12TH ICCRTS
"Adapting C2 to the 21st Century"

Using JCIDS DoDAF Architecture Primitives to Assemble a Repository for Enterprise-wide Analysis and Decision-Making

Topics:
Track 1: C2 Concepts, Theory, and Policy
Track 8: C2 Technologies and Systems
Track 3: Modeling and Simulation

Lawrence McCaskill,
Robert Hicks,
and
Ian Komorowski

POC: Lawrence McCaskill
Whitney, Bradley, & Brown, Inc.
1604 Spring Hill Rd, Suite 200
Vienna, VA 22182
703-448-6081 x127
lmccaskill@wbbinc.com

Introduction

Abstract

As part of the Joint Capabilities Integration and Development System (JCIDS) process, the Department of Defense (DoD) has mandated development of DoD Architecture Framework-compliant architectures in support of the Net-Ready Key Performance Parameter used in both requirements (Initial Capabilities Documents, Capabilities Development Documents, Capabilities Production Documents) and acquisition documentation (Information Support Plans). With this requirement, the DoD is recording vast amounts of information, including information flows that move on and off from the described platforms, with uses far exceeding the sphere of JCIDS. However, this information is not being gathered into repositories that will enable reuse of this information not only in the JCIDS process, but for myriad other uses including Modeling and Simulation.

This paper will propose a methodology for capturing and amalgamating information collected during the JCIDS process, enabling true enterprise architectures to be built using their constituent parts, with far-reaching application beyond the realm of JCIDS. Additionally, the paper will recommend a governance process by which this information can be maintained throughout the life cycle of the various programs for use in the JCIDS process, as well as other applications, including Modeling and Simulation supporting analyses for acquisition, operations, and simulation of operations.

Thesis

DoD Architecture Framework (DoDAF) artifacts, developed in support of the JCIDS process, contain a plethora of information about individual systems. When properly structured, this information can be used to develop enterprise architectures representing the amalgamation of several systems in a coherent and executable framework. Amalgamation, as discussed in this paper, is a process of bringing several pre-existing system architectures together in a coherent manner. It should no be confused with developing a “System-of-Systems” architecture with well defined boundaries that initially include each of the subject systems.

The data compiled in the DoDAF architectures is also very useful in establishing the structure and baseline functional environment for modeling and simulation (M&S) efforts. Architectures describe system functionality, how they are used operationally, and what information and data flows between components. M&S provides an operational laydown in which the disparate architectures are interconnected and exercised.

Interrelated information is compiled via DoDAF architectures, the M&S community, and any of a number of efforts across the spectrum of Doctrine, Organization, Training, Materiel, Leadership, Personnel, and Facilities (DOTMLPF), as well as the testing and financial communities. This information should be made accessible in searchable, updatable data stores available to the disparate communities who will use the data, using a governance policy enforceable through technical means. Technologies to bring this to fruition are beginning make themselves available.

Architecture "primitive", a definition

As defined by the Encarta Dictionary: English (North America), a primitive is a "a simple element of a computer program or graphic design from which larger programs or images can be constructed" or "something such as a concept, feature, or formula from which something else is derived".

Architectures are composed using discrete parts, including activities, system functions, business processes, and ties to doctrine (Universal Joint Task List [UJTL], Service Task Lists, Department of the Navy Common Systems Functions List [DoN CSFL], Net-Centric Operations and Warfare Reference Model [NCOW RM], etc.). These composeable parts are what we are addressing when we speak of architecture primitives, and these primitives can be used and reused across multiple architectures in multiple domains.

Background

The Net Ready-Key Performance Parameter (NR-KPP) is a required element of the Capability Development Document (CDD), Capability Production Document (CPD), and Information Support Plan (ISP). Within the context of these documents, the NR-KPP provides the underlying information and structure needed to assess the interoperability and supportability, and provide system developer and managers the means to effectively and efficiently build and maintain the required capabilities.

Relationship of JCIDS documents and the Information Support Plan (ISP)

The Initial Capability Document (ICD) describes capability gaps in joint warfighting functions and establishes the need for a materiel approach to resolve a specific capability gap derived from the JCIDS assessment and analysis process.[1] While a NR-KPP is not required for the ICD, the basic concepts of interoperability should be an inherent part of the resulting documentation

The Capability development Document (CDD) defines the user's requirements for the specific capability. It provides the operational performance attributes, including interoperability and supportability, necessary for the acquisition community to design the proposed system. As part of the CDD process, The NR-KPP shall be defined by the acquiring authority, certified by the Chairman of the Joint Chiefs of Staff, and documented in the CDD.[2]

[1] DODI 4630.8, June 30, 2004, Para 6.2.2

[2] *ibid*

The Capability Production Document (CPD) is evolved from the CDD during the System Development and Demonstration (SDD) phase of the acquisition process. It provides the operational performance attributes necessary for the acquisition community to produce a single increment of a specific system. It presents performance attributes, including KPPs, to guide the production and deployment of the current increment. CPD development is guided by the integrated architectures and relevant JCIDS documentation. A NR-KPP, certified by the Chairman of the Joint Chiefs of Staff, shall be documented in the CPD. [3]

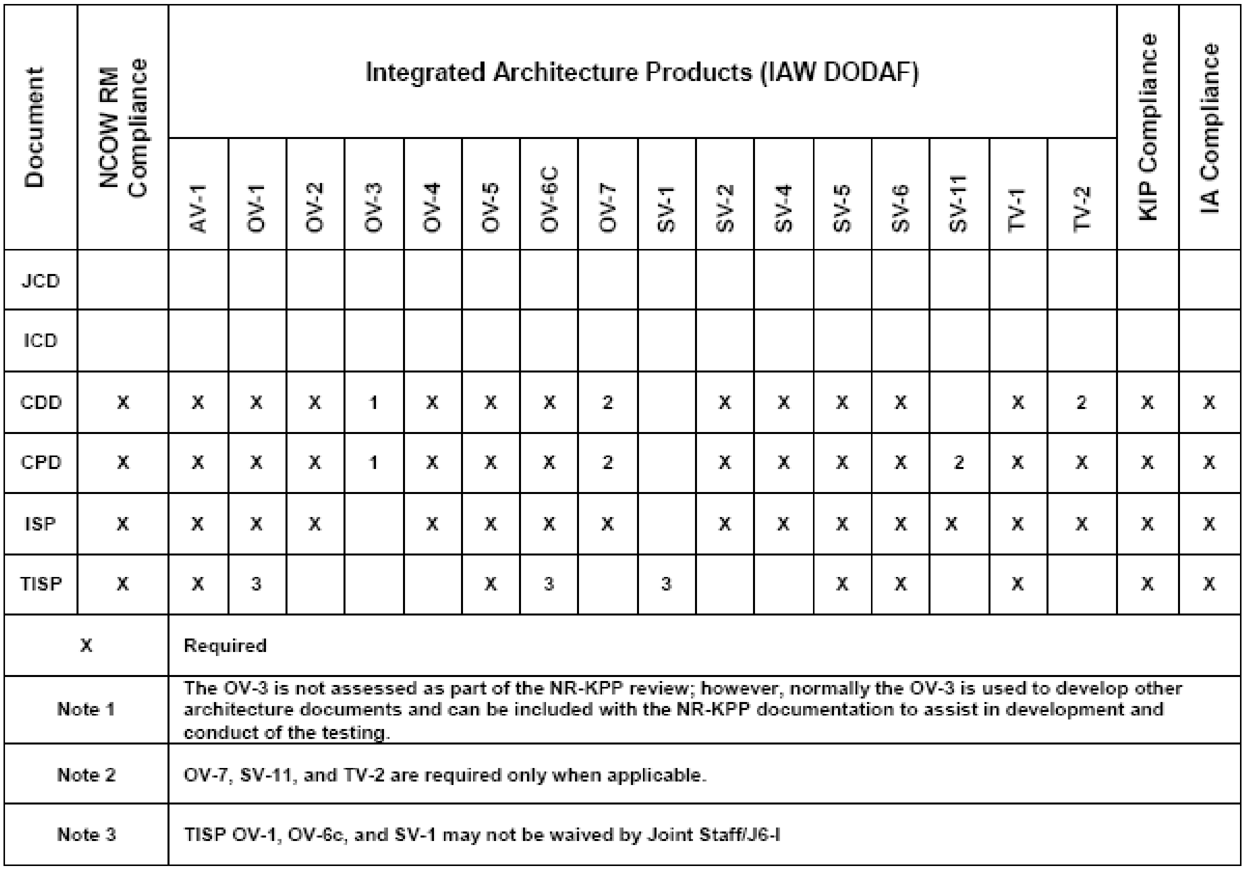
While the ICD, CDD, and CPD are JCIDS documents produced by requiring organizations, the Information Support Plan (ISP) is an acquisition document produced by the acquisition community. The ISP identifies Information Technology (IT) and National Security Systems' (NSS) information needs, dependencies, and interface requirements, focusing on interoperability, supportability, and sufficiency. The ISP includes an operational employment concept; system interface descriptions; required information exchanges; IT and NSS information support requirements derived from analysis of applicable Joint Operational Concepts (JOCs), Joint Functional Concepts (JFCs), and JCIDS documentation, and the associated integrated architecture(s); potential issues; and proposed solutions. IT and NSS systems dependencies and interface requirements are described in sufficient detail to enable test planning for verification of the NR-KPP.[4]

