# Global Information Assurance Certification Paper

# Using the Department of Defense Architecture Framework to Develop Security Requirements

*GIAC (GSEC) Gold Certification*

Author: James E. A. Richards, james.richards1@gmail.com
Advisor: Christopher W. Walker

Accepted: February 7, 2014

Abstract

Requirements for security in an organization or enterprise that uses information technology can be difficult to develop, given the complex organizational policies, technologies and processes that affect each requirement. Integrated architectures can serve as a vehicle to bring order to this complexity, and if constructed using a common data model, can be reused when developing future requirements. This paper outlines a basic approach to using an integrated architecture, built using the Department of Defense Architecture Framework (DODAF), to derive security requirements. It also examines a case study that illustrates the potential use of these techniques, and provides an overview of the relevant portions of DODAF.

# 1. Introduction to Integrated Architecture

Integrated architectures embody the discernable parts of a system and their relationships with each other in a single, normalized data repository. It takes some method, rigorous thought, imagination and art to choose relevant data to illustrate aspects of the system in a meaningful way. These are the foundational concepts that make integrated architecture a versatile discipline that has the potential to bring the unmanageable amount of information that typically goes into an enterprise or a system under control. It allows stakeholders to wield those insights while making decisions, expose unaddressed issues, and even derive security requirements. Once built and maintained, an integrated architecture can even support or serve as the basis for future projects that require a holistic analysis of that enterprise or system.

The DOD Architecture Framework (DODAF) is the US Department of Defense's customized framework for integrated architecture. As an architecture framework, it describes the rules and conventions that DOD architects use when building integrated architectures so that the resulting work is compatible, at least at a basic level. DODAF was created to support many facets of the DOD, and was intended to be useful even when just the parts of the framework that pertain to a given purpose or project are implemented. One of the first steps in building an integrated architecture under many frameworks is to determine the scope of the effort, which helps determine which parts of the framework are needed to succeed. An integrated architecture that contains the details of an organization, system, or enterprise serves as a normalized model that can be analyzed to show the linkages or connections between elements in the real world. These linkages can be many degrees deep, showing second-order and tertiary connections in a repeatable and reusable way. Investing initial effort to build a model of an enterprise, then using and reusing that model to avoid performing the same analysis on the same elements of that enterprise, is the focus of the techniques singled out from the DODAF here. Put another way, it could refine the situation in Figure 1 to the situation in Figure 2:

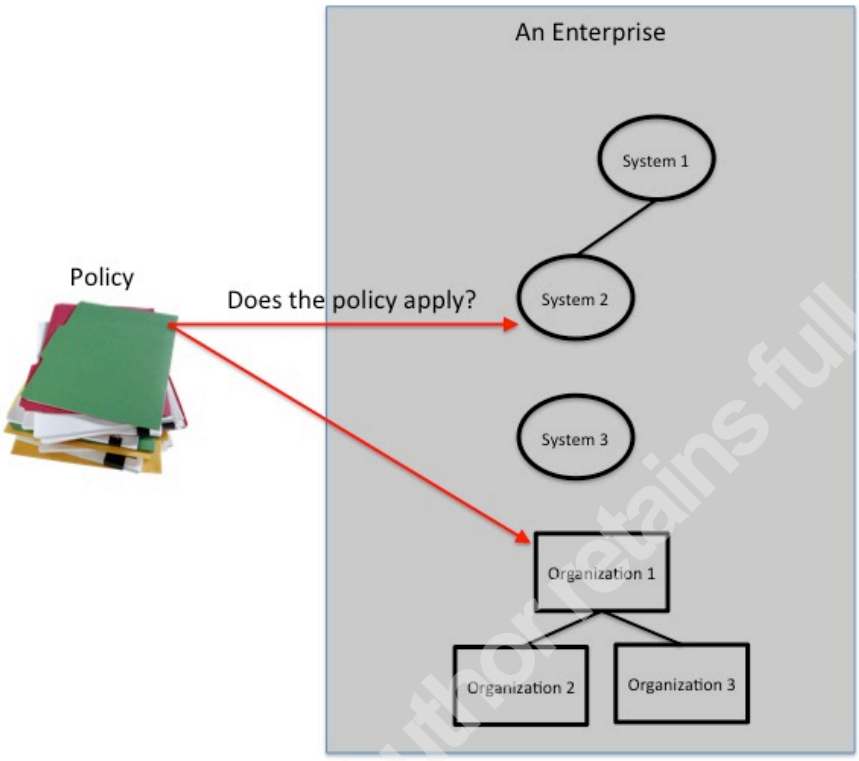


Figure 1: Trying to apply a policy without the aid of integrated architecture

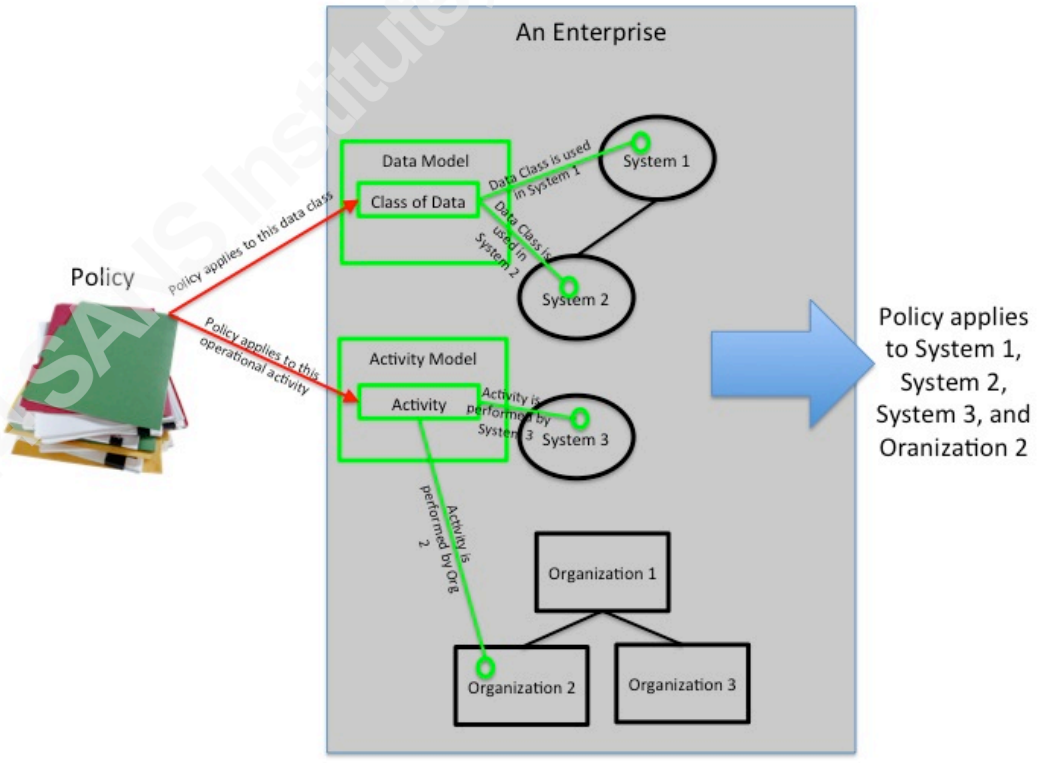


Figure 2: Derive policy application to enterprise by using integrated architecture

James E. A. Richards, james.richards1@gmail.com

## 1.1. The Department of Defense Architecture Framework (DODAF)

The DODAF is the Enterprise Architecture framework that the Department of Defense (DOD) uses for capability development and acquisition. Its roots derive from the C4ISR[1] Architecture Framework, which was renamed and extended to produce DODAF version 1.0, then further back to Zachman's more general work on the same topic, which is discussed in Appendix A (Sowell, 2000). The legislative basis for the use of such an architecture framework by the DOD comes from the Clinger-Cohen Act, with specific guidance in the US Office of Budget and Management Circular (OMB) A-130. As Kathie Sowell points out in her overview of the C4ISR Architecture Framework's history, a common architecture framework is necessary to synchronize and integrate the efforts of multiple organizations independently doing capability development, and by extension, architecture development. The benefit to the government of integration through a common framework here is balanced by the deliberate lack of rigorous adherence to a single methodology or standard.

When using integrated architectures such as those that conform to DODAF, it is important to keep in mind the purpose behind building them: rigorous and consistent linkage of complex information. Throughout all of its revisions, DODAF's purpose has been to provide a normalized means of describing systems or capabilities that the DOD has bought or built to ensure the next thing it buys or builds interoperates in the intended way with what is already in the inventory, and that the second and third order effects of its relationships with those systems are taken into account. Rigor, consistency, and leveraging existing analysis makes architectures not just suitable for building weapons systems, but for deriving security requirements as well.

