13th ICCRTS: C2 for Complex Endeavors

A Service Oriented Architecture (SOA) Approach to Department of Defense Architecture Framework (DoDAF) Architecting

C2 Architectures

C2 Concepts, Theory, and Policy

Collaborative Technologies for Network-Centric Operations

Kevin M. Gunn
The MITRE Corporation
903 Enterprise Pkwy, Suite 200
Hampton, VA 23666
Phone: 757-896-8574
Email: kgunn@mitre.org

Fatma Dandashi
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102-7539
Phone: 703-983-7914
Email: dandashi@mitre.org

David J. Edwards
The MITRE Corporation
4830 W. Kennedy Blvd., Suite 790
Tampa, FL 33609
Phone: 813-287-9457 x356
Email: edwardsd@mitre.org

© 2007 The MITRE Corporation. ALL RIGHTS RESERVED

Approved for Public Release; Distribution Unlimited, Case #: 07-1337

A Service Oriented Architecture (SOA) Approach to Department of Defense Architecture Framework (DoDAF) Architecting

Abstract

The scenario: (1) you have an environment with some idea of the requirements to transfer information in support of decision-making; (2) you know most of the "players" who will produce and consume this information, but perhaps not all of them; (3) there are existing automated systems in place that facilitate some of the information exchanges, and some systems that are planned or assumed; and (4) you are required to invest your budget constrained funds to migrate to a service-oriented architecture while maintaining your current capabilities.

This paper starts with describing the "traditional" DoDAF approach to establish a common understanding of the problem space. We then define an approach toward guiding the infrastructure and investment plans toward a service-oriented solution.

Keywords: DoDAF, SOA, information, architecture

Introduction

"The Department of Defense (DoD) Architecture Framework (DoDAF), Version 1.0, defines a common approach for DoD architecture description development, presentation, and integration for both military mission and business operations and processes. The Framework is intended to ensure that architecture descriptions can be compared and related across organizational boundaries, including Joint and multinational boundaries." (DoDAF, 2004) In other words, the DoDAF is used to communicate architectural concepts in a common vocabulary, of sorts. But, the DoDAF was conceived before the concepts and principles of Service Oriented Architectures (SOA) became well-understood. Therefore, organizations trying to document their architectures using foresight are stuck between a framework they kind of understand and a concept they sort of grasp, but no way to connect the two sets of thoughts in their minds. When investment decisions must be made using these two, apparently un-connected concepts, a recipe for confusion is likely to emerge.

The Scenario

In support of a DoD component, the authors where faced with this scenario: (1) you have an environment with some idea of the requirements to transfer information in support of decision-making; (2) you know most of the "players" who will produce and consume this information, but perhaps not all of them; (3) there are existing automated systems in place

that facilitate some of the information exchanges, and some systems that are planned or assumed; and (4) you are required you are required to invest your budget constrained resources to migrate to a service-oriented architecture while maintaining your current capabilities.

Faced with this scenario, the authors have defined an approach to utilize DODAF to produce architecture descriptions that are service oriented. The approach described in this paper, was based on a real use case that has been generalized for wider use. We've seen other scenarios very similar to this one, so we are confident the approach can be applied to those situations, also.

The environment can be characterized as follows (Cherinka, 2007):

- **Dynamically assembled**: often integrated from existing components
- **Evolving requirements**: typically articulated as vision statements or broad architectures. Often contradictory.
- **Emergent functionality/behavior**: from the interaction of the components themselves without specific direction
- **Crosses program boundaries**: competition for resources & alternative solutions

Based on the authors' experiences with other DoD programs such as Global Command and Control System (GCCS), and NECC, these characteristics are consistent with a wide range of problem spaces, therefore making this approach applicable to a wide audience.

Approach

Our approach included four basic steps, executed over about nine months with four part-time architects and several domain-area subject matter experts (SMEs).

Educate the Customer

While DoDAF's goal (described above) is intended to provide a common architecture "vocabulary", the DOD has focused on creating *products* (OV-1, SV-2, etc.) to support required documentation vice developing a truly *integrated architecture*. This errant focus has, in the authors' opinion diluted efficacy of the DoDAF. The first step, then, is to re-educate the customer on how using a DoDAF approach can help solve their "problem" (described in the "scenario" above). We started with relating what the customer wants (a Capability—with a big "C") with what the customer *buys* (tools, systems, widgets, etc.) through a simplified diagram (see Figure 1).