Platform Architecture Primitives Collected in the NR-KPP

The NR-KPP embodied in the ISP is maintained and subject to recertification throughout the life-cycle of the system. This puts the NR-KPP in the unique position of being initially developed in the requirements community (CPD & CDD), redefined and enhanced during system development (ISP), and maintained and updated while the system is in operation (ISP). During the course of NR-KPP development and refinement, the system architect collects an expansive set of data that forms the underlying structure of the integrated architecture. Consistent with DoDAF guidance, and the Core Architecture Data Model (CADM), the following products (see table 1) are identified in CJCSI 6212 01D (Chairman of the Joint Chiefs of Staff Instruction) to be used to collect and present the data in the NR-KPP.

[3] i*bid,*

[4] *ibid*



Document	NCOW RM Compliance	Integrated Architecture Products (IAW DODAF)																KIP Compliance	IA Compliance
		AV-1	OV-1	OV-2	OV-3	OV-4	OV-5	OV-6C	OV-7	SV-1	SV-2	SV-4	SV-5	SV-6	SV-11	TV-1	TV-2		
JCD																			
ICD																			
CDD	X	X	X	X	1	X	X	X	2		X	X	X	X		X	2	X	X
CPD	X	X	X	X	1	X	X	X	2		X	X	X	X	2	X	X	X	X
ISP	X	X	X	X		X	X	X	X		X	X	X	X	X	X	X	X	X
TISP	X	X	3				X	3		3			X	X		X		X	X
X		Required																	
Note 1		The OV-3 is not assessed as part of the NR-KPP review; however, normally the OV-3 is used to develop other architecture documents and can be included with the NR-KPP documentation to assist in development and conduct of the testing.																	
Note 2		OV-7, SV-11, and TV-2 are required only when applicable.																	
Note 3		TISP OV-1, OV-6c, and SV-1 may not be waived by Joint Staff/J6-I																	

Table 1: NR-KPP Products Matrix[5]

The current requirements and acquisition process is structured to define and build individual platforms and systems. They do not, in a single program, address broad spectrum requirements to acquire specific "capabilities." As a result, the NR-KPPs and supporting integrated architecture normally address individual platform or systems. However, given this constraint, and the need to truly address interoperability issues, we must define a method to logically connect the platform level architectures into enterprise architectures suitable for end-to-end analysis. The enterprise architecture needs to reflect the interoperability of numerous systems operating within well defined environments.

[5] CJCSI 6212 01D, 8 March 2006

Difficulties in the Development of Enterprise Level Architectures

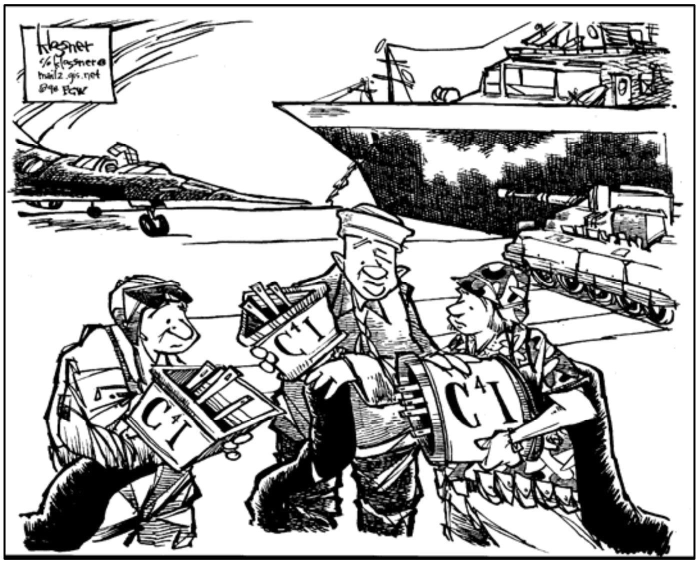
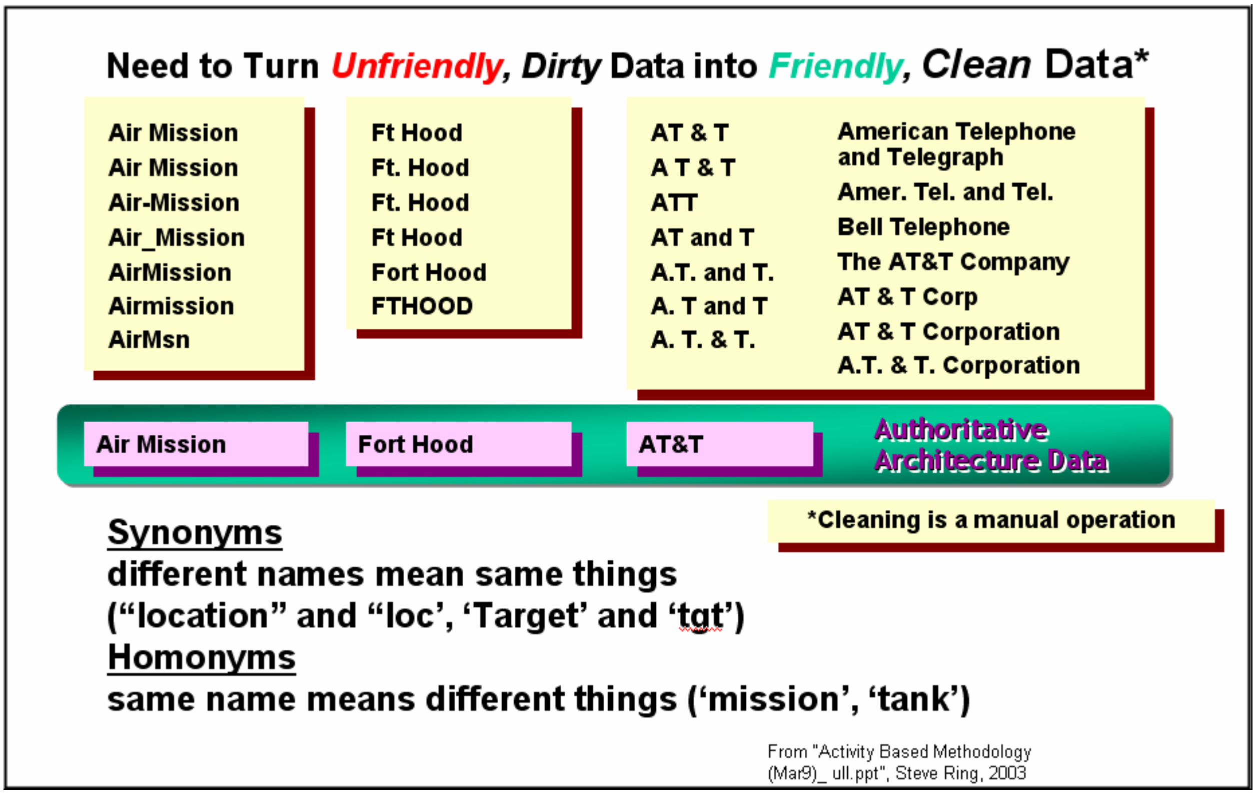


Figure 1: Joint C4I Interoperability
(Jim Klossner, Federal Computer Week, 1998)

The overarching goal of the development of enterprise architectures is the interoperability of systems across the sphere of Joint operations. A precondition of this interoperability should be the interoperability amongst enterprise architectures. However, even with DoD architects “speaking the same language” (i.e. DoDAF), enough inconsistencies exist across architectures to make the above cartoon applicable. An inexhaustive list of reasons for these inconsistencies is noted in the following paragraphs.