Since being renamed to DODAF, three versions of the framework have been published:

| Version | Changes |
|---|---|
| DODAF 1.0 | DODAF version 1.0 inherited a system of four "views" or classes of architecture data presentation products from the C4ISR Architecture Framework, along with the Core Architecture Data Model (CADM), |

[1] C4ISR is a military acronym for "Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance".

| | |
|---|---|
| | a common data model to facilitate integration. |
| DODAF 1.5 | DODAF version 1.5 introduced modifications designed to address the decoupling of technology, such as the use of services, and a simplification of the data model. |
| DODAF 2.0 | DODAF version 2.0 introduced additional reductions in explicit requirements in favor of a focus on a data-centric approach and minimalist rendering of the data on views. It also retired the CADM in favor of the DODAF Conceptual Data Model (CDM), which achieves integration through the specification of physical data transfer rather than specifying a data model intended to be used as a common repository. |

Table 1: Versions of DODAF

Versions 1.0 and 1.5 of the framework are described in three static volumes, and version 2.0.2 of DODAF is available at: http://dodcio.defense.gov/dodaf20. Versions since 1.5 are updated more frequently, but can still be downloaded as a single PDF from the website above.

## 2. Basic Technique to Develop Security Requirements from Integrated Architectures Using DODAF

Using DODAF to derive security requirements is an integral part of why the framework was created and mandated. OMB Circular A-130 explicitly requires that security guidance from the National Institute of Standards and Technology (NIST) be incorporated into enterprise architectures, among other requirements that involve integrating security requirements alongside the rest of the information in an architecture. It is the incorporation of the security requirements into the *same* integrated architecture and the linking of those requirements to other elements in that architecture that leverages it to develop security requirements for the enterprise.

While it may seem like this method merely identifies security requirements that already exist to be applicable to elements in an architecture, populating an architecture has the potential to expose implied and unforeseen security requirements through second-order and tertiary linkages. Assembling information into an integrated architecture also organizes it in a way that makes it easier to manipulate, since it is normalized. By building an architecture that represents the enterprise you are interested in, you are also investing in the future by being able to extend linkages from future general analyses to

elements throughout the architecture. Put another way, the utility of an architecture that is built with analytic rigor lies both with its initial use and with reuse over time. A discussion of the significant methodologies for modeling an enterprise that are useful in DODAF is in Section 7; readers who are not familiar with DODAF or integrated architecture in general may find it useful to read that section before the rest of the paper.

## 2.1. Model Your Enterprise Using Integrated Architecture

The first step in developing security requirements from architecture is to build the architecture so that it models your enterprise. Scope is an important issue at this point, and a good way to select an appropriate scope is to work backwards from the views or information products that will be needed at the end to the information on hand now. Modeling the entire enterprise is not necessary to divine its security requirements, but modeling the things you want to protect, the policies that constrain your ability to protect, and the resources you can use to protect are an excellent core set of pieces to model. For example, if the information you need is a view that shows all personnel who need specialized security training (i.e. HIPAA), the critical information product to produce is a list of personnel and the training each one needs. One possible critical path through an integrated architecture to produce this would be: determine what specialized training anyone in the enterprise would need, link the policy requiring the specialized training to the applicable elements in the enterprise, link those elements to operational activities, link those activities to specific roles or organizations, and finally, link personnel to a role or organization. Those linkages are only possible if you have an accounting of training policies, a model of the elements to which they link (potentially OV-2, OV-3, or OV-7) an activity model (OV-5), and a model of roles or organizations (OV-4).

Building certain core views is important in DODAF to enable creating linkages between the Operational Views and the System Views (plus other additional implementation-level views in DODAF 2.0) and thus, building them first is highly recommended. Specifically, start with the OV-5 Operational Activity Model, the OV-7 Logical Data Model and the OV-3 Operational Information Exchange Matrix to facilitate the modeling of an enterprise's processes (OV-5), data and business rules (OV-7), and communications (OV-3). These are only a start; most enterprises will require more views

to create a basic integrated architecture that adequately represents the enterprise, but with these three views complete, building the rest of the views in the DODAF taxonomy is largely derivative, since the elements in the OV-5, OV-7, and OV-3 permeate throughout the rest of the OVs. Also, the OVs model the enterprise as a whole, and the SVs typically model information system implementations, so building the OVs first provides a standardized basis for efforts to model the information systems which presumably automate a portion of what is depicted in the Operational Views. Likewise, the "core" linkages explained above and DODAF's other common linkages are depicted well in DODAF 1.5, Volume II, Section 7, Figure 7-1, pictured below in Figure 3 (DOD CIO, 2007).

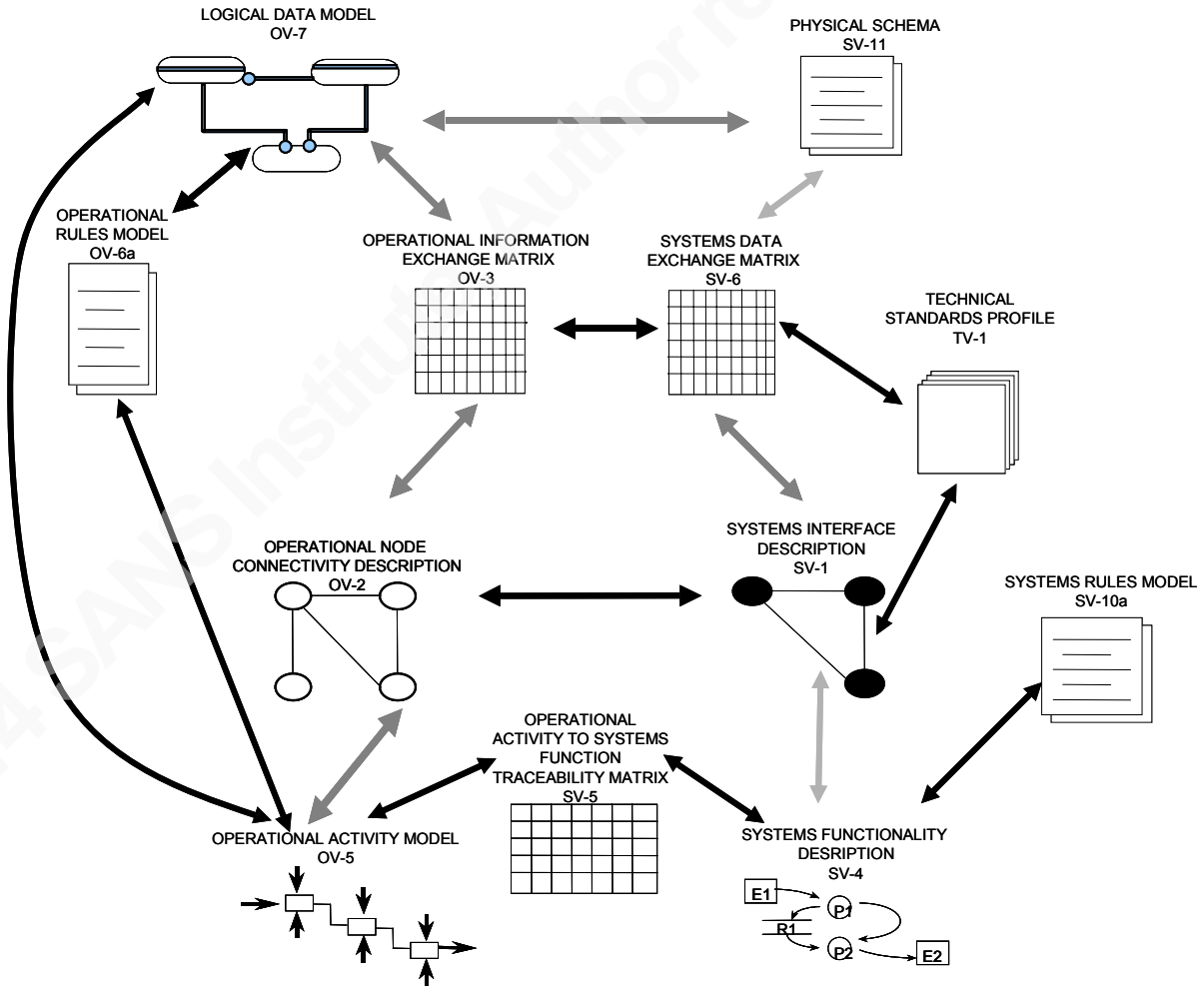


Figure 3: Linkages between DODAF Views (DOD CIO, 2007).

Modeling itself is a series of iterative projects and by design should not be accomplished in a "single pass". Considering the example of an Army unit from section 7.4; if Figure 7 was an initial effort to model the unit, the next step would be to relate the elements in Figure 7 to other elements through the use of a matrix or the construction of another diagram with some of the same elements. Subsequent diagrams or models do not always need to contain or reuse an element that already exists, but reuse of elements is a good method for discovering elements that are missing from the model. As an example, in the IDEF0 Activity Modeling methodology, each of the activities can be decomposed by creating a new diagram that has the same ICOM arrows "entering" the diagram (i.e. coming from somewhere not depicted on the diagram) and "exiting" the diagram (i.e. pointing to nothing). The iterative nature of modeling comes from elements added to the architecture that result in the discovery of additional information that should be added to views already constructed.

