints describing them, DoDAF V2.0 infers that Services include people, systems, software-as-a-service (SaaS), hardware-as-a-service (e.g., Cloud Computing), and storage-as-a-service (various continuity of operations vendors offer offsite backup/replication of data). So, "Service" in this section of the DoDAF V2.0 isn't addressing Web Services – it's a higher level of abstraction, and includes just about anything for which one can write a contract, enumerated in the contract via contractor line item numbers (CLINs). Lacking a better term, one could call this concept a "Commoditized Service" – we'll use this term throughout the rest of this document to address this concept. Further evidence of DoDAF V2.0's use of "Service" to mean "Commoditized Service" is the overarching diagram often referred to as the "wedding cake:"



By placing the Services Viewpoint between the Operational Viewpoint and the Systems Viewpoint, the implication is that the Operational Viewpoint defines the relevant business processes that make up the Capability. As we move layers downward in the "Wedding Cake," one functionally allocates everything in the DM2 categorized as "Performer" (Organization, Person Type, System, and Service) to the Activities that define one's Capability in the Operational View. Moving through the layers, here's the implication of allocation in the various Viewpoints:

- The Capability Viewpoint enumerates associated Operational Activities. Using CSAR as the example, example activities might include Report Downed Airman, Assign Recovery Team, Locate Downed Airman, and Extract Downed Airman.
- The Operational Viewpoint allocates Activities to Organizations, defines their orchestration, captures business rules about the orchestration (i.e., doctrine, tactics/techniques/procedures, etc.) and information required to be exchanged to implement the Capability. Using CSAR as the example, this elaborates on the processes associated with CSAR, and information exchanged during the execution of this capability. Further allocation shows that the organization receiving the report of the downed airman is the Joint Personnel Recovery Center (JPRC), who will task CSAR assets (Assign Recovery Team) to make the attempt to Locate and Extract a Downed Airman. Such doctrine as the "word of the day" is also captured during the rules associated with the activities
- The Services Viewpoint was designed to be able to functionally allocate higher-level Operational Activities to lower level Service Functions (i.e., types of Services that could then eventually be allocated to actual Services that accomplish them). The implication is the Services Viewpoint is a "contracting viewpoint" – and details how organizations could build services that were useable across the enterprise, including the paradigm of allowing multiple similar Service Functions to be assigned to Services that implement them. It allows the organization to allocate the activities to the systems, sub-organizations, Web and other Services contracted for, and people to run the systems.

- The Systems Viewpoint was left in as ostensibly a "legacy" viewpoint, and thus, it was left largely unchanged from DoDAF V1.5. The implication in several places in the text was that the Systems Viewpoint would eventually "go away." Since the original release, the authors have backed off on this assertion, but the original language in the DoDAF V2.0 stated the Systems Views were legacy views, and that their use would diminish to the point where they would no longer be used.

THE NEED FOR A NEW DEFINITION: THE COMMODITIZED SERVICE

OK… with that explained, we might have been willing to hit the Staples Easy Button, and say "we believe" – it's a viable story. But, if one looks closer at the underlying DoDAF V2.0 Views associated with the Viewpoints, the argument breaks down completely, because Systems and Services Viewpoints have almost ***exactly the same*** views, and call for you to capture almost ***exactly the same*** data. As taxpayers, we were appalled – as consultants we were intrigued. This creates an entirely unneeded level of complexity for the architect, via the creation of yet another set of artifacts to relate via matrix-based views (SvcV-5's, SvcV-6, which oh-by-the-way one needs to align with the SV-5's and the SV-6's in addition to the OV-3 and OV-5. Add to that matrices for the CV's and PV's too…).