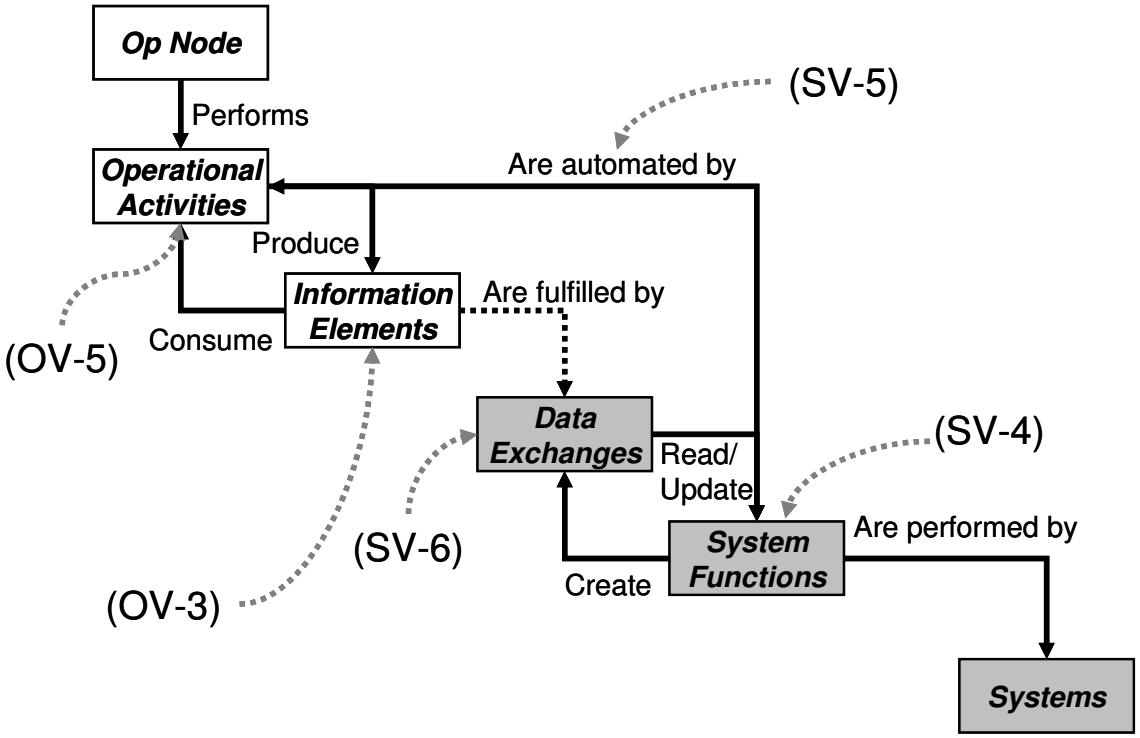

Figure 1. DoDAF-Lite

This "DoDAF-Lite" approach allowed us to describe, in accessible terms, how we can use DoDAF to define the Capabilities required and connect that to the "things" they should buy (Systems) to enable those Capabilities (see Figure 2).