Regardless of the approach used, the task is essentially to accumulate the necessary information, normalize it to conform to the architecture framework, and add it to the "integrated dictionary" or architecture repository. In this case, "normalize" means that the elements you add to the architecture conform to the underlying data model. A good analogy is entering an address into a particular application that tracks your contacts – the application handles addresses, but it may require you to classify part of the address as the "street address", "postal code", and "state", which may not work for an address in a country without postal codes or perhaps without subordinate states. Extending this to more complicated things, as elements of an enterprise can be, may seem overly dogmatic but doing so introduces the analytic rigor that gives architecture its strength.

The type of architecture element that this process can be centered on depends on what you are trying to model; if you best know what an organization does you could reasonably start with activity modeling and use it as a basis for collecting data from experts or references. If the most authoritative information to model is about information or data, a reasonable start would be in data modeling.

Data-centric approaches begin with the data model, using it to guide construction of the activity model and information exchanges. Activity-centric approaches naturally start with either the activity model or other sets of data that specify and constrain the

enterprise similarly, such as the information found in the OV-6 series of views. The important part of this step is to start with the information available and ensure integrity and validity of the data that is being entered into the architecture. DODAF 1.5, Volume I, Section 2 provides a good treatment of how to build an architecture, and Section 3 describes the habitual uses for it, to include a matrix that relates common uses to views (Figure 3-1 in the DODAF 1.5, Volume I) (DOD CIO, 2007). As an example, the following figure from page 2-2 describes the general six-step process for building an integrated architecture, but there are some steps that are specific to DOD and may not apply to all enterprises:

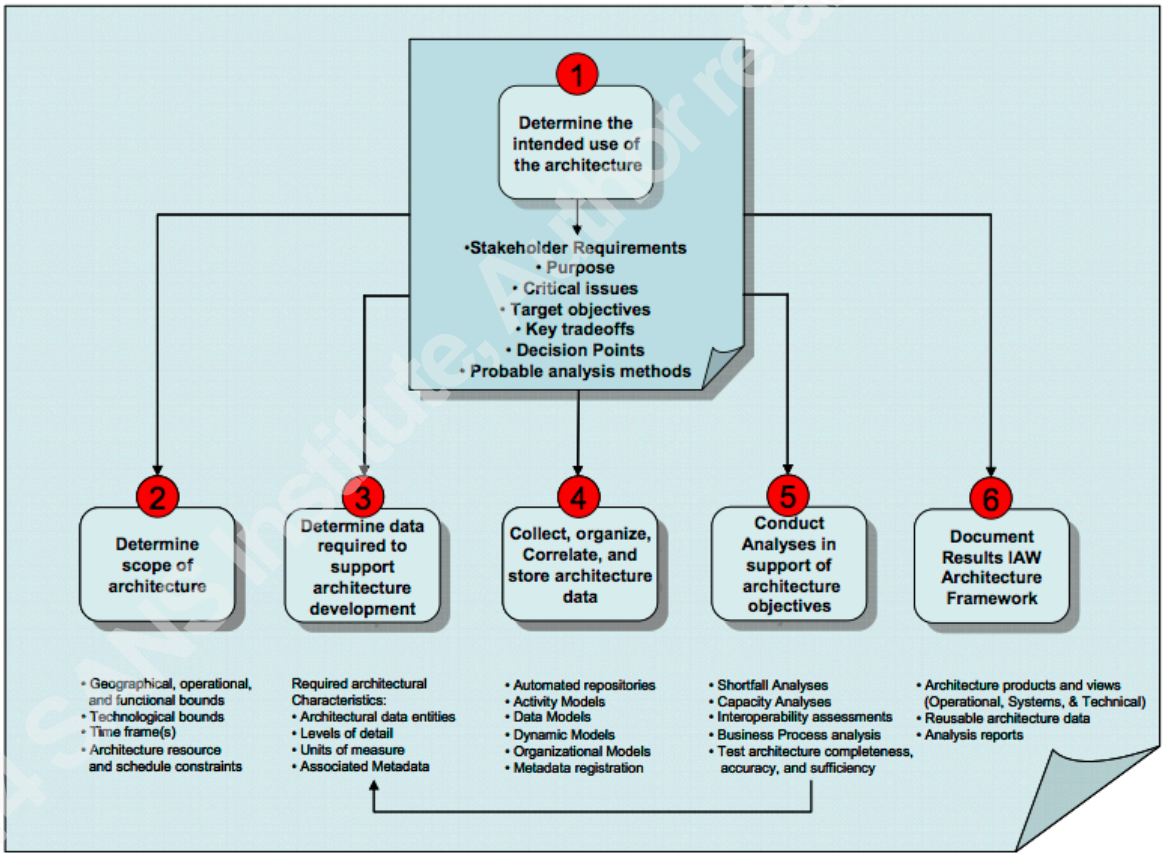


Figure 4: The six step architecture development process from DODAF 1.5 (DOD CIO, 2007).

Although it is inherent in the modeling of the enterprise, all known security policies, standards, and other requirements should be added to the integrated architecture. There are a few places where this is common:

- Technical View (TV) 1 for DODAF 1.0 and 1.5, and Standards Viewpoint (StdV) 1 for DODAF 2.0 that are the "Standards Profile" view for the integrated architecture. This view is essentially a list of applicable standards (i.e. "FIPS 140-2"), that is mapped to specific elements in DODAF 2.0.
- As Control ICOM arrows, guard conditions, or the equivalent in the activity model or OV-5.
- In the specification of information exchanges (OV-3) or system data exchanges (SV-6).

One of the pitfalls of this phase of architecture development is to ignore gaps in data that are critical to establishing the linkages that make it useful. There are many reasons why an architect has trouble finding data, but whether the reason is political or practical, the simple truth is that the completeness of the analysis you get from an architecture depends on the completeness of the data you put into it. In some cases, it is not a lack of information that prevents the addition of something to the architecture, it is the existence of more than one version of it, such as a "to-be" and an "as-is" version or multiple hypothetical versions of an element being considered for acquisition. However, each version can be added to the same integrated architecture as separate elements to retain both, and various solutions exist to maintain the integrity of the analysis that results. For instance, views can be tailored to show both side by side or several views can show each one by itself in the same context to highlight the differences.

## 2.2. Choose Views to Validate Compliance

Once you have built an architecture, it is time to extract the information a few dimensions at a time (i.e. system data exchanges, system interfaces, and system nodes), using views to illustrate the collected relationships between elements, then audit the consistency of what was rendered. This is not unlike a barber checking for evenness in the length of hair over a customer's head by grabbing small locks at a time and comparing neighboring hairs. Views may have been the method of collecting information from the enterprise, since an appropriate view is an intuitive vehicle for representing the information collected as it is being collected, but they are also the best way to show the

sum of what has been collected. Presumably, some measure of effort is made each time information is added to the architecture to prevent erroneous information from being added, and in doing so, it might also seem logical that the process would ensure consistency throughout the architecture as well. However, it is the consistency between elements that are added at separate times that this step checks, leveraging the independent additions of information to identify conflicts, gaps, and where there are no issues.

The set of appropriate views for a given purpose is a critical question here. The simplest way is for it to be prescribed, as is the case with some stages of the DOD acquisition system, where there is a standardized document required by regulation at each stage and architecture views required by regulation for the document. Another way is to ask a question, as implied by the interrogatives matrix in Table 4, and determine which views could be used to answer the question. For instance, the question: "How many different kinds of reports will the system generate?" would probably be answered by the SV-6 (System Data Exchanges), with some filtering to remove system data exchanges that are purely internal. Alternatively, an architect could start with the SV-5 (Mapping of System Functions in the SV-4 to Operational Activities in the OV-5) and derive which system functions are mapped to operational activities that produce a report. Though the SV-6 appears to be a simpler view to use, both methods are valid and the second one may narrow the search more quickly. The linkages between views really represent the linkages between elements in one view and elements in another, and they facilitate answering a question, such as the one about reports, when simply listing the contents of the integrated dictionary are not sufficient. If the architect knows about the set of reports an organization generates from the activity model, getting a list of systems that generate those reports requires following the linkages within the architecture from the activity model to the model of system functions, or OV-5 to SV-5 to SV-4. This and other linkages are depicted well in DODAF 1.5, Volume II, Section 7, Figure 7-1, pictured in Figure 3 (section 2.1) (DOD CIO, 2007).

For security requirements, there are a couple of places in particular to use as sources of validating information. The practical application of security naturally requires tailoring of general concepts to the specifics of a system or enterprise, and this is a

primary thing to check: does the architecture show that that the system or enterprise is compliant with its own security standards? The Technical Views (TV) and Standards Views (StdV) are where a complete architecture will list the generalized and tailored standards, and each aspect of the standards is possible to check in a pass of a view. A simple example would be checking for system functions that map to actions that require privileged access according to policy that also map to roles that are not intended to have such privileged access. Another excellent place to look for security requirements, if the views have been developed, is in either the OV-6c or SV-10b views, since they show sequence. A lot of the methodologies used to build integrated architecture views are designed to be agnostic of sequence, which can be counterintuitive when looking at one; it means that an activity depicted on the left of an OV-5 activity diagram does not necessarily imply it is executed before one that is farther to the right. For the OV-6c and SV-10b, sequence is the focus of the view, and examining a fully populated architecture can reveal disconnected sequences that introduce security flaws.

