2004 Command and Control Research and Technology Symposium
The Power of Information Age Concepts and Technologies

C2 Assessment & Tools, #077

An Activity-Based Methodology* for Development and Analysis of Integrated DoD Architectures - *"The Art of Architecture"*

Steven J. Ring
The MITRE Corporation
202 Burlington Rd
Bedford, MA 01730
781-271-8613
sring@mitre.org

Dave Nicholson
The MITRE Corporation
7515 Colshire Drive, McLean VA 22102
703-883-3022
dnichols@mitre.org

Jim Thilenius
The MITRE Corporation
7515 Colshire Drive, McLean VA 22102
703-801-8912
jethilen@mitre.org

Stanley Harris
Lockheed-Martin Corporation
5290 Shawnee Road
Alexandria, VA 22312
703-916-7340
Stanley.harris@lmco.com

* Activity-Based Methodology is a concept developed by
The MITRE Corporation and Lockheed Martin, Copyright © 2003

An Activity-Based Methodology* for Development and Analysis of Integrated DoD Architectures - *"The Art of Architecture"*

C2 Assessment & Tools, #077

Steven J. Ring	**Dave Nicholson**	**Jim Thilenius**	**Stanley Harris**
The MITRE Corporation	The MITRE Corporation	The MITRE Corporation	Lockheed-Martin Corporation
202 Burlington Rd Bedford, MA 01730	7515 Colshire Drive, McLean VA 22102	202 Burlington Rd Bedford, MA 01730	5290 Shawnee Road Alexandria, VA 22312
781-271-8613	703-883-3022	703-801-8912	703-916-7340
sring@mitre.org	dnichols@mitre.org	jethilen@mitre.org	Stanley.harris@lmco.com

Abstract

This paper describes the Activity-Based Methodology (ABM) that establishes a common means to express integrated DOD architecture information consistent with intent of DoD Architecture Framework (DoDAF), Joint Capabilities Integration and Development System (JCIDS) Process, and the Clinger-Cohen Act. The methodology consists of a tool-independent approach to developing fully integrated, unambiguous, and consistent DODAF Operational, System, and Technical views in supporting both "as-is" architectures (where all current elements are known) and "to-be" architectures (where not all future elements are known). ABM is based on a set of DoDAF Operational Architecture (OA) and System Architecture (SA) elements symmetrically aligned to each other from which four Operational and four System architecture elements provide the core building block foundation of an integrated architecture. ABM enables architects to concentrate on the *Art of Architectur*e – that is identifying the core architecture elements, their views and understanding how they are all related together. The associations between these core elements form the basis of an integrated architecture data specification model. From these core elements, several DoDAF architecture elements are rendered and several DoDAF products are generated. ABM facilitates the transition from integrated "static" architectures to executable "dynamic" process models. Workflow steps in creating an integrated architecture, the art of architecting, are detailed. Numerous DoDAF integrated architecture data analysis strategies are presented along with mapping of ABM to warfighting DOTMLPF domains.

Introduction

The DoD Architecture Framework (DoDAF) provides the basis for developing and presenting architecture descriptions in a uniform and consistent manner. It's purpose is to ensure that architecture descriptions developed by DoD commands, services and agencies contain related operational, systems, and technical views, and that the architecture descriptions can be compared and related across organizational boundaries, including Joint and multi-national. To accomplish this, the framework defines twenty-six products to capture specific architectural views.

Product	Architecture Product
AV-1	Overview and Summary Information
AV-2	Integrated Dictionary

Product	Architecture Product
TV-1	Technical Architecture Profile
TV-2	Standards Technology Forecast

Table 1. All Views and Technical Architecture View Products

Product	Architecture Product
OV-1	High-Level Operational Concept Graphic
OV-2	Operational Node Connectivity Description
OV-3	Operational Information Exchange Matrix
OV-4	Organizational Relationships Chart
OV-5	Operational Activity Model
OV-6a	Operational Rules Model
OV-6b	Operational State Transition Description
OV-6c	Operational Event-Trace Description
OV-7	Logical Data Model

Product	Architecture Product
SV-1	Systems Interface Description
SV-2	Systems Communications Description
SV-3	Systems-Systems Matrix
SV-4	Systems Functionality Description
SV-5	Operational Activity to Systems Function Traceability Matrix
SV-6	Systems Data Exchange Matrix
SV-7	Systems Performance Parameters Matrix
SV-8	Systems Evolution Description
SV-9	Systems Technology Forecast
SV-10a	Systems Rules Model
SV-10b	Systems State Transition Description
SV-10c	Systems Event-Trace Description
SV-11	Physical Schema

Table 2. Operational and System View Products

Integrated Architectures

However, before you can use architecture descriptions for any analysis purposes you must first start with an architecture that is integrated, unambiguous, and consistent. There are three definitions of an integrated architecture. First, a single architecture description is defined to be an *integrated architecture* when products and their constituent architecture data elements are developed such that architecture data elements defined in one view are aligned with architecture data elements referenced in another view.[1] By alignment we mean that there is a set of symmetric OA and SA architecture elements that have similar meanings, associations/ relationships, properties, and characteristics. A subset of DoDAF products make up the foundation of an *integrated architecture* and consists of AV-1, AV-2, OV-2, OV-3, OV-5, SV-1, and TV-1 at a minimum. In ABM, OV-4, SV-4, and SV-5 have been added as additional products necessary for an integrated architecture. The SV-5 product, in mapping OV-5 activities to SV-4 System Functions, enables integrated OA and SA Views within a single architecture.

For the second definition, an *integrated architecture* can be defined among multiple architectures when similar or related single architectures, each based on the same set of DoDAF integrated products and constituent aligned architecture elements, can be combined for further development and analysis purposes. For the third definition, in a recent memo published by Deputy Secretary of Defense [2], an *integrated architecture* has been defined as an architecture consisting of different views or perspectives (operational view, system view, and technical view) that facilitates integration and promotes interoperability across Family-of-Systems/System-of-Systems and compatibility among related mission area architectures. All three definitions are consistent and in agreement with each other.