Even if multiple architectures have been developed by the same team, many times the “data” (i.e. artifacts collected within the architecture) is “dirty” (see Figure 2) when compared between separate efforts. Spelling, abbreviation (or lack thereof), synonym and homonym differences all will exist in the naming conventions, unless the organization has developed well defined ‘pick lists’ for <u>common</u> artifacts used in the architecture.



Need to Turn *Unfriendly*, *Dirty* Data into *Friendly*, *Clean* Data*

Air Mission
Air Mission
Air-Mission
Air_Mission
AirMission
Airmission
AirMsn

Ft Hood
Ft. Hood
Ft. Hood
Ft Hood
Fort Hood
FTHOOD

AT & T
A T & T
ATT
AT and T
A.T. and T.
A. T and T
A. T. & T.

American Telephone and Telegraph
Amer. Tel. and Tel.
Bell Telephone
The AT&T Company
AT & T Corp
AT & T Corporation
A.T. & T. Corporation

Air Mission
Fort Hood
AT&T
Authoritative Architecture Data

*Cleaning is a manual operation

Synonyms
different names mean same things
("location" and "loc', 'Target' and 'tgt')
Homonyms
same name means different things ('mission', 'tank')

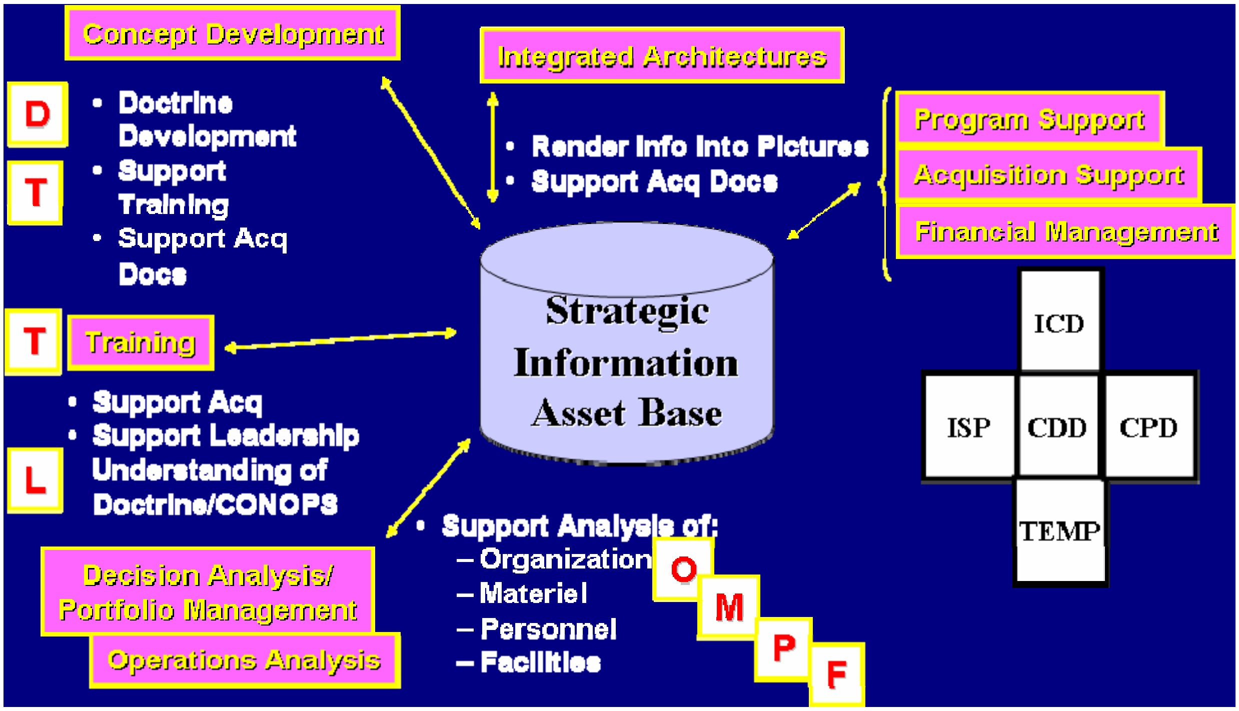
From "Activity Based Methodology (Mar9)_ ull.ppt", Steve Ring, 2003

Figure 2: Clean Data Concept

So what should be done when one platform's architecture defines <X> information exchange as being sent to anther platform, but the receiving platform's architecture defines the receipt of <Y>? It comes down to an either/or choice, that of 'scrubbing' the dirty data to clean it, or creating a bridge between the two artifacts. Neither choice is easy.

There is currently no facility being employed to "check the homework" of the programs. Since the process of accomplishing integrated architectures as part of the JCIDS process is relatively new, the organizations responsible for ensuring correctness of architectures being submitted to the JCIDS process (OASD/NII, Joint Staff/J-6I, and service CIO architecture representatives) have been checking architecture viability by checking whether the artifacts within the architecture are internally consistent. While internal consistency within platform architectures important and necessary, it is not sufficient for establishing interoperability. Thus, we assert the primary focus of these "homework checking" organizations should be the interrelationships between architectures, ensuring that the individual components of their respective enterprises are correctly and accurately represented, such that they can amalgamate the information and perform enterprise-wide analysis of the architectures in order to determine possible gaps, economies-of-scale that can be leveraged, and provide primitives for use in subsequent architectures.

However, there is no facility or methodology being employed that can pull disparate platform architectures into a coherent enterprise. In performing the cross-enterprise analysis, the CIO-oriented organizations need to be focusing on collecting and amalgamating information for analysis as depicted in Figure 3. These datasets can (and should) be made available across the enterprise, across all facets of DOTMLPF, finance, and enterprise resource planning (i.e., it's not just about IT). This is what was codified in the Clinger Cohen act, and it simply isn't happening.


Concept Development

D

- Doctrine Development
- Support Training
- Support Acq Docs

T

T

Training

- Support Acq
- Support Leadership Understanding of Doctrine/CONOPS

L

Decision Analysis/ Portfolio Management

Operations Analysis

Integrated Architectures

- Render Info Into Pictures
- Support Acq Docs

Strategic Information Asset Base

- Support Analysis of:
  - Organization
  - Materiel
  - Personnel
  - Facilities

O

M

P

F

Program Support

Acquisition Support

Financial Management

ICD

ISP

CDD

CPD

TEMP

Figure 3: Strategic Information Asset Base

Vast amounts of information is being collected via several different disciplines (M&S, Operational Testing and Evaluation [OT&E], etc.) as well. None of it is going into a federated data store for use across multiple disciplines (Portfolio Management, Manpower, Doctrine, etc.). Assuming one is collecting information about disparate programs across the life cycle of the enterprise, these pieces of information should be feeding a federated data store such that information gets used, reused, and updated as better information becomes available. Some examples of extreme waste: every platform architecture depicting air operations contains references to AWACS, E-2, and other command and control (C2) platforms; however, none of them have the same depiction of the platforms, and often create their own from scratch. Testing data related to the platform links isn't in a centralized location; example: one shouldn't be forced to come up with Information Assurance, size, throughput, etc. about Link 16 – it's a program of record, and thus should be available for programs to use via "pick list" type data sets. Modeling and Simulation has performance characteristics built in to each model; if one is to assume these are correct , why can't these performance criteria be made available for others to use? As for the manpower equation, shouldn't one be able to leverage from these datasets to provide the primitives for use in manpower analyses?

The answer is obvious, and yet we continue to develop these sets of information in stovepipes. There is also the question of whether the amalgamation of information makes it classified, but in the endgame, one should be able to perform that analysis on what to make available to whom via their "role in the world," and tailor or filter the information presented to the appropriate level of visibility and classification.

The Enterprise Primitive Amalgamation Process

We posit that the DoDAF architectures that have been and are currently being created for the JCIDS acquisition process, while generally focused at the platform level (aircraft, missiles, vehicles, etc), and often not being built with the idea that they must be merged with other platform level architectures, are a rich resource that can be parsed as sets of primitive elements to be utilized in developing the enterprise level architecture and for analysis in modeling and simulation.