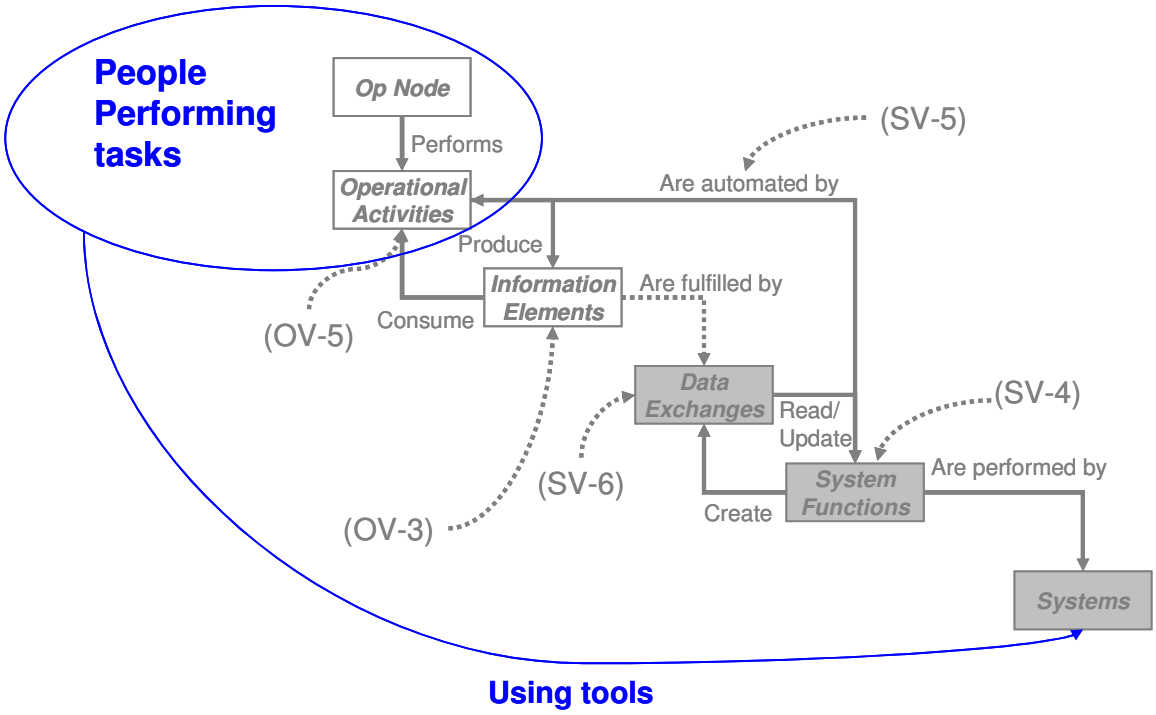

Figure 2. Capability = People + Processes + Tools

Start with What You Know

The second step in our approach began with capturing how the customer operates today, what systems they use, what information is required, etc. This is commonly called the "as-is" view using a vetted scenario (i.e., mission thread) in the problem space. That data can readily be captured in any number of architecture tools or simply collected into a database and/or set of diagrams. The key in this step is to get the customer thinking about the context of the problem set and serve as a starting point for the follow-on analysis (finding the "gaps" in the architecture). This activity allowed the architecture team to ramp-up and synergize as a team; however we concluded that it was not necessary to spend a great deal of time on this step because the value to be realized was in the following steps.

Document How Things Should Be Done

In a Department of Defense (DOD) problem space, this third step would involve going to published Doctrine and Tactics, Techniques, and Procedures (TTPs). It might also require looking at operational plans and contingency plans to get the full picture of what the mission(s) require in terms of people and information. To augment the documentation, in-depth interviews with SMEs are also required to get "pie in the sky" ideas, and common sense.

This data can also be captured in an architecture tool, a database and/or set of diagrams. This is the "to-be" view of the architecture.

Focus on the Information Flows

The fourth step in our process (which will be described in more detail in the next section) entails looking carefully at the information flows between operational nodes (the "information exchange requirements" or IERs). This activity identified IERs that would make good candidates for Services (if they are not already so enabled—which should be evident from the as-is architecture). All actions in the architecture (Operational Activities) are fed by, produced, or triggered by information (IERs). In addition, all services produce and/or consume information. It makes sense, therefore, that a close look at IERs will reveal opportunities for service development.

IERs and Services

The Organization for the Advancement of Structured Information Standards (OASIS), defines "Service" as: "The means by which the needs of a consumer are brought together with the capabilities of a provider." (OASIS, 2006). In our DoDAF model, this equates to enabling an IER. How do we do this? Well, let's start with defining exactly what we mean by IER. An IER consists, at a minimum, of:

- Source Node (Producer)
- Source Activity
- Information
- Sink Activity
- Sink Node (Consumer)

Now, a Service is really a technical solution for an IER, so it is not directly represented in any operational view. However, it *can* be reflected in the system views as a system performing a system function to produce a data exchange to satisfy the IER (see Figure 3)