Systems Views		Services Views	
SV-1 Systems Interface Description	The identification of systems, system items, and their interconnections.	SvcV-1 Services Context Description	The identification of services, service items, and their interconnections.
SV-2 Systems Resource Flow Description	A description of Resource Flows exchanged between systems.	SvcV-2 Services Resource Flow Description	A description of Resource Flows exchanged between services.
SV-3 Systems-Systems Matrix	The relationships among systems in a given Architectural Description. It can be designed to show relationships of interest, (e.g., system-type interfaces, planned vs. existing interfaces).	SvcV-3a Systems-Services Matrix	The relationships among or between systems and services in a given Architectural Description.
		SvcV-3b Services-Services Matrix	The relationships among services in a given Architectural Description. It can be designed to show relationships of interest, (e.g., service-type interfaces, planned vs. existing interfaces).
SV-4 Systems Functionality Description	The functions (activities) performed by systems and the system data flows among system functions (activities).	SvcV-4 Services Functionality Description	The functions performed by services and the service data flows among service functions (activities).
SV-5a Operational Activity to Systems Function Traceability Matrix	A mapping of system functions (activities) back to operational activities (activities).	SvcV-5 Operational Activity to Services Traceability Matrix	A mapping of services (activities) back to operational activities (activities).
SV-5b Operational Activity to Systems Traceability Matrix	A mapping of systems back to capabilities or operational activities (activities).		
SV-6 Systems Resource Flow Matrix	Provides details of system resource flow elements being exchanged between systems and the attributes of that exchange.	SvcV-6 Services Resource Flow Matrix	It provides details of service Resource Flow elements being exchanged between services and the attributes of that exchange.
SV-7 Systems Measures Matrix	The measures (metrics) of Systems Model elements for the appropriate timeframe(s).	SvcV-7 Services Measures Matrix	The measures (metrics) of Services Model elements for the appropriate timeframe(s).
SV-8 Systems Evolution Description	The planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation.	SvcV-8 Services Evolution Description	The planned incremental steps toward migrating a suite of services to a more efficient suite or toward evolving current services to a future implementation.
SV-9 Systems Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future system development.	SvcV-9 Services Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future service development.
SV-10a Systems Rules Model	One of three models used to describe system functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.	SvcV-10a Services Rules Model	One of three models used to describe service functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SV-10b Systems State Transition Description	One of three models used to describe system functionality. It identifies responses of systems to events.	SvcV-10b	One of three models used to describe service functionality. It identifies responses of services to events.
SV-10c Systems Event-Trace Description	One of three models used to describe system functionality. It identifies system-specific refinements of critical sequences of events described in the Operational Viewpoint.	SvcV-10c Services Event-Trace Description	One of three models used to describe service functionality. It identifies service-specific refinements of critical sequences of events described in the Operational Viewpoint.

This also tells the architect there's no difference between the Systems and the Services Viewpoint – which is not necessarily true if one uses the explanation outlined above – the original Views that make up the Services Viewpoint were just poorly elaborated. Additionally, the authors consistently refer to "Service" meaning "Web service" in support of Net-Centricity throughout the document, which is a different concept than the "Commoditized Service" concept implied earlier in the documentation.

Thus, we're proposing that the DoDAF V2.0 differentiate between the two types of services via the following explanation of the definition for Commoditized Service in DoDAF:

Commoditized Service: a service is the result of one or more Activities forming a Capability providing something of potential intrinsic value when instantiated by a Performer*. A Commoditized Service ***is not*** in-and-of-itself a Performer; it is a commodity item that is not realized until it is invoked or instantiated by an organization performing at one or more locations and requires Performers for invocation/instantiation. The expectations for what the service does, the resources it consumes and/or provides, and how well it should perform (including security and management considerations) are controlled via a contract or SLA, whether this contract/SLA is implicit or explicit.

* **Note**: the subtype PersonType under Performer needs to be expanded to include all biological systems – concrete examples: bomb/drug/assistance dogs, monkeys assisting the disabled, dolphins performing mine hunting, etc. Each of these is indeed a Performer.

This is distinctly different than a Web Service, which is defined by the W3C as follows:

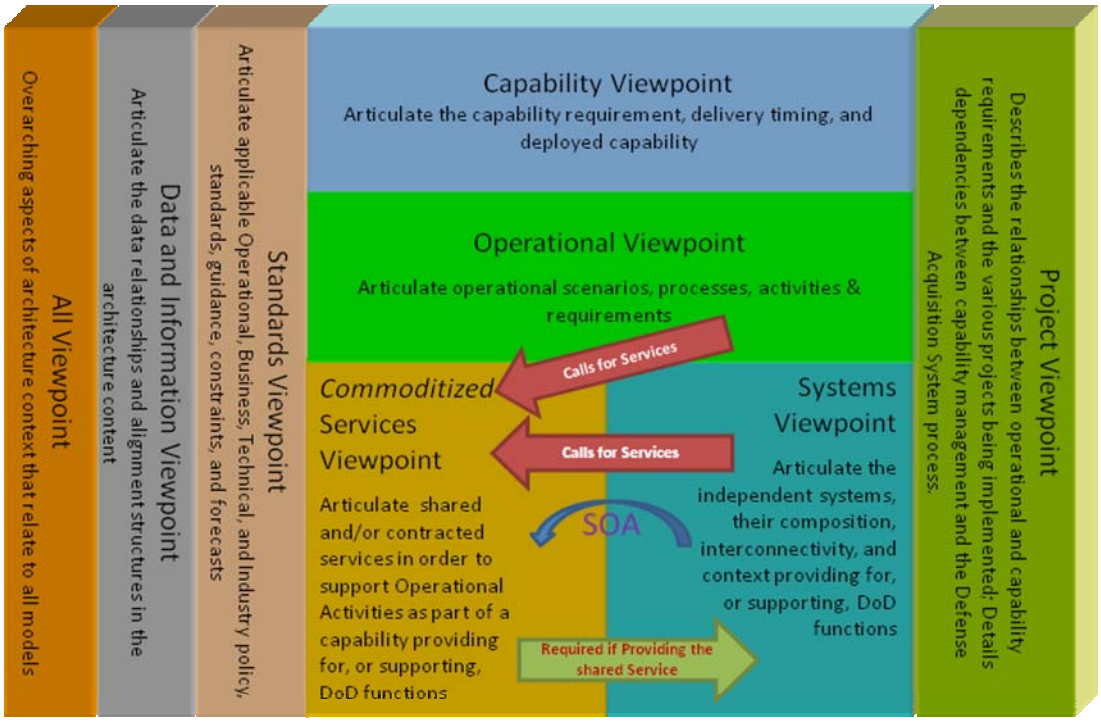
*A **Web service** is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards (http://www.w3.org/TR/ws-arch/#id2260892).*

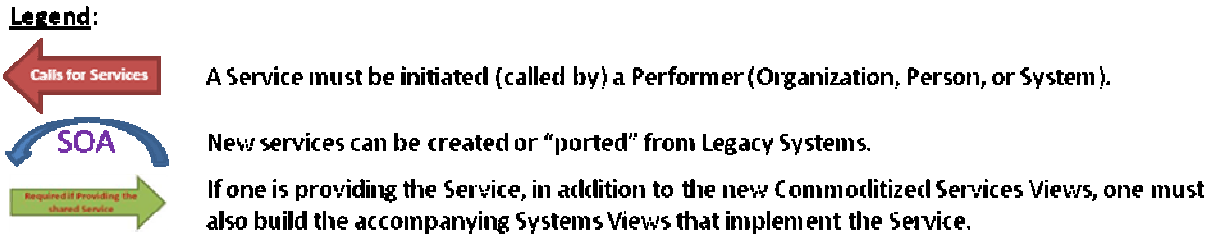
This paradigm would enable the definition of the allocation and contracting for Commoditized Services, while allowing for those services to be composed of, in whole or part, Web Services. While both Commoditized Services and Web Services generally share the following SOA paradigms, the Commoditized Service is a "supertype," which encompasses Web Services, Systems, and Organizations (down to PersonType at its most granular instance) performing the service. The SOA paradigms shared by Commoditized Services and Web Services are:

- Abstraction: beyond what is described in the service contract, services hide logic from the outside world
- Autonomy: services have control over the logic they encapsulate
- Composability: collections of services can be coordinated and assembled to form composite services
- Discoverability: services are designed to be outwardly descriptive so that they can be found and accessed via available discovery mechanisms
- Formal Contract: services adhere to a communications agreement, as defined collectively by one or more service description documents
- Loose Coupling: module interacts with another module through a stable interface and does not need to be concerned with the other module's internal implementation
- Reusability: logic is divided into services with the intention of promoting reuse

"WEDDING CAKE" CHANGES DRIVEN BY THE COMMODITIZED SERVICE (INFORMATION HIDING AS PART OF THE COMMODITIZED SERVICE SLA)

This effectively changes the "wedding cake" diagram as follows:





This representation makes Commoditized Services co-equal to Systems. This makes sense, because of the following 3 conditions:

1. **All Activities Can Be Functionally Allocated to Systems:** There are cases where all Activities as part of a Capability are functionally allocated to Systems Functionality (i.e., Systems Functions) that will be bought outright or built internally, with no intention of offering any of the functions for consumption as a "Commoditized Service." Real time systems and systems that are internal to other platforms fit this niche. In this case, all Activities are allocated to System Functions and subsequently Systems in the Systems Viewpoint. When fully allocated, the System Functions are allocated to the System types, Organization Types, and Person-Types (i.e., Roles), to perform the Activities (these can be further realized by identifying locations, actual organizations, and actual systems nomenclature; this allows one to move from what the DoDAF V2.0 calls "Role Based" to "Actual"). The implication is the systems being built or utilized are not

foreseen to be made available via service contracts to outside organizations. This obviously can change over time – system functions in the future can be outsourced via re-allocation to Commoditized Services and/or offered as a service to others via building of the appropriate interfaces to be discussed in the Commoditized Services Views.

2. **Some or all Activities Functionally Outsourced:** Some or all Activities (or sub-Activities), as part of a Capability, are allocated to external Commoditized Services. In this case, the Activity allocated to the Commoditized Service is modeled as an external Function, and all that one needs to know about it is the Function name, its Inputs, and its expected Outputs. These are expressed in the contract for the Commoditized Service (i.e., the SLA). No further modeling of the service is required in this case, as the information isn't required; one has a contract for provided Capability via the SLA. Key implication: Activities are either implicitly or explicitly "contracted for" at this point from an Organization that performs the Commoditized Services; this further implies there's a known portfolio of Organizations and their related services, including service levels, to choose from. This implies a level of governance of services made available by organizations – "a means" by which to accomplish this will be covered in subsequent paragraphs in this paper. However, this also implies that a Commoditized Services View for this portfolio of available services is required.

   If an Activity is outsourced, the Commoditized Services Viewpoint describes needs to describe the mechanisms by which the Commoditized Services are contracted for, using only information that is "exposed to the world" which provides information hiding of how the "service industry services" implement their contracts (i.e., functionally allocate Performers within their service), allowing for the commoditized service to have control over, and potentially change the allocation of underlying Performers, as long as the contract (i.e., the Service Level Agreement [SLA]) is met (thus meeting the tenets of performance-based contracting). This implies another Commoditized Services View that is required.

3. **Some or All Activities Functionally Allocated to Services Owned or to be Built by the Organization:** Some or all Activities (or sub-Activities), as part of a Capability, are functionally allocated to Commoditized Services that are going to be made available internal to the enterprise, or both internal to the enterprise as well as external to the enterprise. In this case, a "Service Contract" or SLA is required for development in the Commoditized Services Viewpoint, as well as the underlying Systems Views in the Systems Viewpoint.

   Note: we didn't say Services Views intentionally – we're developing Systems that instantiate a Service at this point. DoDAF V2.0 states Systems (and Services, for that matter… the descriptions in the text are a "block copy" of one another) consist of hardware, software, and Organizations/PersonType that run them.

   Thus, in addition to defining the interfaces to the Commoditized Service in the Commoditized Service Viewpoint, there is a need to elaborate on the high-level design of the Commoditized Service. The Systems Views handle this lower-level allocation of

Activity to system functionality perfectly fine, without having to create an entirely separate set of Views to accommodate the relationships depicted. Thus, the Systems Viewpoint will describe the lowest level of functional allocation – the full implementation of organizations, people, systems (hardware and software), and "Web Services" that are interconnected in order to provide each performer's respective parts of the overarching system-of-systems providing a capability.

All of these Views are managed via the Project Viewpoint, which manages and maintains all the functional allocations, and gets the system-of-systems built and into the field. This construct allows one to describe the allocation of functionality to a Commoditized Service, as well as its underlying "hidden" functionality, which are required to be elaborated on as part of a program office's duties. In this case, one develops the Commoditized Service Views that will be exposed to the world, allowing for the service to be contracted for, as well as Systems Views that provide the requirements for development. But… one is reminded we're documenting functional allocation rather than actual implementation – we're allowing for the Commoditized Service to be manifested in many ways and evolve over time. As long as the SLA is met and the interfaces remain stable, the underlying technology can, and will, change based on market forces without affecting the users of the service.