When inconsistencies, disconnects, and other findings are identified, this process has succeeded in focusing attention on areas where existing security requirements may not have been met or where previously unknown security requirements are located. Though the goals of integrated architecture are broader than that, analyzing these findings from a security perspective is an easy step to overlook and is not mutually exclusive of looking at the findings from a technical or operational perspective. The key is to link the known security requirements into the part of the architecture where it is intuitive to do so, then trace the linkages to other elements in the architecture to discover the impact.

## 2.3. Assess and Refine Architecture

Examination and analysis of the architecture, especially if involves linking policy or security requirements and tracing the impacts as described above, may reveal changes that have to be made to the enterprise. These changes should also be reflected in the architecture, and the resulting architecture should be validated to discover if the new elements and linkages are what is intended. With unlimited time and resources, and the patience of a computer, an architect could repeat this cycle ad infinitum and address all findings until a cycle revealed none. That is a potentially open-ended effort, though with

practical bounds that are set in advance, the repetition could be limited to a reasonable number of iterations.

It is also noteworthy that this is where an automated tool to develop architecture really pays dividends. The most straightforward benefit is that it is more efficient to change something in an integrated architecture once and have the change automatically propagate to everywhere the changed element is used than to do it manually. The reduction of error from repeatedly applying the same change to multiple models, views and elements can have just as big an impact, however, especially for a big architecture. The bottom line is that in the case of rote tasks like this, automation is best.

## 2.4. Using the Resulting Architecture To Support Specialized Security Projects

Investing in the creation of an integrated architecture using DODAF has a lot of potential value to an enterprise on its own but it can also serve as the core of other efforts to manage the enterprise as well, including those focused on security. As a basic accounting of the enterprise itself, the AV-2 Integrated Dictionary is a rich resource for techniques that require detailed information about the enterprise in order to be effective. Though DODAF incorporates security requirements generically, leveraging to support a more specialized security effort can combine the benefits and realize more than either the architecture or the other security effort could alone. As examples, a DODAF integrated architecture could support Real-Time Adaptive Security or a Sherwood Applied Business Security Architecture (SABSA) effort (http://www.sabsa.org).

Dave Shackleford describes Real-Time Adaptive Security as the practice of expanding the set of information used by network security infrastructure to include contextual information to characterize events (Shackleford, 2008). This contextual information includes information about baselines of behavior for users and network endpoints as well as current and historical operational metadata (such as vulnerability scan results and network traffic statistics). With the additional information about the network, an automated sensor has more a priori knowledge with which to characterize anomalies and has the potential to focus analysts' attention more accurately with alerts.

Ultimately, to make Real-Time Adaptive Security work, an organization must also continuously "tune", or monitor, assess, and improve its security infrastructure.

An integrated architecture has the potential to help immensely with the tuning part of Real-Time Adaptive Security. Using the right Operational, Systems, and Services Views, you can model the elements of a network relevant to information assurance, such as critical nodes, the ports expected to be open on them, the services behind those ports, and nodes expected to be consumers of the services (Hamilton, 2006). Such a model probably exists in disjoint pieces throughout a typical organization, with the hardware, operating systems, and software described in the IT department and the knowledge of which services must be provided residing in the operations departments. Beyond modeling just the network itself, an integrated architecture can model people, processes, data, information exchanges, and other aspects of an enterprise that the information assurance-based architecture would not contain, but that could be linked to it through normalization and entry into the same integrated architecture.

An architecture that integrates the model of the network with the model of the business processes and the people in the enterprise can be leveraged to tune an implementation of Real-Time Adaptive Security in many ways. One potential method would be to identify the minimal set of information exchanges required to fulfill the organization's mission, then use a strict whitelist approach to securing those elements in a time of crisis, such as initiating point to point IPSec for all the identified connections and ceasing all other connections. A less extreme example might be assigning an increased severity to IDS alerts that are associated with that minimal set. In addition to this analysis, as the enterprise changes, the benefits of reusing the architecture are realized, since the same views that provided the minimal set of critical elements can be refreshed and provide the new set of critical elements to protect, if it has changed.

Basing the underlying information for Real-Time Adaptive Security as described is an example of aligning a DODAF architecture at a systems level, but DODAF could also support a SABSA implementation that would align at an enterprise level. The SABSA architecture methodology provides a way to manage operational risk through comprehensive rigor in enterprise design and specification, culminating in a holistic

tailored security solution for an enterprise (Sherwood, 2009). In SABSA, the "assets" of an organization are described using a customizable taxonomy of attributes called the SABSA Business Attributes Profile, and classic enterprise architecture techniques are used to model the business itself (Sherwood, 2009). This facilitates security requirements analysis that is focused on the elements of the enterprise that need protecting but that is done in a way that both complements the business and minimizes negative effects (Sherwood, 2009).

The generic linking of security requirements to a DODAF architecture is taken to the next level with SABSA by conceiving or deriving those requirements as the architecture and the security components of it are created. The principal difference is that a complementary and effective enterprise security design is a mandatory and primary objective in a SABSA architecture, but it just one of many competing, optional objectives in DODAF. As both SABSA and DODAF trace back to Zachman's original work on integrated architectures, mappings similar to those done by The Open Group that map The Open Group Architecture Framework (TOGAF) views and elements to their counterparts in SABSA are also possible (Sluiter, 2011). More practically, a DODAF architecture can serve as a good starting point to begin a SABSA architecture effort, and following the SABSA methodology will yield excellent input to enrich the DODAF architecture in turn.

# 3. Case Study: Medical Communications for Combat Casualty Care (MC4)

## 3.1. Overview of MC4

MC4 is a suite of software and hardware originally developed by the US Army to perform automated medical documentation in tactical environments. Since its inception in 1999, it has expanded the automated functions it provides to include more modern medical record requirements, provision of care via teleconference, and even extending record collection down to handheld devices that synchronize with a laptop. The basic system design consists of a laptop, handheld Personal Data Assistant (PDA) style devices, and several software programs, not all of which are directly interoperable with

each other. The hardware is almost all commercially produced, while the software is mostly custom-built for MC4. It is an ideal example of a system that would benefit from using DODAF through its acquisition lifecycle, both for security requirements development and for general integration, because of the disparate sources of the system's components.

Details about the hardware and software components of MC4 can be found on the program's website: http://www.mc4.army.mil.

## 3.2. A Use For Integrated Architecture With MC4

Theoretically, an integrated architecture could be built for MC4 that traces to the DOD's Business Enterprise Architecture or to the strategic level of DOD strategy and policy by referencing and using policies and other architecture efforts within itself, such as through constraints in its OV-5 Activity Model and the reuse of definitions. Though this does not result in complete integration, most of the external sources it references are not specific enough to build a stronger linkage. For instance, referencing a common policy within the security domain, DOD Instruction (DODI) 8570.01, would apply constraints to qualifying roles within the architecture, but it would not be of much more benefit to model DODI 8570.01 and incorporate that model into the architecture. The linkage that exists through the constraints on operational activities is very important, however, and is the subject of this case study (DCMO, 2013).

With this integrated architecture built, two requirements from policy would trace down to every system administrator of the operating systems on the hardware that comprise the system: HIPAA compliance training and Information Assurance certification. The former is the implementation of HIPAA compliance at the user level in the DOD that amounts to an online course once a year that takes a couple of hours to complete. The latter is the set of training and commercial certifications that individuals with "privileged" access to DOD information systems must successfully obtain, with "privileged" referring to the level of access afforded to system administrators. To find the traceability quickly, a search through the BEA AV-2 Integrated Dictionary at http://dcmo.defense.gov/products-and-services/business-enterprise-architecture/10.0/classic/products/av-2.pdf, that contains the definition for each element

in the architecture, for "8570" and "HIPAA" would yield the operational activities that are constrained by these policies. The theoretical MC4 architecture would either link operational activities to the BEA operational activities using ICOMs or reusing the BEA operational activity itself, or it would map system functions to this activity with the appropriate entries in the SV-5 System Function to Operational Activity Matrix. Depending on the level of detail in the architecture, direct traceability to system entities in the SV-1 Systems Interface Description through the SV-5 may or may not be possible. These two requirements are each directed at one of two roles for the MC4 system, namely the system administrator who maintains the system and the medical provider who operates it.