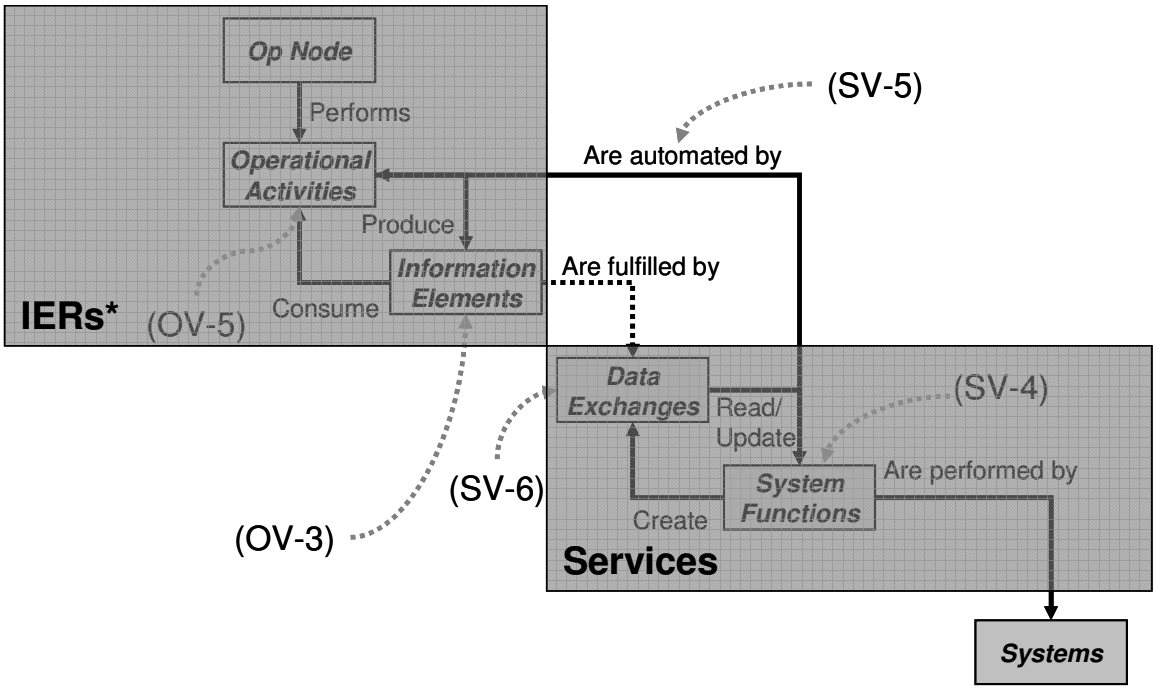

Figure 3. IERS and Services

So, in our model, we now have a working definition of a Service: Encapsulating a "System Function" and it's corresponding "Data Exchange(s)" into a visible, accessible, and understandable interface to fulfill an IER in an operational thread." In other words, it is an abstract of the underlying system and subsystems - therefore hiding its complexity.

Automating an IER—Bottom-up approach

Figure 4 illustrates a bottom-up approach to automating IERs (i.e., creating and implementing services). As the reader can see, the systems performing the system functions that automate an operational activity or produce or consume the data exchange to meet the IER, must be directly available to the operational node performing the activity (e.g., point-to-point solutions).