Integrated architectures usually have associated with them a time frame, whether by specific years (e.g., 2005-2010) or by designations such as “as-is”, “to-be”, “transitional”, “objective”, “epoch”, etc. In all cases, this reduces to either *inventories* of current capability (“as-is”) or *blue-prints* of future capability (“to-be”) based on some future need or objective.

Domain experts, program managers, and decision-makers need to be able to analyze these architectures to locate, identify, and resolve definitions, properties, facts, constraints, inferences, and issues both within and across architectural boundaries that are redundant, conflicting, missing, and/or obsolete. The analysis must also be able to determine the effect and impact of change (“what if”) when something is redefined, redeployed, deleted, moved, delayed, accelerated, or defunded. In most “as-is” architectures, details about activities, nodes, roles, systems, etc. are fully known and architectures analysis can be readily accomplished.

Unlike “as-is” integrated architecture, the present approach to developing “to-be” integrated architectures and their analysis does not fully enable them to be used for true system engineering purposes to discover future enterprise rules, patterns, practices, relationships, and system and organizational requirements. That is because not all architecture details are known resulting in architecture descriptions that are based on unknowns and abstract elements. By examining aggregations and clusters of activities, nodes, roles, systems, etc and by performing gap analysis and assessments (i.e., which activities are not performed by any roles), new system and organizational requirements can be derived. This would, in turn, support justifications for future funding decisions of new systems, their elements, their components, and their supporting operational organizations.

Activity-Based Methodology

A new paradigm for architecture development, ***Activity-Based Methodology (ABM)***, was developed to establish a common means to express integrated architecture information consistent with intent of DoDAF, CADM, Joint Capabilities Integration and Development System (JCIDS) Process and Clinger-Cohen Act. It consists of a tool-independent approach to developing fully integrated, unambiguous, and consistent DODAF views in supporting both “to-be” architecture and their gap analysis while still providing for “as-is” architectures and their analysis.

ABM uses a data centric approach for architecture element and product rendering instead of a product centric approach. A data centric approach supports cross-product relationships based on an integrated core set of architecture building block element primitives. These, in turn, enables several architecture elements to be automatically generated and several architecture products to be automatically rendered. ABM was designed to also capture sufficient representations of “static” activity/ information flow architectures models to transition them to “dynamic” executable process models for analysis of operational and system behavior over time and their related costs.

The ***Activity-Based Methodology*** is based on six principles:

1) There exists a set of symmetrically aligned OA and SA elements divided into three object classes: entities, relationships, and attributes
2) Four OA and four SA object entities provide the core foundation building block primitives of an an integrated architecture
3) The associations between the OA/SA sets of core primitives are represented by a triple three-way set of relationships defined in an integrated architecture data specification model
4) Architecture entities are manually entered once from a specific Framework product

5) Several DoDAF relationship and attribute architecture object classes (e.g., Information Exchanges) can be automatically formed from core entities
6) Two DODAF products can be totally rendered graphically (e.g., OV-2, SV-1) and two can be totally rendered as report documents (e.g., OV-3, SV-6)

Principle #1 – Symmetric Alignment of OA/SA Architecture Objects

OA and SA constituent architecture elements are symmetrically aligned with each other. This means that there are OA architecture elements corresponding to similar SA architecture elements. For example, Operational Activities are aligned with System Functions, Operational Nodes with System Nodes, etc. These architecture elements can be thought of as architecture objects and divided into three object classes: entities, relationships, and attributes. In following an E-R-A approach to architecture objects, **E**ntity objects are the objects about which architecture data is collected, **R**elationship objects are the associations between entity objects, and **A**ttribute objects identify characteristics of entity and relationship objects.

On the OA side, Information, Activities, Nodes, Roles, Processes, and CONOPS represent the primary architecture objects. Need Lines represent associations between Information, Activity and Node entities with the Information Exchange providing the attributes of Need Lines. Organizations (Org) represent associations between the Role entity objects and the Knowledge, Skills, and Abilities (KSA) attributes of the Roles. Similar associations exist on the SA side.