Having established this linkage through the architecture, it is clear what either a system administrator or a medical provider must do in order to function in their intended roles for the system. However, under the reasonable assumption that these two roles are not held by the same person the architecture would not depict what to do if a single person is both a system administrator and a medical provider. It is possible to depict an operator and a system administrator on the system views[2], and if such a role were envisioned by the MC4 designers, it would be possible to depict a hybrid system administrator/medical provider operator as well. The intended use of the system, though, is to have two separate individuals perform these roles. Despite the design and intended use, the MC4 program does not participate in the local operation of the system after it is delivered to the units that use it. With respect to the question of roles, a unit could select the same individual to perform both a system administrator and a medical provider role. During the 2011 fielding to 2nd Heavy Brigade Combat Team (HBCT), 1st Armored Division (2/1 AD), that is what the unit chose to do[3].

From personal experience during the fielding of the MC4 system to 2/1 AD, I know that its decision to combine the roles came out of necessity; there were not enough system administrators to fully support the number of MC4 systems that were fielded to

[2] The addition of the operator could be as an external element, which would be depicted outside the "boundary of the system, or it could be as a component of the system. Depending on the architecture approach used, either way is appropriate as long as it is consistently used.
[3] This is based on the personal experience of the author in that unit during these events.

the unit. Moreover, the system administrators that were present would not always be close to an MC4, due to the way they were employed by the unit. Medics, on the other hand, would always be present when an MC4 was present, since they were the primary users of the system and they were the ones who physically accounted for them. 2/1 AD sent medics through the MC4 training that instructed system administrators how to perform required maintenance tasks, and began using them in that role. Unfortunately, the medics put in that role were not compliant with DODI 8570.01 requirements for individuals with privileged access to systems, specifically since they lacked commercial certifications, required information assurance training, and either several years of experience in systems administration or supervision by someone with that experience. As an organization focused on medical operations requirements, the MC4 program was diligent in ensuring that all system administrator personnel received HIPAA training, that is required for anyone with access to medical records and that one would have without restricted access to an MC4 laptop or handheld device. The program did not, however, ensure medical providers being trained to be system administrators had the appropriate information assurance credentials. The enforcement of those policies actually falls to the unit that did not enforce them either, but may have done so if it was identified early enough during the initial fielding of the system for them to prepare their medical providers to also be system administrators.

In this case, an architecture could help expose requirements like the DODI 8570.01 requirements for the medical providers as system administrators after observations of units like 2/1 AD were incorporated into the architecture, and the hybrid system administrator/medical provider was added as a system entity or role. Though a conclusion like this may be possible to reach without adding the new dual-role back into the architecture, if it were a more complex combination of elements, adding the new information to the architecture would be critical to ensure nothing was missed in analyzing the existing linkages and their relationship to the new elements. With anything more than the discovery of a simple new role or similar element, the investment in the theoretical MC4 architecture would have paid off.

## 4. Conclusion

Integrated architectures based on DODAF can be rigorous, comprehensive tools to organize complex systems and expose or derive security requirements, or even just confirm that they are being covered. However, the impact they can have in developing this knowledge depends on the rigor and effort that goes into their construction. Also, it is useful to build one preemptively if an organization or system serves as a source of requirements or constraints, so that an integrated architecture for new components or systems can reference it and leverage existing work while simultaneously validating requirements from both perspectives.

## 5. References

DOD Deputy Chief Information Officer (DCIO) (2013). DoD Architecture Framework Version 2.02 [Website Map]. Retrieved April 21st, 2013 from: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf

DOD Chief Information Officer (CIO) (2007). DoD Architecture Framework Version 1.5, Volume I: Definitions and Guidelines. Retrieved April 21st, 2013 from: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_Volume_I.pdf

Hamilton, J. A. (2006). DoDAF-Based Information Assurance Architectures. *CrossTalk: The Journal of Defense Software Engineering, 19(2),* 4-7. Retrieved December 12th, 2013 from http://cross5talk2.squarespace.com/storage/issue-archives/2006/200602/200602-Hamilton.pdf.

Knowledge Based Systems, Inc. (2013) IDEF1x Data Modeling Method [Informational Webpage]. Retrieved October 8th, 2013 from http://www.idef.com/IDEF1x.htm

Knowledge Based Systems, Inc. (2013) IDEF0 Function Modeling Method [Informational Webpage]. Retrieved October 8th, 2013 from http://www.idef.com/IDEF0.htm

Office of the Deputy Chief Management Officer (DCMO). (2013) Business Enterprise Architecture [Informational Webpage]. Retrieved October 8th, 2013 http://dcmo.defense.gov/products-and-services/business-enterprise-architecture/10.0/classic/index.htm

James E. A. Richards, james.richards1@gmail.com

Office of Management and Budget (2000). "Circular No. A-130, Revised, (Transmittal Memorandum No. 4)". Retrieved October 8th, 2013 from http://www.whitehouse.gov/sites/default/files/omb/assets/omb/circulars/a130/a130trans4.pdf

Shackleford, D. (2008). "Real-Time Adaptive Security". Retrieved January 1st, 2014 from http://www.sans.org/reading-room/analysts-program/adaptiveSec-Dec08

Sherwood, J., Clark, A. & Lynas D. (2009). *Enterprise Security Architecture*. Retrieved January 14th, 2014 from http://www.sabsa.org/white_paper.

Sowa, J. F. & Zachman, J. A. (1992). Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal, 31(3),* 590-616. Retrieved April 21st, 2013 from http://www.zachman.com/ea-articles-reference/50-1992-ibm-systems-journal-extending-and-formalizing-the-framework-for-information-systems-architecture

Sowell, P. K. (2000) "The C4ISR Architecture Framework: History, Status, and Plans for Evolution." Retrieved April 21st, 2013 from http://www.mitre.org/work/tech_papers/tech_papers_00/sowell_evolution/sowell_evolution.pdf

Sluiter, J. (2011). *TOGAF and SABSA Integration: How SABSA and TOGAF complement each other to create better architectures*. Retrieved January 14th, 2014 from http://www.sabsa.org/sabsatogaf

Zachman, J. A. (1987). A Framework for Information Systems Architecture. *IBM Systems Journal, 26(3),* 276-292. Retrieved April 21st, 2013 from http://www.zachman.com/ea-articles-reference/49-1987-ibm-systems-journal-a-framework-for-information-systems-architecture

Zachman, J. A. (2000). *Conceptual, Logical, Physical: It Is Simple*. Retrieved April 21st, 2013 from http://www.zachman.com/ea-articles-reference/58-conceptual-logical-physical-it-is-simple-by-john-a-zachman

Zachman, J. A. (2007). *Architecture is Architecture is Architecture*. Retrieved April 21st, 2013 from http://www.zachman.com/ea-articles-reference/52-architecture-is-architecture-is-architecture-by-john-a-zachman

Zachman, J. A. (2009). *Yes, "Enterprise Architecture Is Relative" BUT It Is Not Arbitrary*. Retrieved April 21st, 2013 from http://www.zachman.com/ea-articles-reference/57-eanotarbitrary

# 6. Appendix A: The Work of John Zachman

The basic concepts of integrated architecture are illustrated very well by John Zachman in his seminal article, "A Framework For Information Systems Architecture" (Zachman, 1987). Zachman was one of the first to write about integrated architecture, and likened it both in name and in philosophy to the architecture used in constructing buildings. His premise is that the process of building an information system requires many perspectives to synchronize, each of which is scoped to a different subset of the project and each of which requires a different level of detail. Constructing a building is much the same; the plumber and carpenter each require a drawing that isolates a particular aspect of the building. The "architecture" is a consistent collection of information from which those drawings are produced that keeps the two synchronized with each other, and with any other drawings produced from it. Synchronization is also achieved from broader, less detailed perspectives, such as the commissioner of the building project (Zachman calls this perspective the "owner" perspective).

Zachman's work describes the unification of multiple, existing methodologies of documenting information systems by making the data homogeneous across them. It then relates the varying perspectives for which these methodologies are used to each other by keeping this consistency in meaning across different levels of detail (Zachman, 1987). For example, it is possible to depict an aspect of an information system, or a building, in many different ways and have all the depictions remain consistent with each other. Similarly, but on a process level, the process of making tea can be modeled using both UML Activity Diagrams and Flow Charts. Both methodologies would likely portray tea making such that it would be understandable, and if you defined elements that are common to both in the same way, then either one could be used to describe the same tea making process. Purifying water, procuring the tea, and disposing of the waste could all be similarly modeled. Zachman's idea of integrated architecture takes this a step further, using the integration achieved by the data to make it possible to relate these processes at

the execution level to methodologies that account for the people doing the work and drinking the tea, or the technical standards used in coffee production.

Zachman and John Sowa expanded on the idea of integrated architecture by adding conceptual graphs to it (Sowa, 1992). Conceptual graphs are a general means of expressing relationships and constraints by using a blend of existential graphs, dependency grammars, and semantic diagrams; in other words, Zachman and Sowa use a blended methodology to accommodate nearly everything into an integrated architecture (Sowa, 1992). Though the use of conceptual graphs as Zachman and Sowa advocated is not widespread, their idea of normalizing the modeling methodology as well as the data has endured in the form of architecture frameworks. Zachman's original treatment of integrated architecture appropriately left out any specification of a preferred methodology or data representation, which would have hindered his idea as a progenitor of modern integrated architecture. His update of the idea with Sowa introduced the most generic methodology possible as an extension of the original framework to be a compromise between specifying a set of preferred methodologies and needing to be able to relate both data and modeling. Organizations that use integrated architectures have chosen to constrain the development of integrated architecture along the same lines as Zachman and Sowa, publishing their own frameworks for internal use.