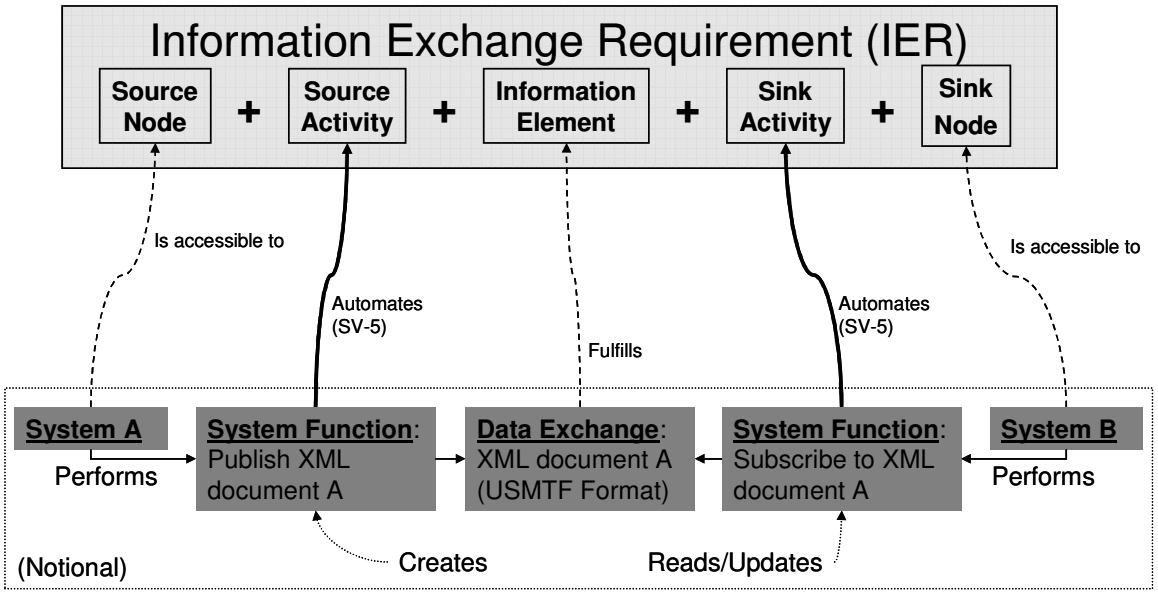

Figure 4. Automating an IER--Bottom-up

Often, however, we do not have the fidelity to fully characterize *all* of the system functions or data exchanges explicitly. This is especially true if you have a large number of systems each exchanging data. There are a potential N(N-1) interfaces between those systems with each interface containing one or more data exchanges (the "N-squared" problem).

Automating an IER—Top-down approach

If we, instead, first focus on the systems used to automate activities and/or produce data, then we can simplify the approach somewhat (see Figure 5).

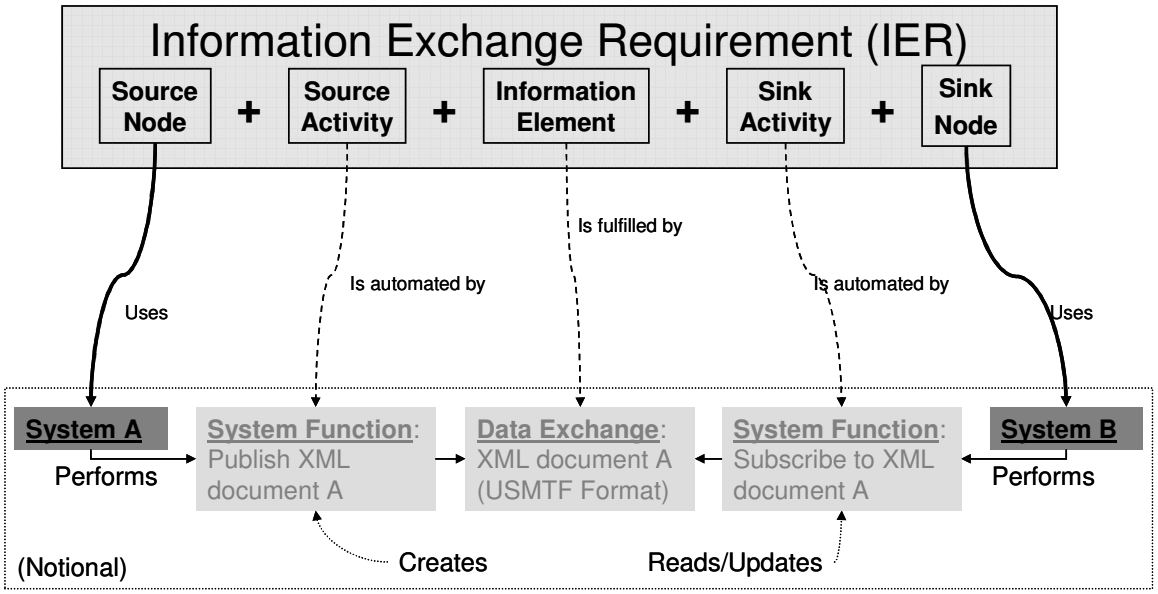

Figure 5. The Top-down approach to automating IERs

We can then break the problem down into "what does each system function *produce* and what does it *consume*?" Looking at the *Source* side of the transaction, we can identify the "Produce" (i.e., publish). Likewise, the *Sink* side represents the "Consume" (i.e., Subscribe) -see Figure 6 and Figure 7.