Common models for the exchange of information have been an unachievable goal for many years. Several efforts aimed at standardizing information structure across the DoD enterprise have failed due to a number of reasons, including cost of entry, difficulty in implementation, and lack of a overarching authority to enforce adherence to the standards amongst programs. In the endgame, we do not believe achieving a huge, monolithic data resource for all entities across the DoD is possible. However, we do believe a methodology exists that can amalgamate disparate data resources for analysis on an as-needed basis. This methodology has the ability to assist the architect in analyzing how disparate architecture artifacts may be combined into a larger architecture for use in subsequent analyses. Additionally, using this methodology, one can begin to develop reusable architecture, data, and reference primitives from which to draw information for future analyses.

Integrating independently developed architectures is a difficult process. Regardless of the automation involved, it will take manual intervention by a person with unique qualifications: they must be technically inclined and be extremely knowledgeable in the subject area (i.e., for military applications, they must have a keen understanding of military operations).

One advantage that the DoD has at this point is that all architectures created for JCIDS are in roughly the same language, that of the DoD Architecture Framework.   Therefore, there is a basic starting point, as theoretically the architectures are described in the same format, and most of the architectures have the same set of "buckets" for the structure of their artifacts. However, since the framework does not provide strict guidance with regards to the types of notations to be used (IDEF, UML, BPMN, structured analysis, etc), and in the methodologies (ABM, the FEAC process) to develop the architecture, there are often variations in what artifacts are captured.

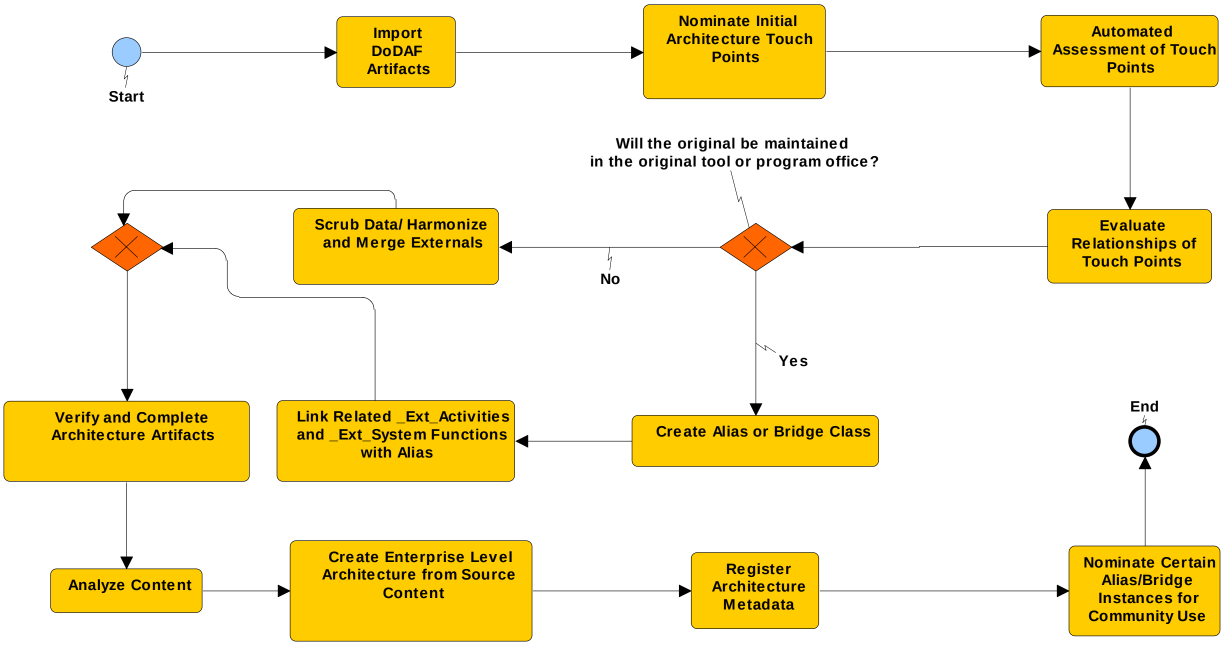
With that being understood, the primitives that can be compared between architectures are the following:

- OV-2 Operational Nodes
- OV-3 Information Exchanges
- OV-4 Organizations and Roles
- OV-5 Activities
- SV-1/2 Systems Nodes
- SV-1/2 Systems
- SV-4 System Functions
- SV-6 System Data Exchanges

However, one person's enterprise is another person's system. Differences in scope, purpose, and viewpoint also create issues, but the basic DoDAF structure does allow for federation of the architectures. If disparate architectures, no matter which modeling tools or modeling tool suites they were created in, are collected into a repository, they can be analyzed for touch points and potential touch points. When those are identified, they are in essence federated, as no changes have been made to the original work.

Federation though, does not lend itself to easy analysis, amalgamation, or abstraction up to the next or enterprise level, due to the probable difference in purpose, scope, view, and in terminology. This is where emerging technologies allowing for fuzzy search, or a search that is not 100% literal (results are returned that may be similar, and are ranked as to the degree of similarity) may provide assistance.

A nominal flow for taking independently developed architectures and amalgamating them follows in Figure 4.



Start
Import DoDAF Artifacts
Nominate Initial Architecture Touch Points
Automated Assessment of Touch Points
Will the original be maintained in the original tool or program office?
Scrub Data/ Harmonize and Merge Externals
No
Evaluate Relationships of Touch Points
Yes
Verify and Complete Architecture Artifacts
Link Related _Ext_Activities and _Ext_System Functions with Alias
Create Alias or Bridge Class
End
Analyze Content
Create Enterprise Level Architecture from Source Content
Register Architecture Metadata
Nominate Certain Alias/Bridge Instances for Community Use

Figure 4: The Enterprise Primitive Amalgamation Process

- **Import DoDAF Artifacts**
  It is assumed that one is using an automated tool, based on a relational database, to accomplish this methodology. Most modeling tools on the market currently won't allow for the queries and analysis that are demanded by the methodology, but a new class of tools is emerging that will allow this. As a result, the source architectures, no matter which tool they were originally created in, are imported into the repository management tool.
- **Nominate Initial Architecture Touch Points**
  The analyst or amalgamator will be able to recognize some potential artifacts that may be related: Operational Activities, Operational Nodes, Information Exchange Requirements, System Functions, System Nodes, and/or System Data Exchanges. Even if the analyst did not create the original architecture, their expertise of the enterprise should allow them to identify these, even if they are at different levels of abstraction.
- **Automated Initial Assessment of Touch Points**
  In addition to the amalgamator's expertise in the identification of touch points, an automatic analysis is performed on the source architectures. On all but the simplest DoDAF architectures, there is generally a high level of detail. Due to this factor, fuzzy searches will be performed, and architecture primitives will be nominated that may be additional touch points.
- **Evaluate Relationships of Touch Points**
  A manual assessment is made of the architecture primitives that have been nominated as touch points in the previous two steps. Do the objects that have been selected really mean the same thing? Are there issues with levels of abstraction?
- **Decision Point**
  The decision point is whether or not the original architectures will be maintained by the program office, or by whomever is building the enterprise architecture. This is a question of stewardship, or who owns the architecture when it's done. If the owner of the enterprise architecture will also own all the data, then matters become somewhat easier – one proceeds to the "Scrub Data/Harmonize and Merge Externals." If the owner of the enterprise architecture does not own the source data, one proceeds to "Create Alias or Bridge Class".
- **Scrub Data/Harmonize and Merge Externals**
  The scrubbing of enterprise architecture data, while potentially able to be assisted by technology, is essentially a manual process. Once the terms that are thought to mean the same thing are collected in the initial steps, one needs to go through them and clean them up. For example, USAF, U.S.A.F, US Air Force, and Air Force may have been used in four different program level architectures to describe the United States Air Force. A common term needs to be agreed on, and then all instances of the above need to be changed to the common term. Since the architectures have been collected into a common repository with this methodology, this can be easily accomplished on the technical side. However, the biggest difficulty in scrubbing is not technical. Who has the authority to enforce these changes in the source architectures? If the "dirty" data is just a matter of different use of abbreviations, this may be easily decided or agreed

upon. However, if the original terms have a social or emotional value, or if they are only understood in the original terms by the audience they were originally intended for, the authority to make the change needs to be well established, and this generally needs to be as a result of funding/budget control. **Note**: at some point, the enterprise architect will need to be able to provide these standardized primitives to the creators of subsequent architectures and mandate their use, or the “do loop” regarding harmonizing the architectures will continue ad infinitum. See also the last step in this process.