# 7. Appendix B: Significant Architecture Methodologies

Though the focus of integrated architecture is data integrity and commonality, it is just as important to be able to manipulate and understand the methodologies for depicting that data. Being able to visualize and organize the data in an integrated architecture is critical to communicating the body of knowledge contained in the architecture, and it is the key to making inferences and discoveries that leverage architecture to better derive security requirements.

## 7.1. The DODAF Meta Model (DM2)

The DODAF Meta Model (DM2) is the current description of how DODAF "contains" data as of version 2.0. This is in contrast to older terminology including "Models, Viewpoints and Views", which are how data is represented to isolate desired

aspects of the integrated collection of data within an architecture. The DM2 does not have a direct practical use in deriving security requirements, but as the "DNA" for DODAF, it is useful to understand when building a DODAF-compliant architecture intended to relate to one built compliant to another framework or integrating one into another. The framework is meant to be "data-centric" and accommodate many different methodologies for modeling things from a broad but abstract perspective to a detailed but narrowly focused one. As such, the DM2 is a Data Model, and has three levels that can be represented using DODAF's own Data and Information Viewpoints (DIV) as described on the DODAF 2.02 website:

| **Meta Model Level** | **Corresponding DIV (and DODAF 1.0/1.5 View)** | **Description** |
|---|---|---|
| Conceptual Data Model (CDM) | DIV-1 | "The conceptual level or Conceptual Data Model ***(CDM)*** defines the high-level data constructs from which Architectural Descriptions are created in non-technical terms, so that executives and managers at all levels can understand the data basis of Architectural Description" (DOD DCIO, 2013). |
| Logical Data Model (LDM) | DIV-2 (OV-7) | "The Logical Data Model ***(LDM)*** adds technical information, such as attributes to the CDM and, when necessary, clarifies relationships into an unambiguous usage definition" (DOD DCIO, 2013). |
| Physical Exchange Schema (PES) | DIV-3 (SV-11) | "The Physical Exchange Specification ***(PES)*** consists of the LDM with general data types specified and implementation attributes (e.g., source, date) added, and then generated as an XSD" (DOD DCIO, 2013). |

Table 2: DODAF Meta Model Levels and Descriptions

Being familiar with the DM2 on a macro-level is important to understanding the mechanics of working with a DODAF-compliant architecture, but the details of the DM2 are not necessary to build and use one. Using every part of the DM2 is neither necessary nor recommended, but it is important to an architecture effort for the architect to know which parts are used and how they integrate with others. The key skill to have in practical application is to be able to classify something as conceptual, logical, or physical. This

understanding facilitates the selection of appropriate modeling and viewpoints for a particular project.

A good way of determining this is by evaluating the level of abstraction typically used by the consumer of the information being modeled:

- Conceptual level artifacts are typically consumed by senior management or executive leadership.
- Logical level artifacts are typically consumed by engineers or system designers, and are often where operational requirements or requirements without implementation details are modeled..
- Physical level artifacts are typically consumed by systems or system operators, and are often where implementation details are modeled.

Architectures that are built with the goal of deriving security requirements will likely focus on the LDM, since it is what integrates the implementation details of the physical data model with more abstract components like business rules and policy. Once you have classified the level of information you are trying to model, selecting a viewpoint and a modeling methodology is as straightforward as looking up that level in the DODAF Viewpoints and choosing a compatible modeling methodology. A good example for DODAF would be a military unit's activities: this information fits at the logical level and might be modeled using IDEF0 Activity Diagrams.

There is no special requirement or training to gain expertise in the DM2 beyond what it takes to model something using a compatible methodology. The DM2 is based on the "International Defence Enterprise Architecture Specification" (IDEAS). Work is still being done to leverage IDEAS to make DODAF architectures more easily integrate with other IDEAS-based frameworks (DOD DCIO, 2013).

## 7.2. Viewpoints

Viewpoints are collections of models that each represent elements from an architecture with a similar purpose. The models are graphical, tabular, or text methodologies that show how architecture elements, and by extension the things they

represent, relate to each other. If the DM2 is the "DNA" of the framework, the viewpoints are sets of genes that apply to specific aspects of an organism built from the DNA.

A model can be abstract and broad, such as a conceptual data model, or narrowly focused and specific, such as the specification of a systems interface. The choice of the right model for a particular application can convey the information in a way that makes analysis possible or optimal, and it has direct practical application in deriving security requirements. DODAF Viewpoints are categorized into the following taxonomy:

| Viewpoint Name | Description |
|---|---|
| All Viewpoint | "… describes the overarching aspects of architecture context that relate to all viewpoints" (DOD DCIO, 2013). |
| Capability Viewpoint | articulates the capability requirements, the delivery timing, and the deployed capability (DOD DCIO, 2013). |
| Data and Information Viewpoint | articulates the data relationships and alignment structures "… in the architecture content for the capability and operational requirements, system engineering processes, and systems and services" (DOD DCIO, 2013). |
| Operational Viewpoint | "… includes the operational scenarios, activities, and requirements that support capabilities" (DOD DCIO, 2013). |
| Project Viewpoint | "… describes the relationships between operational and capability requirements and the various projects being implemented. The Project Viewpoint also details dependencies among capability and operational requirements, system engineering processes, systems design, and services design within the Defense Acquisition System process" (DOD DCIO, 2013). |
| Services Viewpoint | "… the design for solutions articulating the Performers, Activities, Services, and their Exchanges, providing for or supporting operational and capability functions" (DOD DCIO, 2013). |
| Standards Viewpoint | "… articulates the applicable operational, business, technical, and industry policies, standards, guidance, constraints, and forecasts that apply to capability and operational requirements, system engineering processes, and systems and services" (DOD DCIO, 2013). |
| Systems Viewpoint | "… for Legacy support, is the design for solutions articulating the systems, their composition, interconnectivity, and context providing for or supporting operational and capability functions" (DOD DCIO, 2013). |

Table 3: DODAF Viewpoints

The "All Viewpoint" is sometimes useful for analysis, but the Operational, Systems, and Services Viewpoints are really the ones that are useful to focus on when building an architecture that will be used to derive security requirements. Models in those viewpoints depict the core of an enterprise, while the "Data and Information Viewpoint" (DIV) and the "Standards Viewpoint" (StdV) apply abstractly across them to enrich what is there. Of the viewpoints, "Project" and "Capability" will likely not be useful for a general application of the DODAF to deriving security requirements, as they were created to address portfolio management needs within DOD (DOD DCIO, 2013). The DODAF 2.0 description has a graphic that shows this arrangement well:

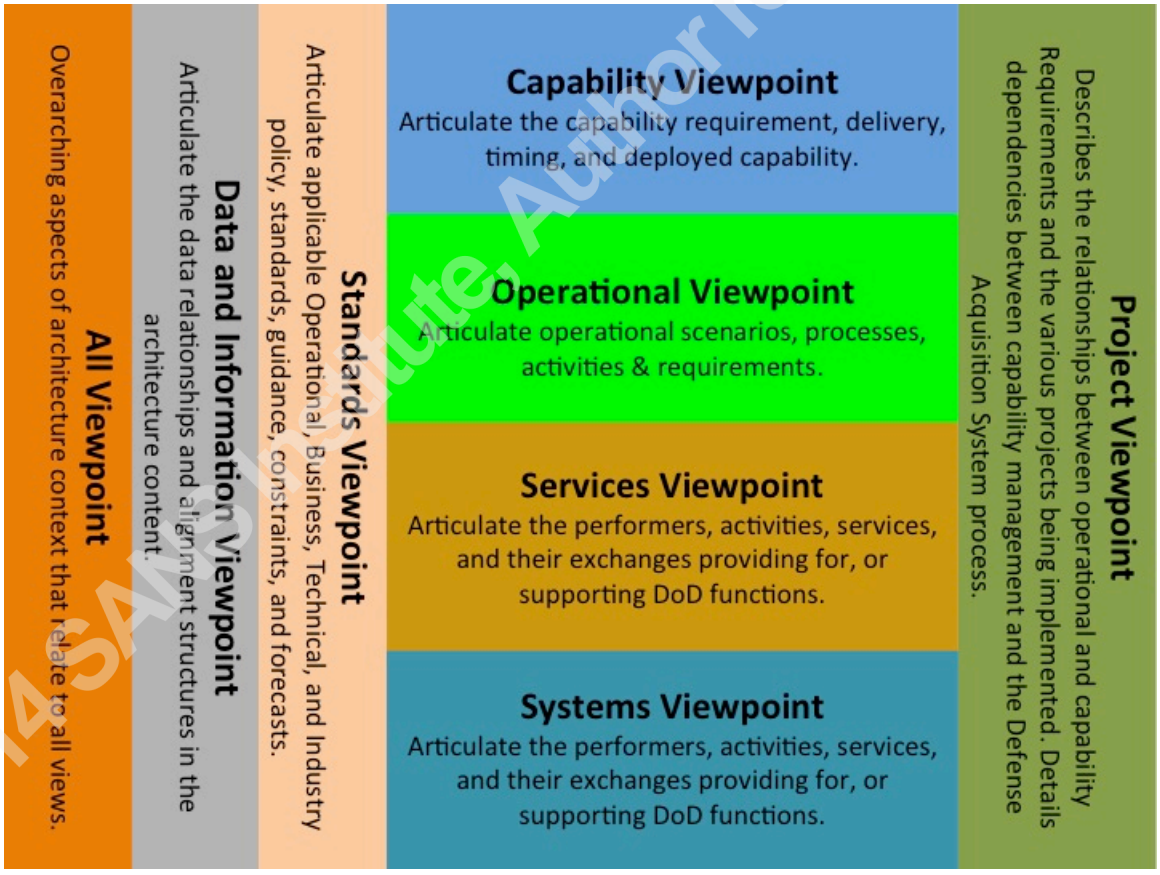


Figure 5: DODAF Viewpoints and their relationships to each other (DOD DCIO, 2013).[4]

[4] This image was reconstructed to approximate the one in the DODAF specification as closely as possible because the one included in those PDFs is low-resolution, but it is not the exact image cited.

## 7.3. Data Modeling

Data modeling is naturally important to a discipline that uses data as a fundamental building block. Data models that are part of an architecture specify the "business rules" for the architecture by constraining the way data is stored. The idea is that in an implementation of the model, the constraints are checked when data is stored to ensure data validity, that in turn ensures data integrity when it is later retrieved. The data model for an architecture is Operational View (OV) 7 in DODAF 1.0 and 1.5 and the Data and Information Viewpoint-2 (DIV-2) in DODAF 2.0. For further details, "IDEF1X" is a good method for modeling relational data, and details on it can be found on the Knowledge Based Systems, Inc (KBSI) website at http://www.idef.com/IDEF1x.htm.

As an example of data modeling, an information system that stores a person's full name as a first name, single middle initial, and last name implements business rules that prevent people with two middle names, a patronymic name (a name that is related to or derived from the father, but is not an exact match), or another non-conforming name from storing their full name. This business rule may be codified in the data model as one way to enforce an enterprise-wide requirement to use a particular technology that only permits storage of names in that way. Business rules such as these can be security-oriented, but accurate modeling is more likely to facilitate the linkage of abstract security requirements to specific parts of an organization, system or enterprise. An example of such a situation is if the Privacy Act of 1974 were to apply to an organization: a data model of the information used or exchanged by the organization could be analyzed for data eligible for protection, then the linkages to systems and activities that use that data would show what parts of the organization must be checked for compliance.

## 7.4. Activity Modeling

Activity modeling records the processes, products and constraints involved in an enterprise. Similar to a data model, an activity model specifies constraints on an enterprise, but in the structure, order, and composition of the work it does. Though business rules can be specified in an activity model, the constraints it is best used to record are the dependencies of an activity performed by the enterprise. Also, activity

models are typically hierarchical, with activities themselves defined in terms of sets of sub-activities down to a level that suits the purpose of the architecture. The activity model is the OV-5 in DODAF 1.0 and 1.5, and the OV-5b in DODAF 2.0. IDEF0 is a good method for modeling operational activities, and details on it can be found on the Knowledge Based Systems, Inc (KBSI) website at http://www.idef.com/IDEF0.htm. Figure 6 provides a basic primer on this method:

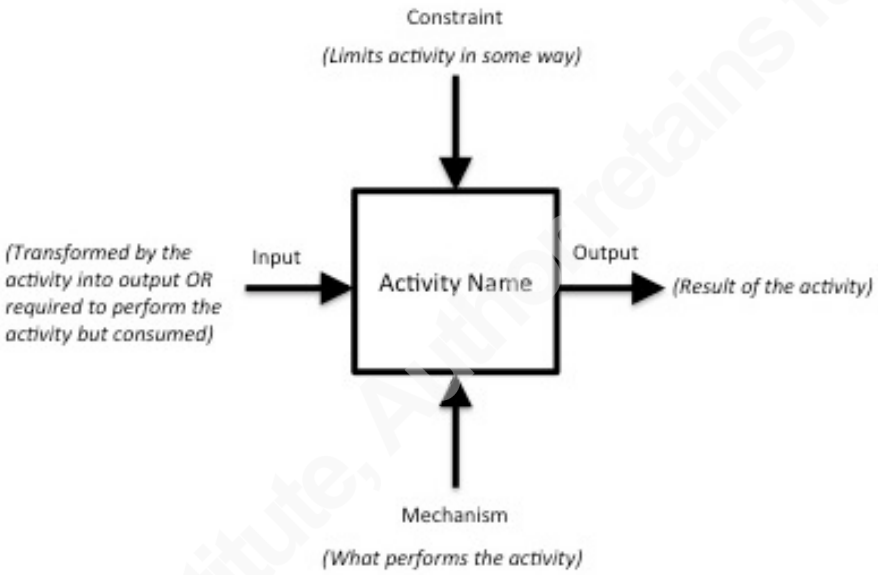


Figure 6: IDEF0 Method with ICOM Arrow Descriptions

As an example of activity modeling, consider a US Army unit. Each unit in the Army has a list of tasks, called the Mission Essential Task List (METL) that it must be able to proficiently execute in order to accomplish the missions for that it was designed. An activity model can be built using those tasks as a baseline, then adding activities that are implied by those tasks, standard Army activities that are required of all units, or even a secondary set of activities undertaken by the unit because of unique circumstances (such as cold weather preparation for a unit in Alaska). Such an activity diagram might look like Figure 7:

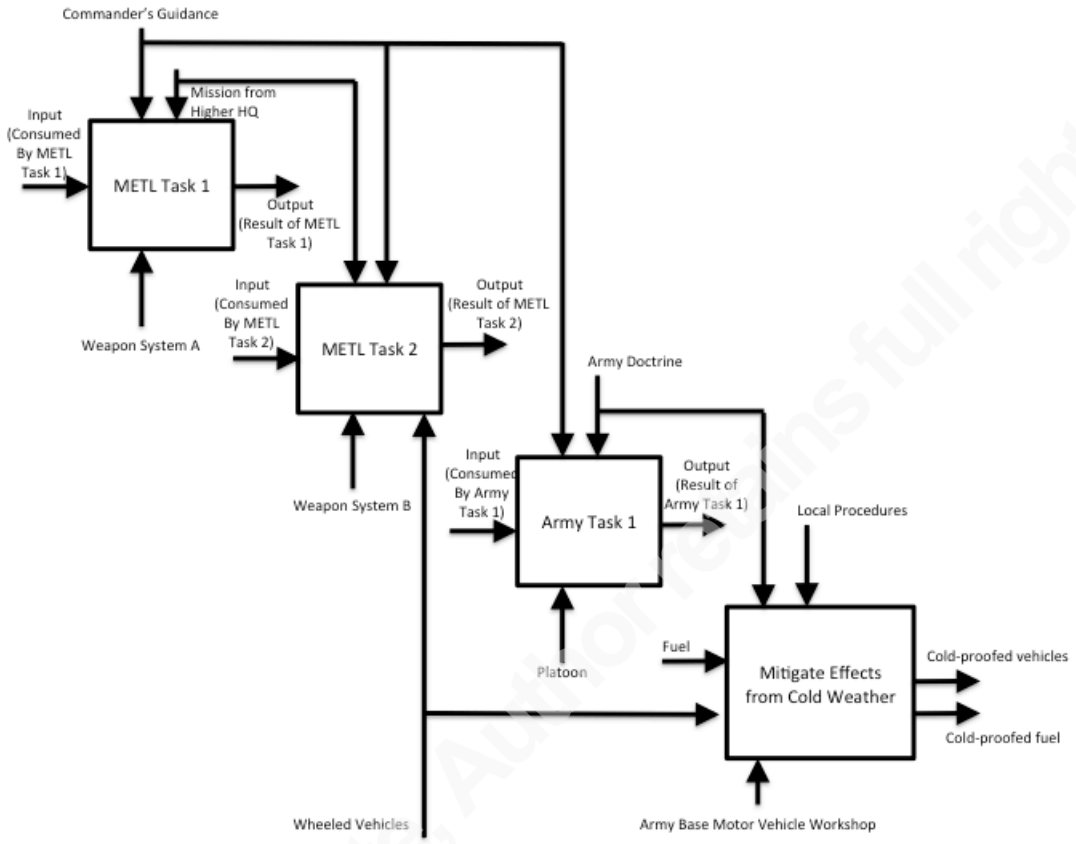


Figure 7: Notional Top Level Activity Diagram for an Army unit

A diagram such as the one in Figure 7 would require each element depicted on it to be defined and included in the All View-2 (AV-2) Integrated Dictionary in order to be truly useful. Assuming it is an accurate depiction of the set of activities undertaken by the Army unit, analyzing the activities and ICOM arrows provides the leverage and insight into the Army unit's organizational processes that permits linkage of abstract security requirements (and other external requirements) to the Army unit. Reusing the example from Data Modeling, if the Privacy Act of 1974 were to apply to this unit, and we had already identified all data classes that are eligible for protection and need to be added to security requirements, we could discover which activities need to implement this protection by tracing the data to the activities through the ICOM arrows, which could contain data as part of their definition. With one more linkage, we could trace from the activities to the linked "mechanism" ICOMs to discover if they represent people (i.e. troops) to build rosters for mandatory Privacy Act protection training.