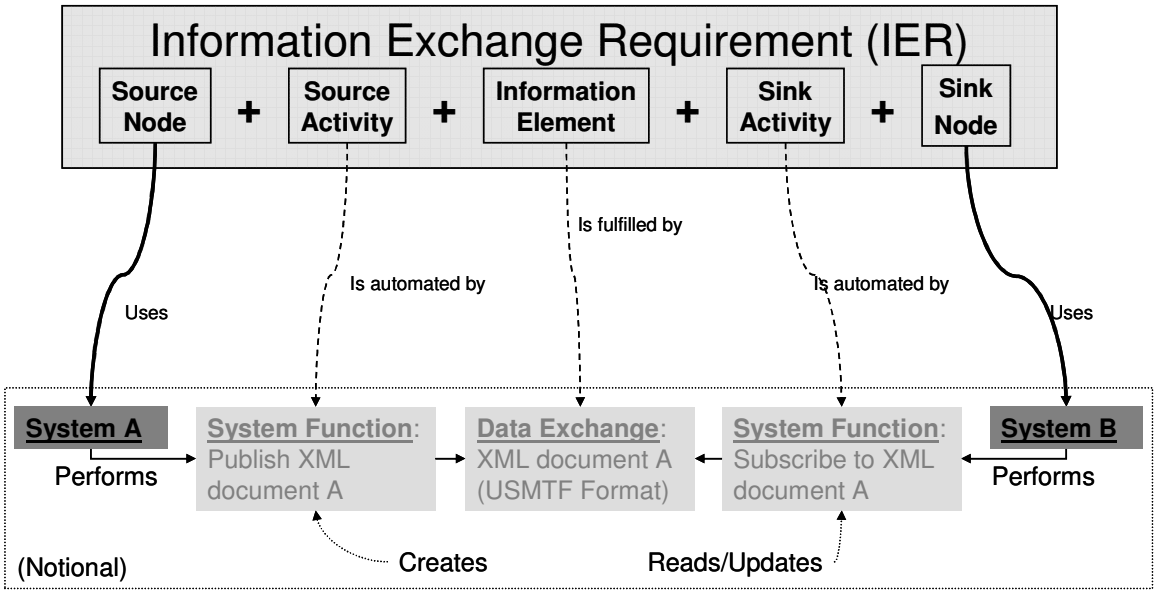

Figure 6. Publishing Services

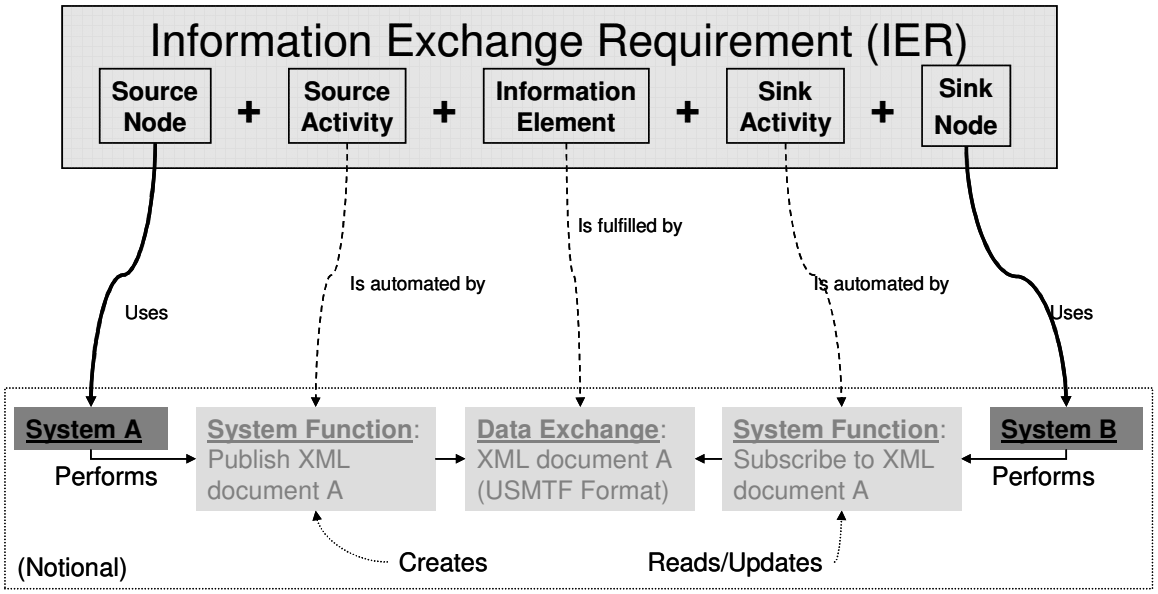

Figure 7. Subscribing to Services

Once we have broken down the problem in this way, we can start looking at how to attack it. One logical prioritization that came to mind was to look at those Information Exchanges (IE) that are consumed by multiple nodes. These exchanges would be good candidates to prototype as web services, since they are in high demand.