Thus, the Commoditized Services Viewpoint is effectively an "outsourcing viewpoint," and only required in the case of Commoditized Services being functionally allocated to Operational Activities, and outsourced (or "insourced") as part of the system-of-systems solution. The Commoditized Services Viewpoint requires its own set of Views that are missing from DoDAF V2.0. These are elaborated on in the following section.

VIEWPOINTS REQUIRED FOR IMPLEMENTATION OF COMMODITIZED SERVICES

The current Services Viewpoint in DoDAF V2.0 is nearly an exact word-for-word copy of what is called for to be collected for the Systems Views. However, if you ascribe to the dogma of Commoditized Services being between the Operational View and the Systems View, the Viewpoints called for by DoDAF are redundant and do not "answer the mail," and thus, we are advocating for the complete removal of the existing Services Views. What is needed for the Commoditized Service Viewpoint are Views that describe the interface to the service provider, govern the services, allow the service provider to hide information related to the service implementation, and thereby allow performance-based contracts for services to be established. The Views required to implement this properly include the following:

- **Commoditized Service Discovery View** (Who and Where): a description of the means by which a Commoditized Service is discovered. It provides the means for capture of the Service Types available for contract, the Organizations providing them, and where the Commoditized Services can be obtained (physical location or Uniform Resource Locator [URL]). This allows for Organizations to enumerate services available for contract, and other Organizations to contract for the services via a standardized mechanism.

- **Commoditized Service Description** (What and Why): describes what the Commoditized Service accomplishes; includes expected inputs, expected outputs, and options available for delivery of the outputs, including such things as latency and means of delivery. Describes intended use of the Commoditized Service (why); this does not preclude the service from being used for other applications, but allows the organization contracting for the service to be more informed in selecting the service. For a publish/subscribe environment, this would also discuss the means of developing the user profile for systems, humans, and Web Services, and the expected results of subscription to the Commoditized Service.

- **Commoditized Service Contract View** (How): this is the Service Level Agreement, and allows the customer and the service provider to establish a shared agreement managing the expectations of the service delivery contract. The parallel paradigm for the SLA methodology related to Commoditized Services is the exposed header code in C++ or any other object oriented language. This header is a performance-based contract "in the small" – it doesn't tell one how the underlying code does something; it merely states what the functions and sub-functions do, expected inputs and their respective data types, and outputs and their respective data types. Here, the Commoditized Service has taken this information hiding construct and applied it to a performance-based contract, using the SLA as the contract's binding agreement. It includes a description of the available means of contracting for the service, delivery options, and clauses regarding metrics-based delivery of the service; includes, but not limited to:
  - Scope of Work
  - Fee structure; examples include:
    - Free with advertising (think Weatherbug or Google Maps "free" editions)
    - Fee for service: describes different funding options, including, but not limited to: fee for one-time use, fee for single user (a Performer: person, system, or service can all be "users") use over time period, fee per multiple user concurrent use over time period, fee per resources used (Cloud Computing, offsite backup, continuity of operations invocation and use), etc.
  - Clauses describing metrics-based expectations regarding delivery of service. This includes such items as: expected service latency, metrics-based quality-of-service expectation, service availability, and service scalability
  - Clauses describing metrics-based expectations regarding the quality of user inputs and/or availability of the user if user input is required during the delivery of service
  - Funding vehicles available and means of accessing these (if applicable)
  - Primary and Secondary Points of Contact (POC) for both the User and the Commoditized Service. The Commoditized Service POC is provided in order to facilitate the User to contact the service provider in the event of service unavailability or improper execution as contracted in the SLA, or in the extreme case, service contract termination. Conversely, if the service provider cannot provide the service, it allows for the service provider to contact the User to let him know the service will be temporarily unavailable, or in the extreme case, the service is to be terminated.

  - Expected User Inputs, including format, and locations where the formats are prescribed
  - Outputs of the Commoditized Service including format, and locations where the formats are prescribed

- **Commoditized Service Security View:** describes security measures adopted as part of applying the service to the target architecture and/or the requirements for security that will be incorporated by the service provider within their Commoditized Service.