- **Create Alias or Bridge Class**
  This step in the flow is reached if the assumption is that the architectures are to be federated, with the original being owned by its creator (assumed to be in a program office). If the different terms for artifacts cannot be changed (whether due to lack of authority or for other reasons), the common term can be identified (and a new artifact can be created if the common term has not been utilized yet).
- **Link Related _Ext_ Activities* and _Ext_ System Functions* with Alias**
  All synonyms can be linked to an Alias or Bridge class instance with a “this is the same as” relationship. This in effect creates a consistent artificial construct that can be used to analyze information flows across platforms or programs. While the bridging concept does create an additional aspect of complexity, it does avoid the political and funding implications of scrubbing the data.
  ***Note:** the _Ext_ Activities/Functions are artifacts of how WBB implements the Activity Based Methodology – certain Activities and Functions have “_Ext_” as part of their name to alert the architecture reader that these are external to the scope of the architecture, and are only included as sources/sinks of information/data. For more information on this methodology, see “An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures”. Regardless of the naming conventions used, there will likely be the same “dirty data” problem discussed earlier – these names will have to be identified as aliases in an Alias or Bridge class of object, which will be used in subsequent queries and analysis.
- **Verify and Complete Architecture Artifacts**
  This is where the difficult part of the analysis begins. One must determine whether the primitives (activities, nodes, exchanges, roles, systems) are indeed the same, and whether or not the architectures are in agreement regarding what pieces of information are flowing between them. In the case where there’s disagreement on the information flow (i.e., there are exchanges depicted in one model, and not the other, or exchanges that cannot be mapped), this is where governance should dictate how to rectify “who is right” and “who changes their architecture.” Governance is covered in more detail in section 5.
- **Analyze Content**
  This is where one can begin branching out of the “strictly architecture” uses for architecture. An example: use the architecture primitives to source modeling and simulations to determine gaps, bottlenecks, and overall feasibility of operational concepts. Results of the analysis would feed back into the architecture, and the data regarding performance (size, bandwidth required, etc.),

as well as the architecture primitives can be made available for reuse in subsequent architectures under the purview of the enterprise.

- **Create Enterprise Architecture from Source Content**
  Just linking the touch points of the source architectures may not be enough. In order for an enterprise perspective to be complete, other artifacts and primitives may need to be created, and/or certain subsets of the source data may need to be abstracted.
- **Register Architecture Metadata**
  As we are not advocating the creation or development of a single massive DoD database for architectures, the metadata that allows for others in the Community of Interest (COI) to identify and find the newly created architecture needs to be registered. Currently, the DoD Architecture registry System (DARS) is the tool that facilitates this.
- **Nominate Certain Alias/Bridge Instances for Community Use**
  In order to facilitate future uses of this methodology, instances of the new alias or bridge class object types can be tagged with metadata that identify it as an object that may need to be included in a common list.

Two possible scenarios are envisioned for this process. The easiest, but nontrivial and highly abstracted case is where there are two architectures being combined with one obvious touch point in the OV-2's of each. An example of this, illustrated in Figure 5, would be a multi-role tactical fighter (Architecture 1) communicating with an airborne C2 Element, such as an AWACS, or E-2 (Architecture 2). Via identification of the "start here" touch points, the obvious articles to examine regarding the architectures are the node names (both operational and system), the information/data exchanges, organization names, roles, and systems. Once these have been rectified via the above process, one is "done" (at least with this iteration of "enterprise via amalgamation").

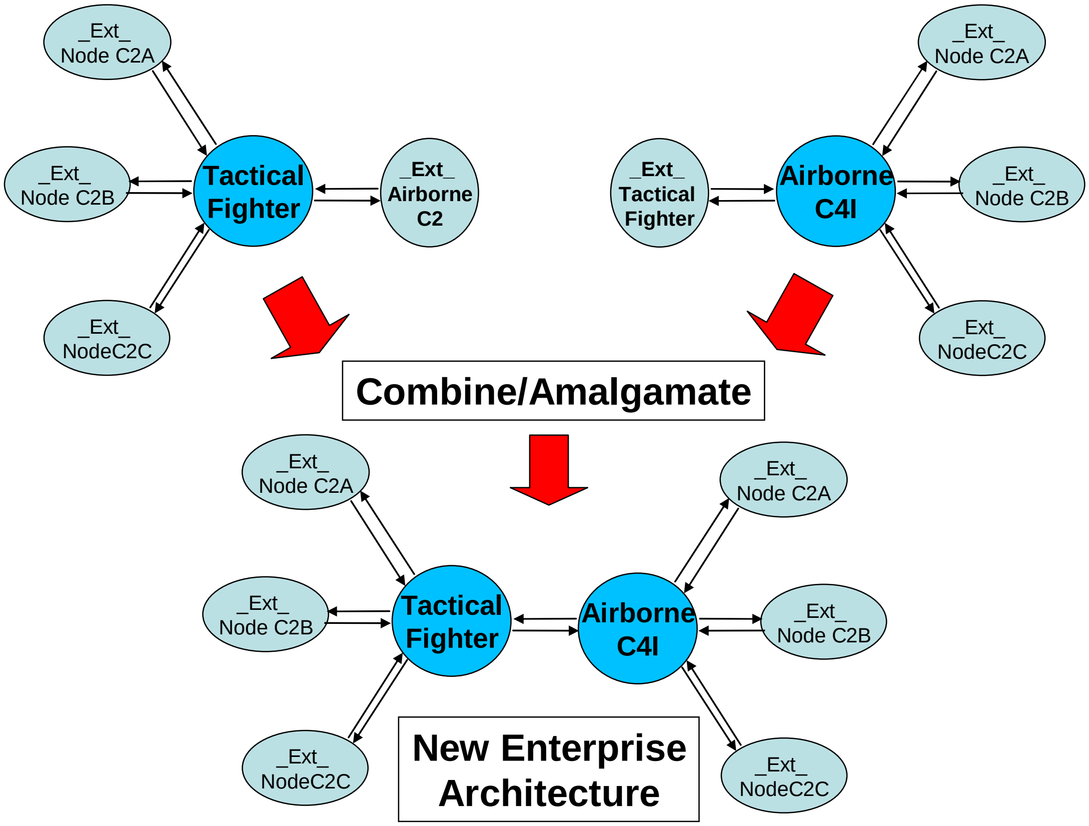


Figure 5: Simple Amalgamation Scenario

The more difficult case, illustrated in Figure 6, is where two architectures are being merged that contain the same or similar elements, but do not necessarily have only one touch point. A concrete example is the attempt to combine the following architectures:

- A Joint Fires Architecture: within which only Air Force support to Army units is depicted, with the Army RSTA (Reconnaissance, Surveillance, and Target Acquisition) units being depicted as internal to the architecture because they are part of the joint fires chain-of-events (i.e., Enlisted Terminal Air Controllers within the Unit of Action (UA) are who initiate the Call for Fires).
- An Army Combined Arms UA Architecture: within which all facets of the UA are detailed.