We then generated a list of all the unique/common Operational Activities (OA). These OAs were used to define candidate services. This is essentially a decomposition activity that is typically used in Systems Engineering. In this process we were developing "buckets" for our services that will ultimately lead to capability requirements for a SOA. Some refer to this as the Service View of the Enterprise.

Finally we normalized (i.e., removed redundancy and focused on uniqueness) these services into a common list of candidate services required for this architecture. This list of services provides the starting point for the mapping of Services to Systems and Services to OAs to show traceability of required capabilities.

Completing the Transaction

Today, a direct link is required between System A and System B to make a transaction work (see Figure 8).

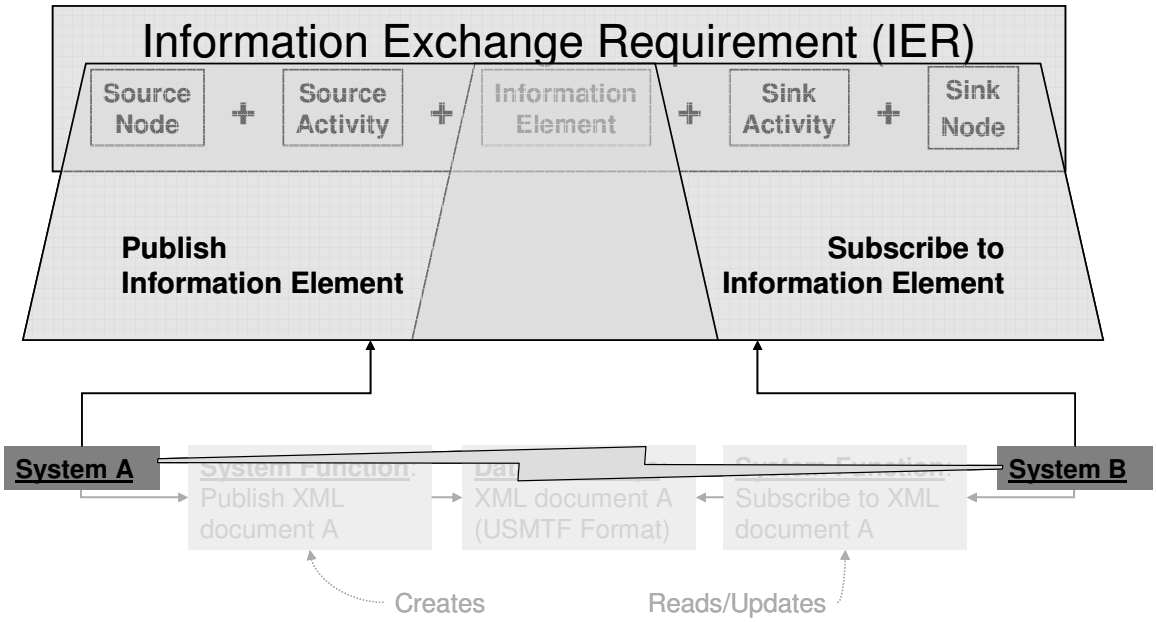

Figure 8. Completing the Transaction

In today's environment, we still have the N-squared problem—whether the system functions and data exchanges are "wrapped" into services or not. In a service-oriented environment, however, we introduce a Service Oriented Architecture (SOA) Infrastructure (e.g., Enterprise Service Bus) to broker the various exchanges (see Figure 9).