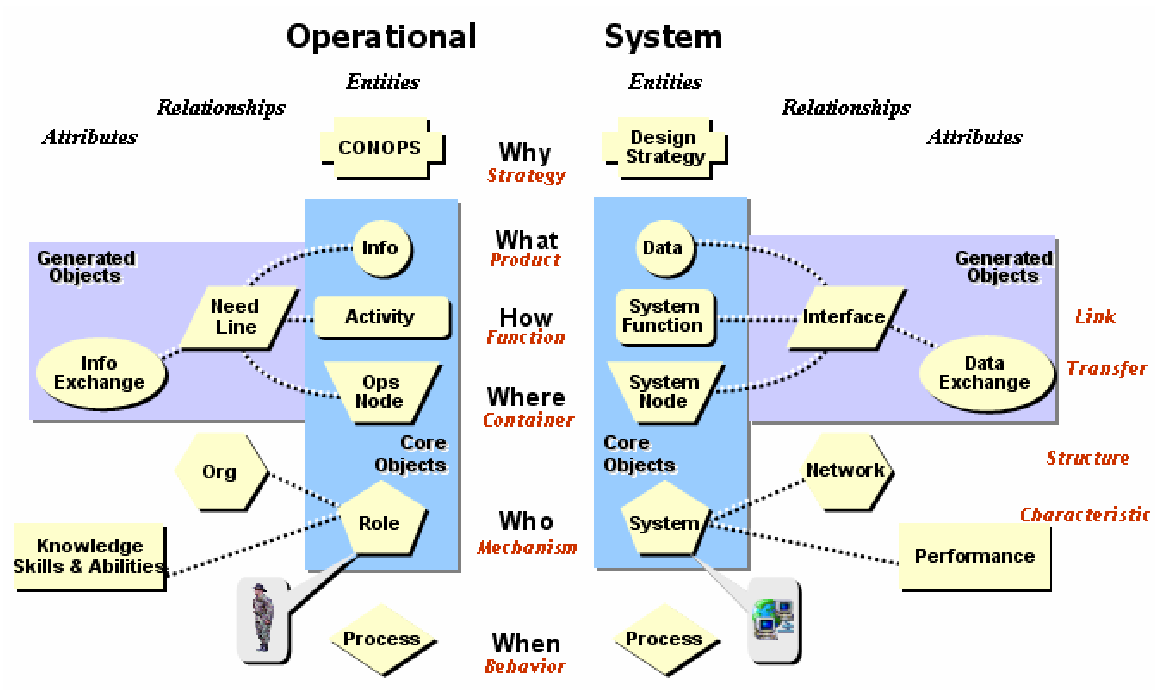

Figure 1: Symmetrically Aligned OA/SA Architecture Objects

Principle #2 –Core OA/SA Entity Objects

Four primary object entities in each view are considered as *core* – i.e., those building block primitives that make up the foundation of an integrated architecture.

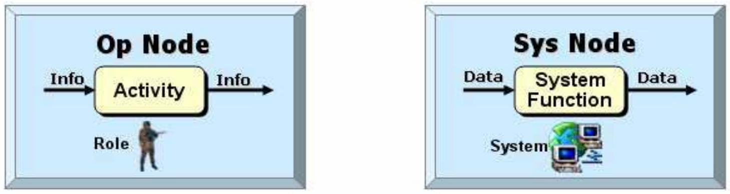

Figure 2: Core DoDAF Architecture Entities

Activities (System Functions)

These represent the actions by which input (I) Information (*Data*) is consumed in being transformed to output (O) Information (*Data*). Activities can be decomposed to sub-activities.

Operational (System) Node

These represent the collection of similarly related Activities (*System Functions*) usually at a place or location. Operational Nodes may, optionally, represent the collection of activities performed by an organization, organization type, logical or functional grouping where activities are performed. Nodes do not represent operational/ human roles - Roles represent Roles. Nodes can be decomposed to sub-nodes.

Role (System)

These are the means by which an Activity (*System Function*) is performed, processed or executed. Roles are resources, characterized by a set of Knowledge, Skills and Abilities (KSA) assigned to humans and are analogous to job titles or job responsibilities. Systems are material resources and are described in terms of their performance characteristics. Roles and Systems are grouped together into a collection that represents a physical organization or a requirement for an organization. Systems can be decomposed into sub-systems but Roles can not be decomposed any further.

Information (Data)

These are formalized representations of data subject to a transformation process and are the inputs and outputs of Activities (*System Functions*). Information (*Data*) can be decomposed into its component items such that, at higher levels of an activity model, an input/output can be considered as a “bundle” or “pipeline” while at the lower levels the input/outputs consists of the “bundled/ pipelined” component data items. For example, “Weather Data” could be made up of “Temperature” and “Humidity” so that “Weather Data” is produced/ consumed at the higher activity levels but that “Temperature” and “Humidity” are separate inputs and outputs at the lower levels. Bundled information is usually graphically depicted as “branchs/ joins”.

Principle #3 – Architecture Data Specification Model

The associations between the OA/SA sets of core primitives are the basis of an integrated architecture data specification model. They are all related to each other such that:

- Each Activity (*System Function*) that produces and consumes information (*Data*) is performed at an Operational (*System*) Node by a Role (*System*)
- Each Operational (*System*) Node contains a Role (*System*) that performs an Activity (*System Function*) that produces and consumes Information (*Data*)
- Each Role (*System*) in an Operational (*System*) Node performs an Activity (*System Function*) that produces and consumes (*Data*)
- Information (*Data*) is produced from and consumed by Operational Activities (*System Functions*) performed by Roles (*Systems*) at Operational (*System*) Nodes

The relationships between them can be represented by a triple set of three-way relationships:

1. Operation Node • Activities • Roles
2. System Functions • System Nodes • Systems
3. Organizational Units • Roles • Systems