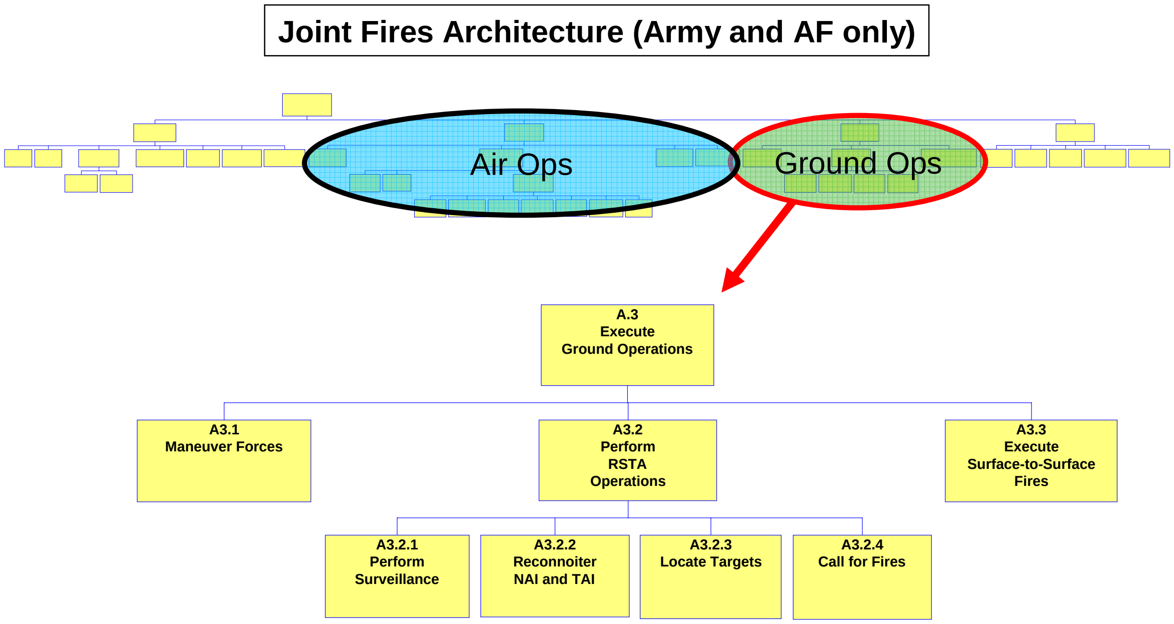


Figure 6: Joint Fires Architecture

Within the Joint Fires Architecture, Army operations are "oversimplified" due to the scope and viewpoint of the architecture, which, from an Army perspective, is only concerned with the activities that generate a call for fires that will be answered by an AF close air support asset.

With the goal of combining this architecture, as depicted in Figure 8, with an Army Combined Arms Unit of Action architecture, Figure 7, one begins to see the complexity of this endeavor.

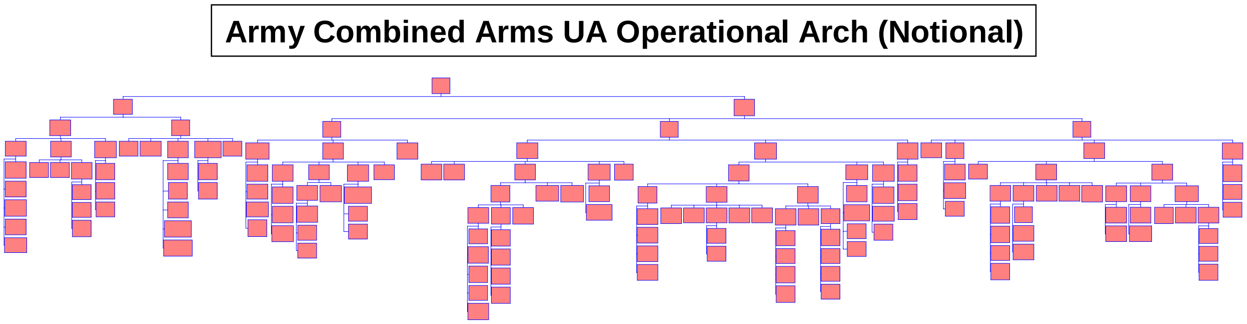


Figure 7: Army Combined Arms UA Operational Architecture

The activities assigned to the Army RSTA unit will be at different levels of abstraction in the two architectures. The activities depicting Army Maneuver, RSTA, and Fires execution will be depicted (in one form or another) in both architectures, but deciding where to "join" the architectures will require both manual and (hopefully) automated analysis aided by tools. Code can be developed to accomplish fuzzy searches at each level of nodes being compared in each model to determine candidate joins in the architecture. In the case of activities being searched, the searches at each level (traversing of levels easily accomplished via n-tree algorithms in any 300-level programming textbook) should accomplish analyses of:

- Is this the same activity (fuzzy search of name and definition to determine relative match of name/key words in definition)
- Inputs/Outputs (fuzzy search of title and definition, similar to above)

Once matches are determined, the tool should facilitate the merging of architectures, including the building of bridge/alias objects, as applicable.

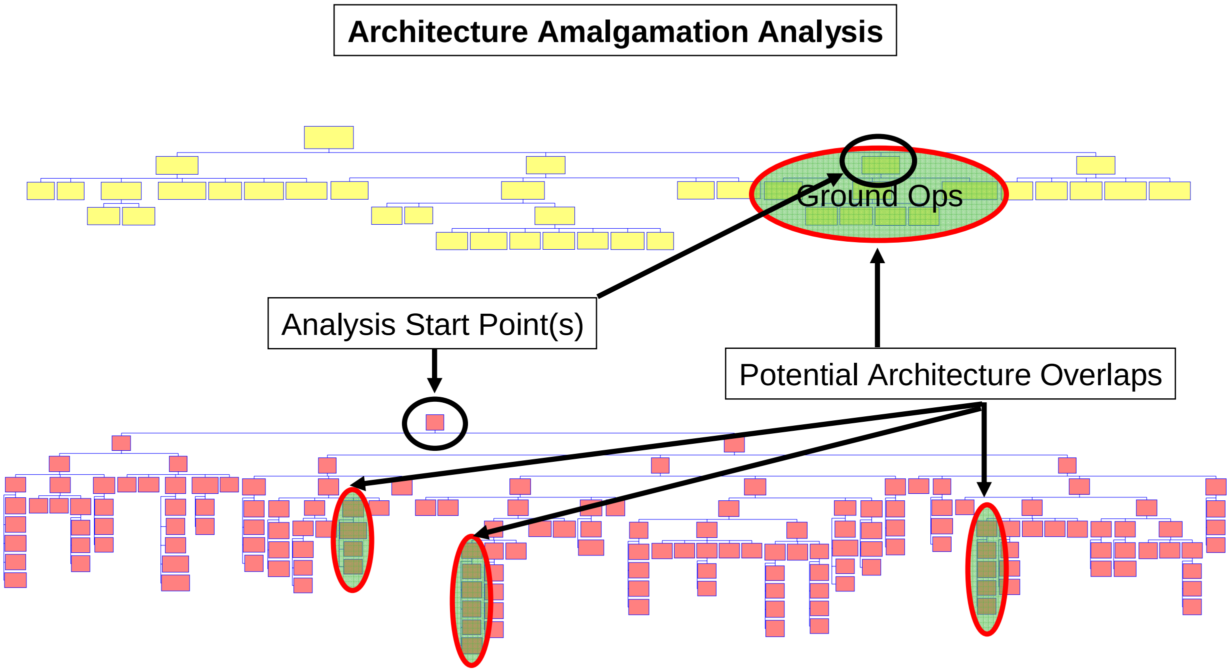

Architecture Amalgamation Analysis
Ground Ops
Analysis Start Point(s)
Potential Architecture Overlaps

Figure 8: Complex

Additional sources for primitives to be amalgamated

The repository that is used for the collection and amalgamation of JCIDS architecture data can be linked to multiple types of tools, additional sources can be utilized for the development of the primitives. These include Modeling and Simulation, Operational Testing and Evaluation, Doctrine and others. The more communities that rely on the repository as the "source of the gospel," the more current, relevant, and usable it becomes. It is a question of getting enough organizations "on board" to reach the tipping point at which the ubiquity of its use becomes an across-the-board enhancement to the enterprise.

Even when political hurdles surmounted, there's going to be the issue of security and access. Essentially what we are proposing with a system such as this is a "net centric in the slow" – tailored, filtered, and access control/security-appropriate information to specific users. The information amalgamated would be provided at certain URLs. To accomplish this, each organization (including the "unanticipated user") needs to have either a Service Level Agreement (SLA) or published documentation regarding "what to expect" from different tailored user views.

Facilities, governance, and most importantly, funding will be required to accomplish these goals. The information collected by the various DoD programs and other components needs to be federated and made available in a user friendly way, in an access control-based or Service Level Agreement-based paradigm across multiple disciplines, not just enterprise architecture. Authority must be established so that the determination of "who's right" and "who's paying for changes" to the architectures can be discussed and resolved. Even though the charter of most CIO and "-6" organizations reflects this, to this date the mandate has been unfunded, and as a result has not been accomplished. This will be discussed in the next section.

Governance Process

Why governance? According to Wikipedia, "The term ***governance*** deals with the processes and systems by which an organization or society operates."

What do we mean here when we assert that a governance process is required to support this methodology? Due to the way in which program offices are funded and staffed, there is simply no impetus for the program manager, with limited budget and resources, to adopt the use of tools such as this without a quid pro quo.