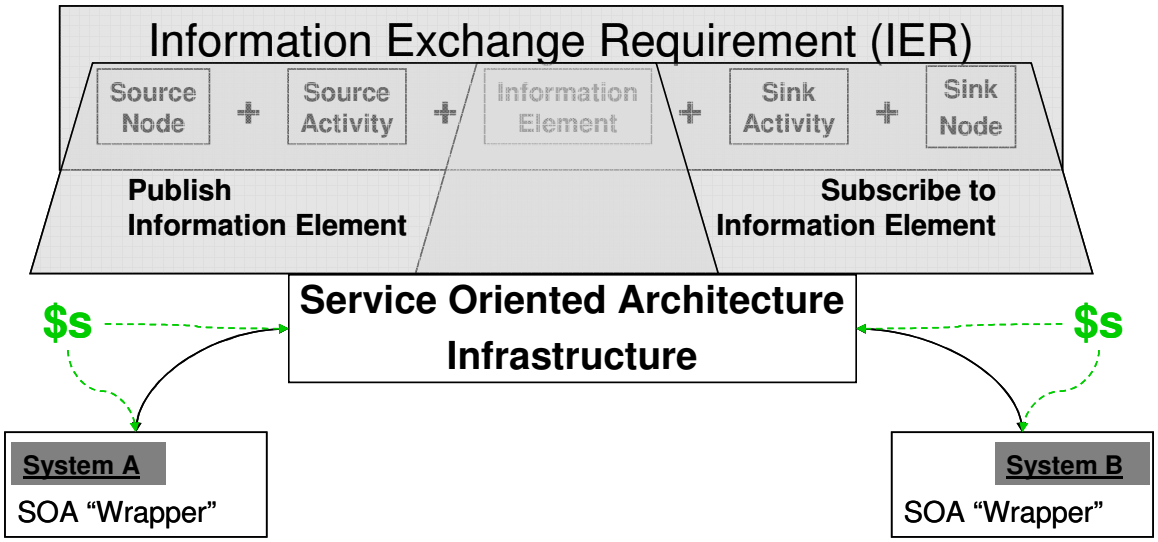

Figure 9. IERs in a SOA

A SOA infrastructure is needed to make services visible, accessible, and understandable (DOD CIO, 2007). The SOA "wrappers" enable legacy systems to publish and subscribe in the SOA. The benefit of the investment in a SOA infrastructure (and make no mistake, it takes money and commitment) enables plugging in new services (or additional "wrapped" legacy services) into the same infrastructure and to allow the infrastructure to act as the broker between service providers and consumers. Such an infrastructure can also be constructed so that queries to find available services (instead of knowing about specific providers a priori) are enabled. In this manner, legacy systems and functions can be brought into a service oriented world, while at the same time enabling new system development to proceed in a service oriented manner. This allows on the order of N-squared transactions with only an order of N interfaces to be accommodated (that between each service and the SOA infrastructure). It also leads to many other benefits such as agility to change, reusability of available services, and interoperability of data through standard interfaces, formats and protocols.

Getting to a SOA

There are many factors to take into account when planning an approach to implement a SOA to replace the current point-to-point interfaces. Among the questions to ask are:

- How much does it cost to "wrap" an existing system to provide/consume a service?
- Can a new service be developed cheaper?
- How much infrastructure is needed?
    - What hardware?
    - What software?
    - Who owns it?
    - Who maintains it?
- How do we invest in "infrastructure?"
- At what point does it make sense to replace functionality of a system with a separately maintained service?

Ultimately these questions lead to a value proposition supported by a business case to migrate the current systems to be "Service Oriented".

Conclusions

In an environment where the DoDAF has been used to define a system or operational thread, it's possible to migrate to a SOA by careful analysis and decomposition of the architecture. This paper defines an approach for using DoDAF-based architecture descriptions to guide infrastructure and investment plans toward a service-oriented solution. The approach proposed in this paper provides a number of benefits:

- A generalized approach to describe operational environments such that one can look at existing environments and identify useful candidates for services.
- With requirements, analysis, and specification being one of the hardest steps in designing a system and specifically defining an architecture, this approach helps in all these steps by establishing a starting point and providing a path in succeeding with defining services on valid requirements.
- A guide for infrastructure investment.

Even when many of the details of the environment are unknown (or undocumented), the operational architecture can be "converted" into a SOA implementation. Essentially, the list of candidate services is a list of required capabilities that could be enabled with suitable investment in a SOA infrastructure. While not an inexpensive endeavor, it is preferable (in the authors' opinions) to the continued investment in "N-squared" point-to-point interfaces in a rapidly changing environment.

References

DoDAF 2004 DoD Architecture Framework Working Group, DoD Architecture Framework Version 1.0, 9 February 2004.

Cherinka, R. and Miller, R. 2007. "Engineering a Complex Information Enterprise: A Case Study Architecting a Department of Defense (DOD) Middle-Out Hourglass." Paper presented at 2007 International Conference on Enterprise Information Systems and Web Technologies (EISWT-07), 9-12 July, 2007, in Orlando, FL.

OASIS. 2006. Reference Model for Service Oriented Architecture 1.0, 2 Aug 2006 (www.oasis-open.org)

DoD CIO. 2007 Department of Defense, Chief Information Officer, Net-Centric Services Strategy--Strategy for a Net-Centric, Service Oriented DOD Enterprise, May 4, 2007.