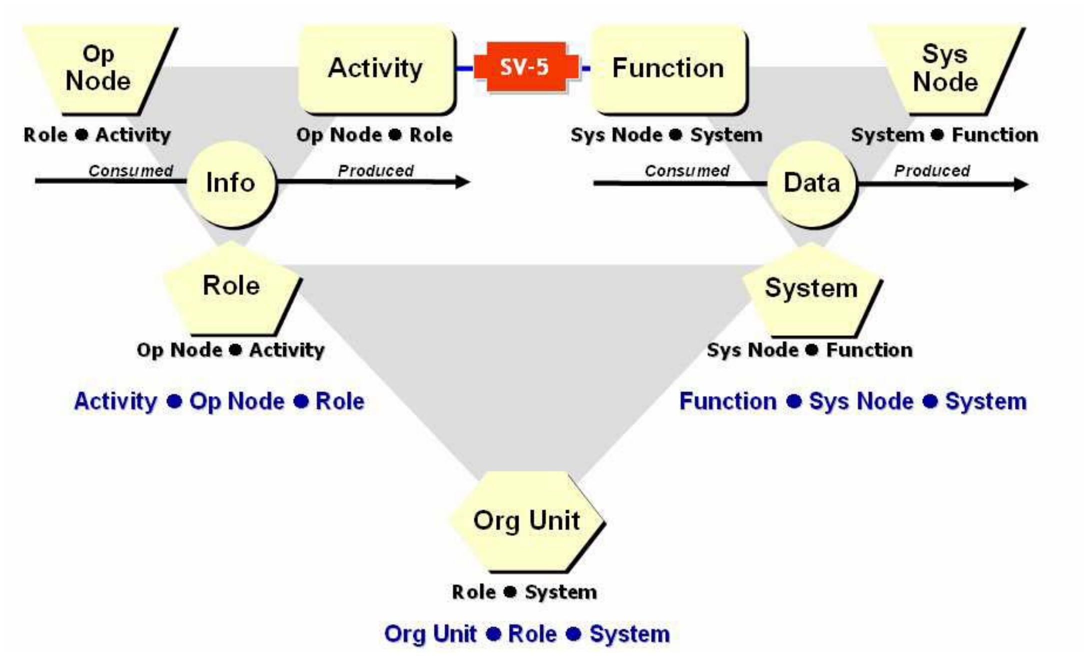

Figure 3 Core Architecture Entity Relationships

In the three-way association between the OA core primitives, the intersection of the association between an Operational Activity and an Operation Node is a Role. Likewise, in the three-way association between the SA core primitives, the intersection of the association between a System Function with a System Node is a System. The intersection of the association between a Role and a System is the Organizational Unit. The association of Organizational Units with Roles already exists in DoDAF OV-4. ABM establishes two additional associations of Organizational Units with Systems and Roles with Systems.

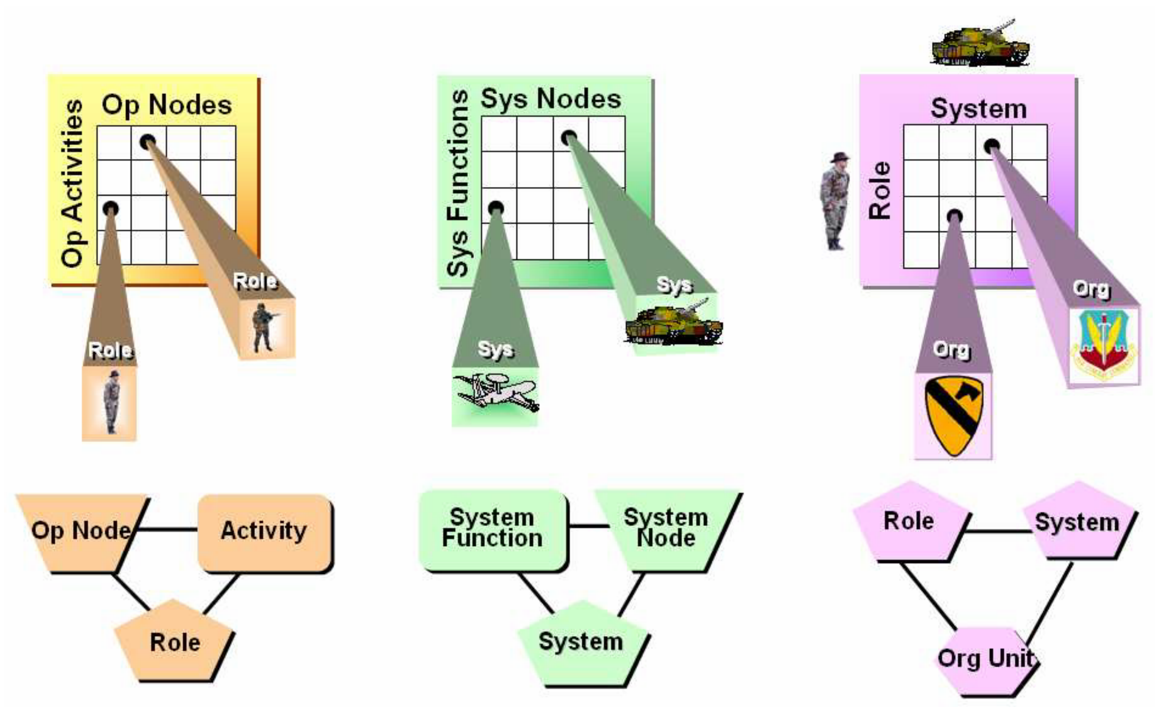

Figure 4 Triple Associations of Core Architecture Objects

The symmetric alignment of DoDAF data centric architecture objects and their three sets of triple associations can all be related together in an architecture data specification model. This Architecture Data Specification Model (ADSM) consists of a set of formal object class specification models for each of the DoDAF products and all their constituent objects.

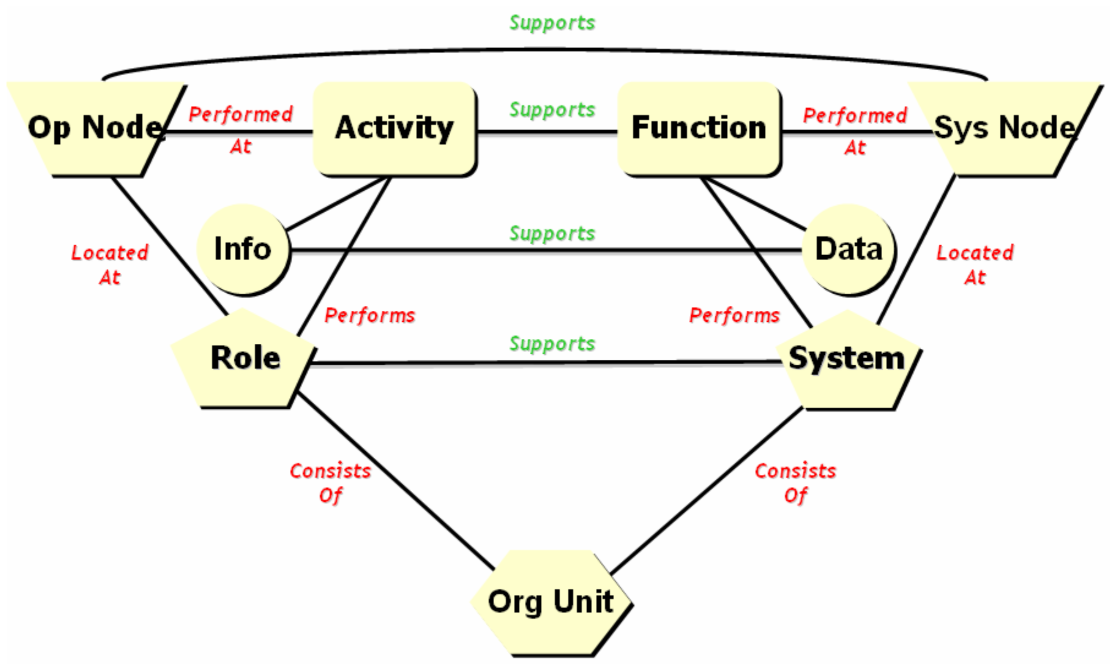

Figure 5 Architecture Data Specification Model (ADSM)