- **Commoditized Service to System Matrix:** links Commoditized Services to the Systems that enable the Commoditized Service. This matrix is only required in the case of a new service being created, maintained, and/or offered by the enterprise that is being described in the architecture; i.e., it is not required for architectures that are outsourcing one or more Commoditized Services.

- **Commoditized Service to Operational Activity Matrix:** details the functional allocation of Operational Activities.

This paradigm also allows for Web Service models (software-as-a-service, storage-as-a-service, computing-as-a-service, etc.) to be used to implement a reusable system being procured or developed as part of a project. These concepts were all readily expressible in the existing DoDAF V1.0 and V1.5 Products, which have been carried forward to DoDAF V2.0 in the Systems Viewpoint and Views. Composablity (a key tenet of SOA) was already "in there" – we don't need to reinvent the wheel to model these constructs. In the case where the Program Office is *building* the Commoditized Service (rather than *contracting for* the Commoditized Service), the Systems Views can also be used to express the constructs "hidden" behind the Commoditized Services Viewpoint.

Reviewing the concept "in a nutshell:"

Capability is the ***result of***
One or more Activities ***provided by***
Functional allocation of Activity to a combination of
System Types and/or Service Types, ***invoking***
Performers (Organizations, PersonTypes, and Systems
([including Systems invoking Web Services]) ***producing***
Something of potential intrinsic value (***a Desired Result or Resource***)

A MEANS OF GOVERNANCE OF COMMODITIZED SERVICES

To simplify the means by which we create and use architectures, as part of the Architecture Federation construct suggested by DoD guidance, we need to do with architectures exactly what the above Commoditized Service construct suggests: Organizations providing Commoditized Services must have a stable interface to the provision of those services (one of the key tenets of

SOA). These key interfaces (described as touch points in the Federal Segment Architecture Methodology) must be explicitly defined, searchable, etc. to realize their full potential.

This in inferred in the DoD Net Centric Data Strategy; the DoDAF V2.0 calls this out as follows:

> *Tiered Accountability (TA) is the distribution of authority and responsibility to a DoD organization for an element of the DoD EA. Under TA, DoD is defining and building enterprise wide capabilities that include data standards, business rules, enabling systems, and an associated layer of interfaces for Department, specified segments of the enterprise (e.g., JCA, DoD Components), and Programmatic solutions. Each tier has specific goals, as well as responsibilities to the tiers above or below them. Architectural Descriptions are categorized when developed to facilitate alignment (mapping and linking), cataloging, navigating, and searching disparate architecture information in a DoD registry of holdings. All Architectural Descriptions developed by the tiers should be federated, as described in the DoD Federation Strategy.*

The guidance seems reasonable, but in practice, it's too vague, isn't explicitly funded, and as such there's no "clearinghouse" for the SLAs that define the DoD Organization's capabilities in order to accomplish the functional allocation of the things organizations accomplish in the doctrine that define what they do. Without the contracts and the specifically defined interfaces to access the services provided by the organizations within the DoD, these mandates for "Tiered Accountability" and "Segmented Architectures" are merely "good ideas." Stated differently, without explicit guidance and funding for the creation of the standard interfaces to our Commoditized Services, we effectively have "unfunded mandates" regarding their use. Therefore, the Commoditized Services model must be implemented as part of DoD Federation Strategy in order to facilitate the federation of lower level architectures (i.e., mission- and platform-level capabilities) with higher level architectures (i.e., strategic capabilities).

One could argue that some of this is being done via the Communities of Interest (COI) model; however, the COIs identified for the Global Information Grid (GIG) Enterprise Services do not match the COIs identified on the DoD Architecture Registry System (DARS), which do not match the COIs being used for the DoD MetaData Registry (MDR), etc. Without formalized COIs providing the standardized interfaces and performance characteristics of their respective Commoditized Services, this cannot work. Sun Tzu stated "*…there is nothing new under the sun…*" – this is the Data Administration concept brought forward into the current time. In both cases, the strategies are failing in large part due to lack of formal staffing and funding; to be successful, the strategy implementation piece needs to be *formally funded,* and *adequately staffed* by personnel that are up to the task. Prior attempts to do the "data part" of this have fallen into the "additional duty" category. If it's an "additional duty" for personnel already putting in upwards of 50 hours a week, it simply won't get done.