Therefore, the governance process we're advocating is a combination of "carrot and stick" – build useable tools, and the user will be inclined to use them (carrot) – codify the use of the tools in a service-level agreement between organizations, with funded, actionable roles that have decision-making power over "outliers" – the stick. Organizations won't "sign up" to the latter, without having the former in place. As a side note, OASD/NII has recognized the requirement for governance, and is in the process of releasing new guidance on this subject as this paper is being written; our suggestions here codify a means (but not necessarily *the* means) for implementation of governance of this process.

While not advocating the monolithic "architecture and everything else" data store, we believe something in-between makes sense. Thus, we're suggesting a 2-tier (or n-tier) structure in which product centers (Navy: NAVAIR, NAVSEA, etc.; AF: Air Armament Center, Aeronautical Systems Center, etc.; Army: Tank and Automotive Command, Aviation and Missile Command, Communications & Electronics Command) build and maintain the repositories containing the "enterprise by amalgamation" architectures, with a feed-in to a larger repository (i.e., Defense Architecture Registry System) that would manage and present enterprise architecture primitives and data via a *useable* interface.

This method would require some degree of service level agreement be enacted at the product centers, in which an arbiter at the product center would be assigned to review individual platform architectures, and adjudicate differences in the assertions made by each of the program offices in building their architectures. But, why would a program office “sign up” for that?

In doing so, the product center would be building a resource that could be reused across all architectures within the product center. For a concrete example, take NAVAIR or Aeronautical Systems Center, which are concerned with building air vehicles for the Navy and Air Force, respectively. If each of these, as a function of building their enterprise data store via amalgamation, could assemble “reuse modules” regarding activities, systems, nodes, etc.; each individual platform architecture would then benefit from not having to develop these from scratch. A concrete example: the architect beginning a platform architecture goes to the portal for the data store (which presupposes one exists, is user friendly, etc.), and upon identifying himself and his purpose, gets a package of “pick list seed data” for his architecture, including architecture artifacts related to air traffic control, aerial navigation, and general “fly the plane” information, all tied to the requisite doctrine.

Upon completing a build of their specific architecture, the program office would submit it up-chain to the product center for assessment. The product center assessors would then assess the architecture for cross-program interoperability and connectivity, and be able to adjudicate inconsistencies between platform-level architectures. For a concrete example, we return to our C2 platform and Tactical Fighter example and look at information exchanges asserted by each in their respective architectures:

- C2 Platform, to/from Tactical Fighter:
  - To (from Tactical Fighter): Air_Tracks, Surface_Tracks
  - From (to Tactical Fighter): Air_Tracks, Surface_Tracks
- Tactical Fighter, to/from C2 Platform:
  - To (from C2 Platform): Air Track, Ground Track
  - From (to C2 Platform): Air Track, Surface Track, PPLI

The product center assessor would need to (at a minimum), facilitate the reconciliation of number and type of exchanges between the two platform architectures. We’ve already discussed “scrubbing vs. bridging” earlier – this merely provides the means to get agreement across disparate architectures of information being sent, such that one can chain together multi-platform architectures for analysis purposes.

As they are built, these groups of “pick list seed data” can be submitted up-chain to DARS or other service-level (as in Army, Navy, Air Force, Marines), enterprise-level architecture data stores for further reuse.

Implications:

- Wherever the “pick list seed data” are amalgamated, serviced, and offered to the user (i.e., product center, service [Army, Navy, AF, Marine] architecture repositories, or DARS), the interface for submitting and retrieving the artifacts must be user friendly. There’s no way of sugar-coating this one: current and previous implementations of reuse libraries (DoD Data Standardization Process, DoD MetaData Registry, DARS, etc.) are “user-hostile.” If the user cannot find information applicable to the task he’s trying to accomplish within about 5 minutes of logging on to the portal (which implies a management process in-and-of itself, including management of profiles, and classes of user…), then the tool simply won’t be used.
- It must be noted once again that the assessor needs to be the “rare breed” that understands both operations and technology – this person cannot be someone “right out of school.”
- Implementation needs to be properly funded. This is a “pay now, or pay more later” alternative. Engineers know that paying to get the design correct up-front saves money in the long run; this is counter-intuitive to financiers, who want to delay all expenses for as long as possible.
- One might argue that the DoD MetaData Registry process, already is providing the facility to accomplish what we’re speaking of here. User-hostileness of the interface aside, “the way it’s supposed to work” is Communities of Interest (COIs) are supposed to build and maintain the structures and formats of information passed within and between COIs. However, there is no COI management process with the DoD; case in point - COIs in the DoD Metadata Registry do not match the COIs within DARS, and don’t match the COIs listed in the Global Information Grid Enterprise Services website - if these don’t match, who’s in charge…?
- In order for a process such as this to work, there needs to be a fully-understood-by-the-program-office Data Administration and Stewardship process put into place. This process would need to contain an agreement similar to service level agreements which state agreed upon responsibilities between program offices and an overarching management process. This would enable an overall data administration process to begin in earnest.

Application to Other Fields

While DoDAF architectures are mandated in support of the JCIDS process and Information Support Plans, the use of architecture products extends much further. One example is found in the modeling and simulation process supporting system development.

Design and development of information infrastructures, processing systems, and communication networks includes the specification of many complex communication mechanisms and protocols whose logical behavior must be verified and whose performance must be evaluated, in order to detect and correct conceptual errors and to minimize overall risk and cost. In this context, at the conceptual level, it is essential to use cost-effective formal description and Modeling and Simulation techniques that will enable inferences to be drawn about the future behavior of the systems being designed and developed.

The modeling, analysis, and performance evaluations of Command, Control, Communications, Computers and Intelligence (C4I) systems and large-scale wireless mobile networks require the use of concise, formal modeling and description techniques that can (1) simultaneously represent configurations of large numbers of components and behavioral mechanisms like parallelism, competition, synchronization by message exchange, and time constraints; (2) allow qualitative analysis to verify properties such as boundedness, liveness, sequence of events, and temporal relation between two events; and (3) allow quantitative analysis to obtain measures of effectiveness (e.g., performance evaluation, capacity planning, dependability, reliability, fault tolerance, removal, and forecasting).

As illustrated in Figure 9, the starting point for defining much of the required data to support modeling and simulation is the development of well formed system architecture. As depicted in the following diagram, the artifacts of the system architecture process are incorporated into the simulation Description File and provides the basis for the simulation.