As important as it is to understand the relationships that exist between the architecture data elements, it is equally important to understand the relationships that do not exist. For example, Systems are not directly related to Activities. They are related, indirectly, first to System Functions and then, via SV-5, from System Functions to Activities. Thus, from this data model, a refined definition of an integrated architecture can be seen as one that *has its data elements related to and associated with each other according to ADSM*.

Integrated OA/SA Data Architecture Analysis

In the data model it can be seen that the basic set of three-way associations is very simple and elegant yet very powerful. From this model one can obtain a much richer and more complete analysis on complex architectures. By examing different sets of relationships, various types of OA/SA architecture analysis can be obtained:

- Functional Analysis - Activities and their related Functions – the "*How*" from both views
- Nodal analysis - Activities and their Op Nodes and their relationships to Functions and their System Nodes – the "*Where*" from both views
- Product analysis – Activities at Op Nodes producing/ consuming information and their relations to Functions at System Nodes producing/ consuming data – the "*What*" from both views

- People, Material and Training analysis – Roles, Systems and their Activities and related System Functions – the "*Who*" analysis
- Timing and cost analysis - Time-dependent behavior and dollar cost analysis of complex, dynamic operations and human and system resource interactions - the "*When*" analysis (discussed later)

Mapping ADSM to DOTMLPF

Based on ADSM, warfighting DOTMLPF domains map to architecture objects as follows:

Doctrine	Activities, Roles Operational Nodes
Organization	Org Units
Training	Roles, Systems
Leadership	Org Units, Roles, Systems
Material	System Functions, Systems, System Nodes
Personnel	Roles
Facilities	Operational Nodes, System Nodes

This leads to better definitions of warfighting capabilities by being able to anticipate effects and assess impact of change on domains and by examing usage (who/ what affects something) and references (who/ what is affected by something).

Gap Analysis for "To-Be" Architectures

Note that for the OA view of *as-is* and *to-be*, activities and nodes are usually known. In *as-is* architectures, Roles are also known. However, for *to-be* architectures, in most cases Roles may or may not be known. Similarly, for the SA views of *as-is* and to-be architectures, functions and nodes are usually known. In *as-is* architectures, Systems are also known. However, for *to-be* architectures, in most cases, Systems may or may not be known. Gap-analysis of *to-be* architectures reveals:

- Orphaned Activities – that is, Activities at Nodes without Roles
- Orphaned Systems – that is, System Functions at System Nodes without Systems

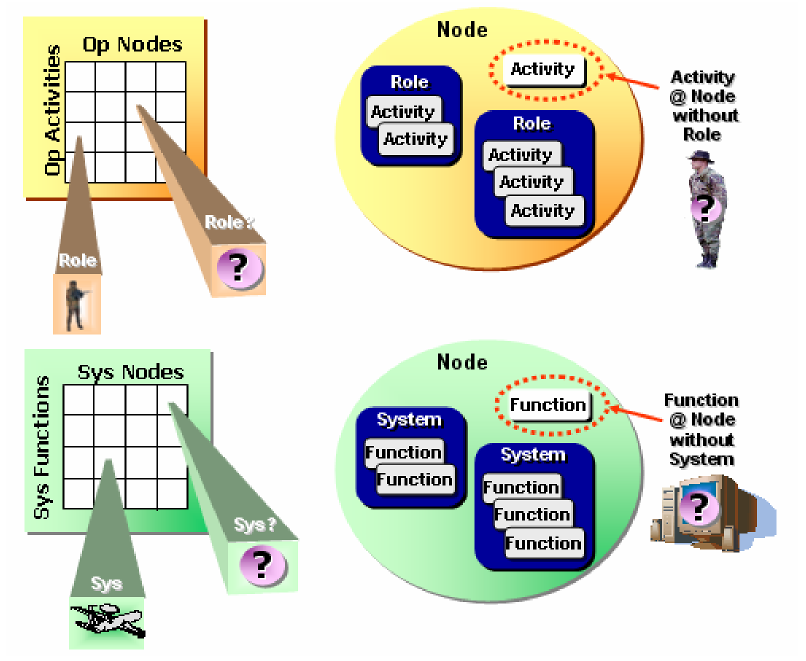

Figure 6 Gap Analysis

Based on this *to-be* gap analysis, by clustering and aggregating orphaned activities and orphaned systems, a set of requirements could be derived for a needed organizational structure and/or a needed system or, depending on how one clusters orphaned system functions, multiple needed systems.

Principle #4 - Core Architecture Entities Entered Once

Each core architecture entity object is entered from only one Framework product. For example, activities are only defined when creating an OV-5 activity model. For associations between, say nodes and activities and roles, these associations are created and managed from a three-way matrix (spreadsheet) editing facility.

Principle #5 – Automatically Forming Relationship/Attribute Architecture Objects

Several relationship and attribute architecture class objects are automatically formed from the core building block entities. Auto generating architecture data ensures data consistency, results in quality architecture products (by eliminating user inputs), and speeds up the entire architecture development process. These generated relationship and attribute architecture class objects lead to a standard, reusable collection of architecture artifacts that can be maintained at the enterprise level and can be shared by all mission area and program/node architects.