Thus, the requirement for administration of Commoditized Services requires the DoD to formalize and adequately staff data administration for the COIs. This will enable the management of data requirements associated with the COI, including (but not limited to) the documenting and exposing the Commoditized Services available from and within the COI.

RECOMMENDATIONS/CONCLUSION

All of this leads to the following recommendations regarding the implementation of Services in DoDAF V2.0 and beyond:

1) Replace the Services Viewpoint with the Commoditized Services Viewpoint, including the complete replacement of the underlying Services Views with the following Views:

    a. Commoditized Service Discovery View
    b. Commoditized Service Description View
    c. Commoditized Service Contract View
    d. Commoditized Service Security View
    e. Commoditized Service to System Matrix
    f. Commoditized Service to Operational Activity Matrix

2) Within the DoDAF Meta Model (DM2), expand PersonType to include all biologic systems. This is needed to describe service animals such as bomb and drug dogs, monkeys trained to aid the disabled, dolphins trained in mine and intruder detection, etc.

3) Assuming this hasn't already been accomplished; explicitly subtype Activity into Operational Activity, System Function, and Service Function in the DM2. However, assuming the Commoditized Service View construct is adopted, Service Function can be removed in its entirety; System Functions suffice for modeling Service Functions.

4) OASD/NII advocate formal codification standardization of Communities of Interest (COIs) within the DoD in order to facilitate the DoD Enterprise Architecture Federation Strategy. It also needs to advocate funding for building, manning, and maintenance of a clearinghouse of available services in the DoD managed by the respective COIs.

These constructs are required for the DoDAF practitioner to make best use of DoDAF constructs, and allow for management of the Commoditized Services within the DoD. In the absence of these constructs, the means by which one models Services (both Commoditized Services and Web Services) in DoDAF are manifold, and creates an unneeded level of complexity in documenting requirements for functional allocation of Capability through Activity through the means of implementation. Stated differently: the goal of the DoDAF is to make active use of the underlying data created as one builds models describing the respective Views. Unless these recommendations are adopted, we foresee a high degree of non-standardized "shelfware" being created to answer guidance that "looks good on paper," but in practice, is untenable. Adoption of these constructs streamlines the DoDAF, and makes it more useable by architecture practitioners in a standardized fashion; this will allow for data to be collected, maintained, and administered standardized fashion for use in actual analyses, providing value to the DoD and taxpayer alike.

Appendix A - References

David Booth, et al., "Web Services Architecture", W3C Working Group Note, World Wide Web Consortium (W3C), February, 2004. http://www.w3.org/TR/ws-arch/#id2260892

DoD Architecture Framework, V2.0, https://www.us.army.mil/suite/page/454707

DoD Architecture Registry System: https://dars1.army.mil/IER2/

DoD Metadata Registry: https://metadata.dod.mil/mdr/homepage.htm

Department of Defense Net-Centric Data Strategy, 9 May, 2003. Office of the Assistant Secretary of Defense (Networks & Information Integration) (NII)/DoD Chief Information Officer (DoD CIO).

Federal Segment Architecture Methodology: http://www.fsam.gov/

Global Information Grid Enterprise Services (GIG ES): http://www.disa.mil/peoges/

Google Maps: http://maps.google.com/maps?hl=en&tab=wl

IEEE-Std-1471-2000 Recommended Practice for Architectural Descriptions for Software-Intensive Systems, Institute for Electrical and Electronics Engineering, New York, NY, 2000.

Service Level Agreement (SLA) description:
Wikipedia: http://en.wikipedia.org/wiki/Service_level_agreement

SOA Description:

- Organization for the Advancement of Structured Information Standards (OASIS): http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf
- The Open Group: http://www.opengroup.org/projects/soa/
- Wikipedia: http://en.wikipedia.org/wiki/Service-oriented_architecture

Staples "Easy Button"
http://www.staples.com/office/supplies/moreviews?catentryId=130700&langId=-1&storeId=10001&catalogId=10051&imageClickSequence=0

Sun Tzu, *The Art of War,* 2007, BN Publishing, ISBN 9562910946

Weatherbug: http://weather.weatherbug.com/