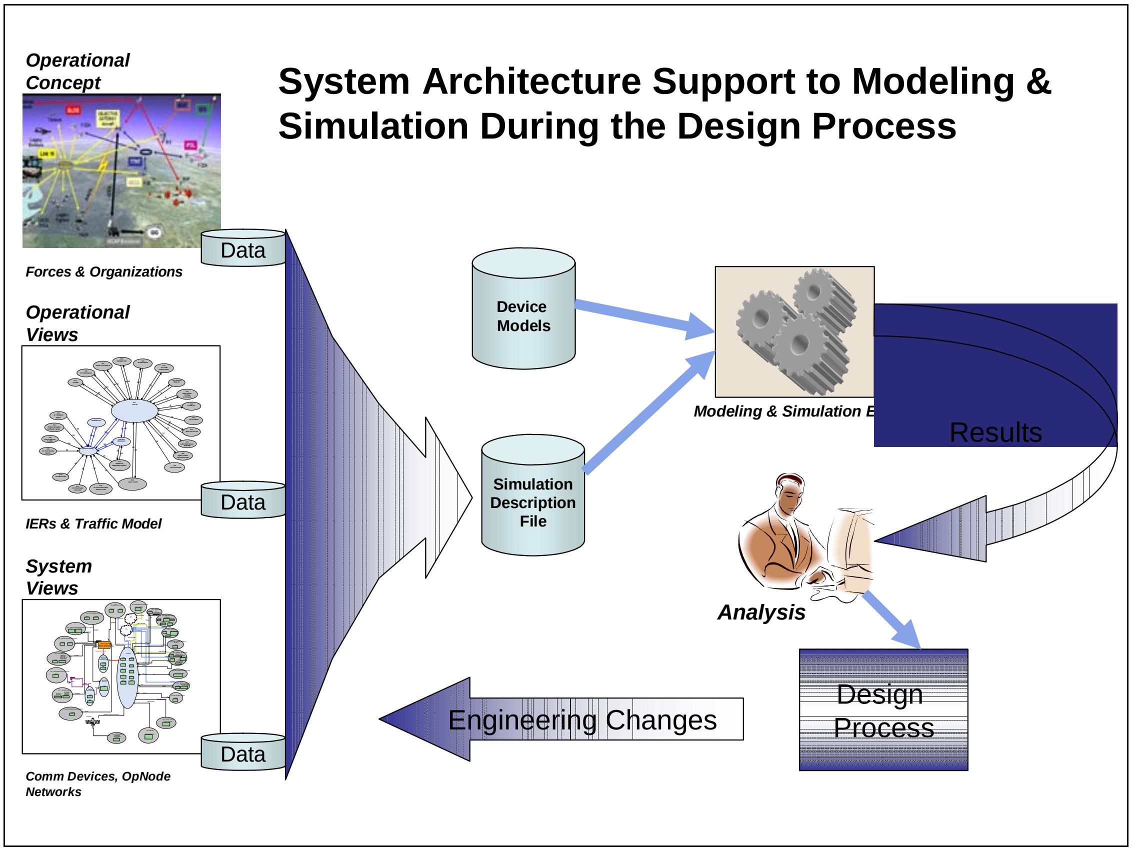


Figure 9: System Architecture Support to Modeling and Simulation

Other Useful Work on This Concept

Other groups and organizations are currently exploring the concepts discussed in this paper as well. Some are focusing on governance (DoD CIO), others on tools (SADIE, Front End). What we have proposed is a methodology that should be able to be used with any toolset that has the necessary technical capabilities and is used with the necessary governance. A list of some (but not all) of the related work follows.

Steve Ring and others of the MITRE Corporation continue to do work and research on the use of architectures for other DoD processes. The "Integrated Architecture-Based Portfolio Investment Strategies" was presented at the 10th annual ICCRTS in 2005. Further information on this and their work on 'executable' architectures can be found at the MITRE website (http://www.mitrecorp.org/news/the_edge/).

The office of the DoD CIO is putting forth an effort to direct the governance of DoD architectures. The recently released draft version of the DoD Enterprise Architecture Federation Strategy (04 December 2006) is the first of a number of documents that will outline how OASD/NII intends to administer the relationships of federated architectures.

The Defense Modeling and Simulation Organization, among others in the M&S community, are actively involved in creating Data Interchange Formats (DIF) for exchange of data among simulation engines. A DIF defines how data will be exchanged between applications; it is a formal specification of the structure and format of data to be interchanged between data producers and consumers. A DIF is specified sufficiently so that tools and applications can be developed which may support the specific subject area to exchange information which both for data input and output. A DIF as the following characteristics:

- Independent of the communications method used to exchange information
- Encourages the development of related tools and applications that can be easily integrated with existing tools and data
- Becomes the blueprint for mapping the data elements from one database to another

For more information on DIFs, see the Defense Modeling and Simulation website: https://www.dmso.mil/.

The SYSCOM Architecture Development & Integration Environment (SADIE) has been developed and maintained by SPAWAR. SADIE can be generally thought of as an application service, where SPAWAR and the SADIE program are providing fee based access to a number of enterprise architecture modeling and integration tools (the Enterprise Elements Repository, the Telelogic System Architect tool suite [multiple versions], Telelogic DOORS, Microsoft Access, and the Citrix MetaFrame Conferencing Manager) to allow collaborative development and configuration management of DoDAF architectures for the Navy Systems Commands. This suite of modeling, repository, and management tools is in essence an enterprise resource that allows for collaborative development, configuration, analysis and reporting on the enterprise. While there is no publicly available website, information about SADIE can be found in “SWEA SADIE 101 rev2.ppt”, written by Paul W. Johnson of SYS Technologies and at promotional booths that SPAWAR sponsors at events like the DoD Enterprise Architecture Conference held October 2006 in Williamsburg, VA. The actual role based access to SADIE is found here: https://sadie.spawar.navy.mil.

Conclusion

The need for enterprise level architectures is intuitive. Everyone understands the need to bring platform level architectures together in larger environments so we can manage the interoperability process and understand the system-wide implication of changes. The best way to accomplish this is, however, not well understood. Some have suggested that a well structured data dictionary will solve the architecture integration problem. Unfortunately, the idea of common data dictionary has been around for more than thirty years without reaching success; and we have no reason to believe we will see success in this area within the next thirty years.

In this paper we have suggested an approach to addressing the problem; we have not suggested a definitive solution. We believe substantial research remains in this area. The research needs to address, concepts for integrating architecture, tools required to support the methodology and governance required to manage the process.

Appendix A – References

“An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures”; C2 Assessment Tools and Metrics Track; S.J. Ring, D. Nicholson, J. Thilenius, S. Harris; 2004 Command and Control Research and Technology Symposium – The Power of Information Age Concepts and Technologies.

“An Activity-Based Methodology.ppt”; S. Harris, S.J. Ring, D. Nicholson, J. Thilenius; March 2003.

Chairman of the Joint Chiefs of Staff Instruction 6212 01D: 8 March 2006.

“Common Activities in Data Interchange Format (DIF) Development”; Peggy D. Gravitz, Jack Sheehan, Thom McLean.

Department Of Defense Instruction 4630.8; June 30, 2004.

DoD Enterprise Architecture Federation Strategy, Draft Version 1.01; 04 December 2006.

“Governance”; *Wikipedia, the free encyclopedia*; 26 Dec. 2006; http://www.reference.com/browse/wiki/Governance.

“Integrated Architecture-Based Portfolio Investment Strategies”; Assessment, Tools, and Metrics; S. J. Ring, B. Lamar, J. Heim, E. Goyette; 2005 International Command and Control Research and Technology Symposium – The Future of C2, June 2005.

“Joint C4I Interoperability” (Cartoon), Jim Klossner, Federal Computer Week, 1105 Media, Inc.

“SWEA SADIE 101 rev2.ppt”; Paul W. Johnson, SYS Technologies; June 2005.
Chairman of the Joint Chiefs of Staff Instruction 6212 01D; 8 March 2006.

Appendix B - Acronym List

ABM: Activity Based Methodology
AF: Air Force
AV: All Views
AWACS: Airborne Warning and Control System
BPMN: Business process Modeling Language
C2: Command and Control
C4I: Command, Control, Communications, Computers, & Intelligence
C4ISR: Command, Control, Communications, Computers, Intelligence, Surveillance, & Reconnaissance
CADM: C4ISR Core Architecture Data Model
CAIV: Cost as an Independent Variable
CDD: Capability Development Document
CIO: Chief Information Officer
COIs: Communities of Interest
CPD: Capability Production Document
DARS: DoD Architecture Registry System
DIFs: Data Interchange Formats
DISRonline: Defense Information Systems Registry online
DoD: Department of Defense
DoDAF: DoD Architecture Framework
DoN CSFL: Department of the Navy Common Systems Functions List
DOTMLPF: Doctrine, Organization, Training, Materiel, Leadership, Personnel, and Facilities
GIG: Global Information Grid
IAW: In Accordance With
ICD: Initial Capability Document
IDEF: Integrated DEFinition methods
ISP: Information Support Plan
IT: Information Technology
JCIDS: Joint Capabilities Integration and Development System
JFCs: Joint Functional Concepts
JOCs: Joint Operational Concepts
KPP: Key Performance parameter
LISI: Levels of Information Systems Interoperability
M&S: modeling and simulation
NAF: NATO Architecture Framework
NATO: North Atlantic treaty Organization
NCOW RM: Net-Centric Operations and Warfare Reference Model
NR-KPP: Net-Ready Key Performance Parameter
NSS: National Security Systems
OASD/NII: Office of the Assistant Secretary of Defense/Network & Information Integration
OASIS: Organization for the Advancement of Structured Information Standards
OMG: Object Management Group

OT&E: Operational Testing and Evaluation
OV: Operational View
PfM: Portfolio Management
RSTA: Reconnaissance, Surveillance, and Target Acquisition
SADIE: SYSCOM Architecture Development & Integration Environment
SDD: System Development and Demonstration
SLA: Service Level Agreement
SPAWAR: Space & Naval Warfare Systems Command (US Navy)
SV: Systems View
SYSCOM: Systems Command
TV: Technical Standards View
UA: Unit of Action
UJTL: Universal Joint Task List
UML: Unified Modeling language
W3C: World Wide Web Consortium