On the OA side, Information Exchanges and Need Lines are formed from OV-5 leaf activities, their information inputs and outputs, and their associations to OV-2

Operational Nodes. On the SA side, System Data Exchanges and Interfaces are formed from SV-4 leaf system functions, their data inputs and outputs, and their associations to SV-1 System Nodes. The methodology does not preclude Information Exchanges being created from non-leaf activities (parent activities) but that Need Lines (Interfaces) are only created from Information exchanges between leaf activities (functions).

Leaf activities are those lowest in an activity model that are not decomposed any further. Activity models are decomposed down to the appropriate level for the purposes of the architecture. Usually, this would be to the level where an activity is capable of being i) associated with a single operational node, ii) assigned to an individual role (person), iii) and/ or has a single input or single output. Usually, leaf activities are some combination of these three and subject to judgment calls. It is only when Need Lines are associated with Information Exchanges formed from leaf activities at different nodes, that a valid and consistent OV-2 Operational Node Connectivity diagram and OV-3 Operational Information Exchange Matrix can be obtained. The same discussion holds for leaf system functions on the SA side and SV-1 and SV-6.

While Information (System Data) Exchanges can be generated, their properties (transport times, security classification, etc.) can not be automatically filled in and, therefore, must be defined manually. This makes Information (System Data) Exchanges persistent architecture data in that, once generated and their properties defined, they can not be deleted. Need Lines, on the other hand, since they also auto generated but carry no properties, can be deleted and regenerated again as the Activity model grows and contracts with additional (or subtractive) activities, information (data) inputs/outputs, nodes, etc.

Principle #6 – Automatically Generating DoDAF Products
On the OA side, an OV-2 can be graphically rendered from Information Exchanges and their Need Lines formed from the four OA core entity objects. In addition, as many individual node-centric OV-2 diagrams can be rendered as there are nodes. An OV-3 is automatically produced since it consists entirely of the collection of Information Exchanges (and their properties) within an architecture model. Similarly, on the System side, an SV-1 can be completely graphically rendered from System Data Exchanges and their System Interfaces formed from the four SA core entity objects. Also similarly, an SV-6 can be automatically produced since it consists entirely of the collection of System Data Exchanges and their properties.

Extended OV-3/SV-6 Architecture Data Mining Analysis

Because of the triple three-way associations between the various architecture objects as defined in the ADSM, an extended OV-3 and SV-6 product can now be obtained to support gap, overlap and redundancy analysis. From/ To Roles are to OV-3 and From/ To

Systems are added to SV-6. The associations between Roles, Systems, and their Org Units are also added to obtain the following:

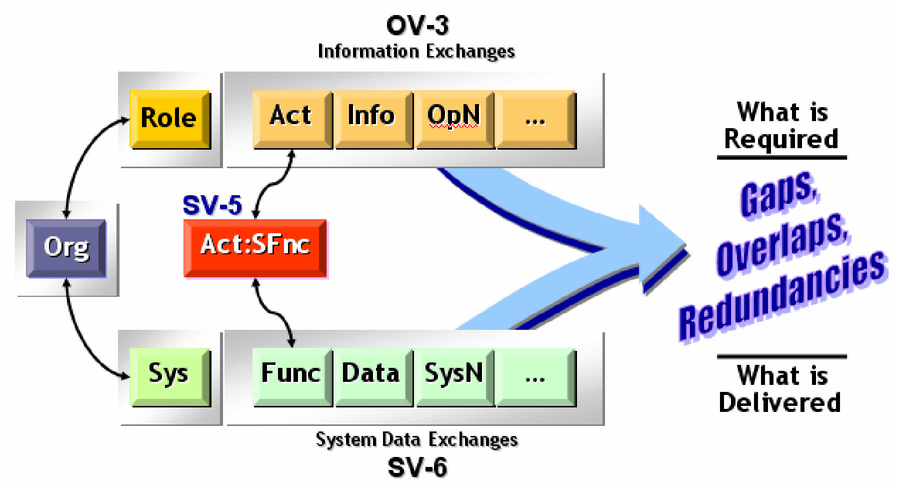

Figure 7 Architecture Data Mining with Extended OV-3/SV-6

This leads to "what if" and "if what" impact assessments between what is required and what is delivered. Based on the relationships between each individual OV-3/SV-6 elements indirectly (via SV-5) to any to other single element or (matched) collections of other elements, one could, for example, assess the impact of losing a System or a System Node on Operations (Activities, Nodes, Roles, etc). In addition, one could obtain a set of requirements for an Operational Node and a Role where such requirements would be derived from the indirect relationship (via SV-5) between Nodes and Roles to Systems, System Nodes, and System Functions.

Transition to Executable Architectures

The Activity-Based Methodology enables the transition to dynamic (over time) executable models. Executable process models enable the associated time-dependent behavior and dollar cost analysis of complex, dynamic operations and human and system resource interactions that cannot be identified or properly understood using static operational models - the "*When*" analysis[3]. Providing time and costs analysis of executable architectures derived from integrated architectures is the first step in an overall architecture based investment strategy where we eventually need to align architectures to funding decisions to ensure that investment decisions are directly linked to DOD mission objectives and their outcomes.

Static operational models only show that Activities "must be capable of" producing and consuming Information. They do not provide details on how or under what input/ output conditions information is actually produced/ consumed. They also do not explicitly identify, for each activity, the number (capacity) of Roles needed or their ordering for the case when multiple Roles perform the same activity (who operates on the first input, who operates on the second, etc). Dynamic executable models go beyond "must be capable of" and define precisely under what conditions information is actually produced/ consumed and the exact number and ordering of Roles. An executable architecture can then be defined in terms of an integrated architecture as a *dynamic model of sequenced activities/ events (concurrent or sequential) performed at an operational node by roles (within organizations) using resources (systems) to produce and consume information.*

The transition is accomplished by starting with the extracted set of leaf activities to which dynamic processing time (duration) and any statistical time distribution, average wait time before processing, continuation strategy, activity cost, and Input/ Output conditions are all defined. By connecting and chaining these leaf activities according to the Information Exchanges defined between them, we can produce candidate activity thread (scenario) models of sequenced actions. Information Exchange properties such as transport times including any statistical time distribution, quantity, and cost are already defined in OV-3. Roles and Systems are the human/ material resources used by each process and they have single/ periodic (un)availability times, set up times, capacity (quantity), processing strategies (FIFO, etc.), and hourly and fixed cost. A starting candidate dynamic process scenario model can be auto generated from an integrated architecture. The candidate model can then be completed in the sense that final behavior is modeled of exactly how inputs and outputs of each process will be consumed/ produced and any trigger inputs and outputs added.