James E. A. Richards, james.richards1@gmail.com

## 7.5. Matrices

A hallmark of integrated architectures is that they permit a multi-dimensional mapping of information they contain to other information they contain. These linkages are one of the critical results of an architecture effort, since they represent the second and third order links between elements that can exist because of integration. For example, relating national level defense policy to a software program in use by the DOD is not normally possible except through linkages like this because it is hard to depict that relationship in a single view. Matrices are also frequently used to collect information in an architecture effort, since an architect can represent a couple of classes of element as dimensions on a matrix and present it to a subject matter expert to simply mark the cells that indicate a linkage or relationship. Matrices that are based on elements from the architecture enrich views that are produced by the architecture by providing additional degrees of depth to the meaning of elements that appear on the views. Important matrix-based views in DODAF are the SV-5, that links operational activities in the activity model to system functions, the OV-3 and SV-6, that link information and data exchanges to nodes that exchange them, and potentially the views that map the Capability Views (CV) and Project Views (PV) to the rest of the architecture in DODAF 2.0, though no format is specified for those views.

The Army unit from section 7.4 could benefit from the use of matrices both as a tool to collect data and to represent data in an efficient, compact way. For example, an architect could build a matrix with activities on one axis and potential "mechanism" ICOM arrows (i.e. "what" performs the activity) on the other axis, then distribute it to experienced soldiers in the unit to populate based on their knowledge. An "X" in a cell on this matrix would indicate that something represented by an ICOM arrow serves as a "mechanism" to execute an activity, or in other words, the thing represented by the ICOM arrow helps accomplish the activity. When enough responses are collected, the superset of matrix entries could be presented in a single matrix to a different set of experienced soldiers for validation. The soldiers filling in the matrix or validating the entries of other soldiers may not know anything about integrated architecture, but an architect can use their responses to enrich an architecture. Aside from the formally specified matrices, the DODAF is really comprised of hundreds of variants of matrices in

the form of linkages between data. Populating an architecture with these linkages can be achieved by isolating the desired linkage between existing elements in the architecture, then facilitating a data call from subject matter experts, as in the example:

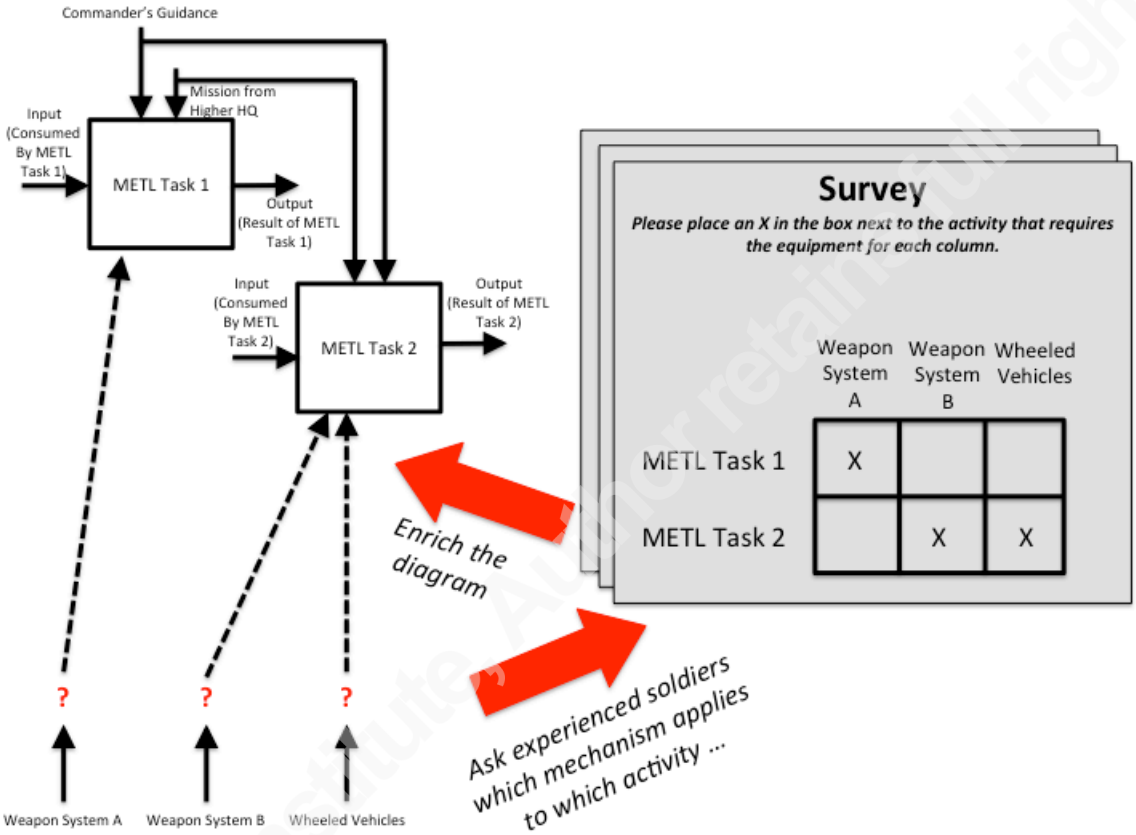


Figure 8: Using a matrix to collect information.

## 7.6. Presentation of Architecture Information

Once information is contained in the integrated dictionary, presenting it consistently with how it was input is a key function that must be combined with using modeling methodologies that are understandable. The architecture elements themselves, once normalized, can be output as page after page of tuples from a database, but that form is not always conducive to analysis and conveying understanding to a reader. DODAF does not require a particular methodology for each view, but it has tacitly recommended some by including examples in view definitions and specified models, such as Unified Modeling Language (UML), Data Flow Diagrams (DFD), and the Integrated Definition (IDEF) series of methodologies. The important part of selecting a view or views to

present the information is that the aspect of the enterprise or system you are examining is exposed by the view with as much relevant context as necessary. In some cases, it may take multiple methodologies rendering the same elements from the architecture to get an optimal view for analysis.

One technique for selecting which views to use for a particular purpose is to determine if the information need from the presentation can be framed as a question. The Zachman framework, one of the architecture frameworks on which DODAF is based, used a construct called an "interrogatives matrix" to map questions about something to views, that DODAF 2.0 has adopted (DOD DCIO, 2013):

| | What (Date) | How (Function) | Where (Network) | Who (People) | When (Time) | Why (Motivation) |
|---|---|---|---|---|---|---|
| **Viewpoint** | AV, DIV | OV, SV, SvcV | OV, SV, SvcV | OV | CV, OV, PV, SV, SvcV | AV, CV, OV, StdV, SV, SvcV |
| **DoDAF-described Models** | AV-2, DIV-1, DIV-2, DIV-3 | OV-5a, OV-5b, OV-6a, b, c, SV-4, SV-10a, b, c, SvcV-10a, b, c | OV-2, SV-2, SvcV-2 | OV-2, OV-4 | CV-2, CV-4, OV-6c, PV-2, SV-8, SvcV-8, Sv-10c, SvcV-10c | AV-1, CV-1, OV-6a, StdV-1, StdV-2, SV-10a, SvcV-10a |
| **Meta-model group** | Information and Data, Project | Activity, Capability, Service, Measures | Location | Performer | All | Rules, Goals |

Table 4: DODAF 2.0 Interrogatives Matrix (DOD DCIO, 2013)

It may seem simple enough to choose a view that "answers the question", but if the view was not built as a means to collect data and populate the architecture, there may be some missing elements; you can only get out what you put in. For example, if you select the OV-5b (Operational Activity Model) but only previously built an OV-5a (Operational Activity Decomposition Tree) and several OV-6cs (Event-Trace Description), you will find that your OV-5b lacks ICOM arrows, since they would not be required to construct OV-5a and OV-6c. The reverse would not necessarily be true, as an OV-5b would be sufficient to generate an OV-5a, and many automated tools do so as a feature. As esoteric an example as this may seem, it is a situation that can be avoided

with some analysis of what information is present in an architecture prior to trying to render a particular model or view out of the architecture repository.