Dynamic analysis starts by defining a measurable objective – some optimum Measure of Effectiveness (MOE). The next step is to define how to assess and analyze the MOE by identifying dynamic model attributes and properties that go into measuring the desired effect. The appropriate data necessary to measure the desired MOE is determined and the model is then simulated. The MOE can then be measured and based on how well the overall objective was met (or not met), the model can be edited, re-simulated and the MOE measured again. This repeats until an optimum MOE is reached. The executable model can also be used to assess Measures of Performance (MOP) and Measures of Force Effectiveness to determine the overall success of the organization's operations and use of resources in accomplishing it's mission.

Workflow Steps to an Integrated Architecture - the *Art of Architecture*

OA and SA development work flow each consists of 9 steps – 3 manual data entry, 1 manual association, and 5 automation as follows. (Note: the description below will be for the OA side but the same workflow holds for the SA side).

1) Create OV-5 Activity Model	1) Create SV-4 System Function Model

2) Create OV-2 Nodes	2) Create SV-1 System Nodes
3) Create OV-4 Roles & Org Units	3) Create SV-1 Systems
4) Manually triple associate Activities with Nodes and with Roles	4) Manual triple associate System Functions with System Nodes and with Systems
5) Auto form three-way associations: Activities, Nodes, and Roles	5) Auto form three-way associations: Functions, Sys Nodes, & Systems
6) Render Information Exchanges	6) Render System Data Exchanges
7) Render OV-2 Need Lines with linked OV-3 Information Exchanges	7) Render SV-1 Interfaces with linked SV-6 System Data Exchanges
8) Generate OV-3 Information Exchange Matrix	8) Generate SV-6 System Data Exchange Matrix
9) Transition to executable architecture models.	9) Transition to executable architecture models.

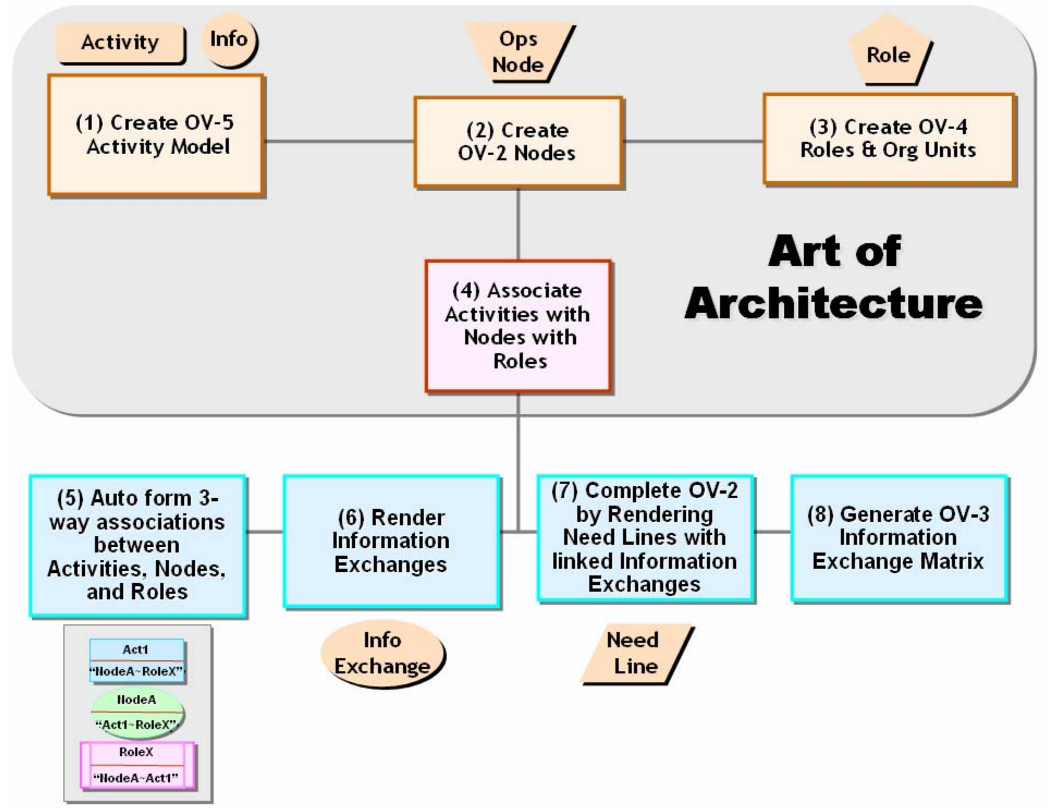
Activity
Info
Ops Node
Role
(1) Create OV-5 Activity Model
(2) Create OV-2 Nodes
(3) Create OV-4 Roles & Org Units
Art of Architecture
(4) Associate Activities with Nodes with Roles
(5) Auto form 3-way associations between Activities, Nodes, and Roles
(6) Render Information Exchanges
(7) Complete OV-2 by Rendering Need Lines with linked Information Exchanges
(8) Generate OV-3 Information Exchange Matrix
Act1
"NodeA~RoleX"
NodeA
"Act1~RoleX"
RoleX
"NodeA~Act1"
Info Exchange
Need Line

Figure 8 Workflow Steps to an Integrated Architecture

The first step in the ABM workflow is to create an OV-5 activity model and the first two of the core architecture objects – Activities and Information Inputs/ Outputs. An OV-5 should consist of a well formed, balanced, and consistent set of activities, their sub-activities and the various inputs and outputs. In ABM, an OV-5 follows some of the FIPS 183 IDEF0 conventions (parent/ child activity decomposition hierarchy, etc.) but ABM does not adhere to all of the IDEF0 conventions. This is because some conventions are incompatible with building an integrated architectures and that takes precedence.

For example, in an OV-5, while mechanism arrows and control arrows are normal parts of IDEF0, they are not used in the methodology. This does not mean that they are not identified. A different approach was taken based on the three-way association between Activities, Nodes, and Roles. Likewise, there is an association between Activities (System Functions) and Standards and Guidance that takes the place of Control Arrows. Again, the consistency of building integrated architectures takes precedence and is more important.

The next step is to create the third core object, Operational Nodes, and then, Roles, the fourth core object by developing an OV-4 Organizational Chart. At this point, the four core objects have been defined and they can be all associated together via the three-way matrix association. The will result in OV-5 activities displaying their corresponding Node-Role association, OV-2 Nodes displaying their corresponding Activity-Role association, and OV-4 Roles displaying their corresponding Node-Activity association.

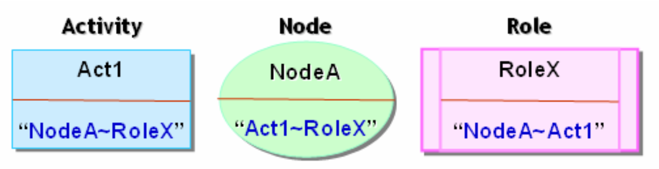

Figure 9 Three-way associations

These first four steps is what can be referred to as the ***Art of Architectur*e**. That is, understanding and identifying what the core architecture objects are and how they are all related together. If one considers the OV-2, OV-4, OV-5 products as your canvas and the core architecture elements – nodes, activities, roles, information – as your set of paints and paint brushes, then developing an architecture is much like creating a painting – i.e., *you paint your architecture*. From this point on, there is sufficient architecture data for automation to take over - generating Need Lines and their related Information Exchanges. OV-2 can now be completed by auto-connecting each Operation Node pair with their corresponding Need Line. The various properties of Information Exchanges are now defined. The set of Information Exchanges together with all their property values becomes the OV-3 product. Finally, candidate activity thread (scenario) models of sequenced actions can be auto generated from the set of the leaf activities together with their Information Exchanges.

Summary

This paper presented the Activity-Based Methodology, a tool-independent approach to developing fully integrated, unambiguous, and consistent DODAF views, and the six principles upon which it is based. An integrated architecture is the basis for all architecture assessments such as impact of change analysis and for identifying redundant, conflicting, missing, and/or obsolete architecture items. The paper explained the *Art of Architecture* and showed how several DoDAF architecture objects can be rendered and products automatically generated based on a symmetrically aligned set of four core OA and SA architecture elements. An architecture description specification model, ADSM, was presented and the mapping of ADSM to DOTMLPF domains presented. The transition from integrated architectures to executable architectures was discussed. Numerous architecture analysis strategies were presented.

In conclusion, architecture development guidance combined with compliant architecture tools and Activity-Based Methodology render integrated architectures. Integrated Architectures combined with simulation tools and scenarios render executable architectures. Together, integrated architectures, executable architectures, analytical tools and methods render quantitative actionable information, which, in turns supports funding decisions, acquisitions, system engineering, and investment strategies.

References

1. DOD Architecture Framework, V1.0, Vol. I and II, 15 August 2003.
2. Information Technology Portfolio Management memo, 22 March 2004, OSD-03246-04
3. "Applying Executable Architectures to Support Dynamic Analysis of C2 Systems", C2 Assessment & Tools, #113, T. Pawlowski, P. Barr, S. J. Ring, 2004 Command and Control Research and Technology Symposium, June 2004

Author Biographies

STEVEN J RING is a Principal Information Systems Engineer at the MITRE Corporation in Bedford, MA. He has over 3 decades experience including technical and managerial roles in commercial/military product development and integration. Mr. Ring received his BEE from Cleveland State University and MS in Systems Engineering from Case Institute of Technology. He has focused on applying information and knowledge-based repository technology to DOD architecture development and integration in support of interoperability and simulation based acquisition. He has contributed in the areas of techniques, methodologies, and tools for integrating, analyzing and validating both static and dynamic DOD architecture models. Mr. Ring is currently examing how architecture analysis can support portfolio management investment decision-making.

DAVE NICHOLSON is a Senior Principle Information Systems Engineer at the MITRE Corporation in McLean, Virginia. He spent over 20 years in Army Research, Development and Acquisition Program Management and for the last 8 years he has been leading architecture efforts with MITRE. Mr. Nicholson is a graduate of the United States Military Academy and received MS degrees from the Naval Post Graduate School in Electrical Engineering and from Stevens Institute of Technology in Technology Management. He is currently the Project Director of MITRE support to the USAF Deputy Chief of Staff, Warfighting Integration.

JIM THILENIUS is a Principal Information Systems Engineer at the MITRE Corporation in Washington D.C. He is currently the Deputy Chief Architect for the Air Force Chief Architect Office supporting the AF Chief Information Officer. For the last 7 years, he has been a lead architect with MITRE working in the domains of command and control, space operations, intelligence, surveillance and reconnaissance. Mr. Thilenius received a MS in aerospace engineering from University of Colorado and a BS in nuclear and bioengineering from the University of Illinois

STANLEY HARRIS is a Senior Systems Analyst working at Lockheed Martin in Alexandria, Virginia. Mr. Harris has twenty years experience supporting C4ISR programs providing data base development and support. The last five years he has been supporting the Command Information Superiority Architecture (CISA) Program by working on a methodology for architectures that will provide data centric continuity throughout the DoDAF Architecture products and the application of that methodology toward the applications that support creation of architectures.