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

C2 Assessment Tools & Metrics Track

An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures

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
703-983-3022
dnichols@mitre.org

Jim Thilenius
The MITRE Corporation
202 Burlington Rd
Bedford, MA 01730
703-801-8912
jethilen@mitre.org

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

An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures

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
703-983-3022
dnichols@mitre.org

Jim Thilenius
The MITRE Corporation
202 Burlington Rd
Bedford, MA 01730
703-801-8912
jethilen@mitre.org

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

Abstract

This paper describes the Activity Based Methodology that establishes a common means to express integrated DOD architecture information consistent with intent of DoD Architecture Framework (DoDAF) 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). It is based on a set of DoDAF Operational and System architecture elements aligned to each other from which four Operational and four System architecture elements provide the core building block foundation of integrated architectures. An integrated architecture is the basis for any type of subsequent architecture analysis for any purposes such as impact analysis and for identifying redundant, conflicting, missing, and/or obsolete architecture elements. The DoDAF Architecture Description Specification Model (DADSM), derived from the aligned DoDAF architecture elements, will be presented as well as the mapping of DADSM to military DOTMLPF. Workflow steps to creating integrated DoDAF operational and system architecture descriptions will be described. The Activity Based Methodology also enables the transition to executable process models and their 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 models.

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-seven products to capture specific architectural views.

Product	**Architecture Product**
AV-1	**Overview & summary**
AV-2	**Integrated Dictionary**
AV-3	Capability Maturity Profile

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 Concepts
OV-2	**Operational node Connectivity Description**
OV-3	**Operational Information Exchange Matrix**
OV-4	***Organizational Relationship Chart***
OV-5	**Activity Models**
OV-6a	Operational Rules Model
OV-6b	Operational State Transition Description
OV-6c	Operational Events/Trace Description
OV-7	Logical Data Model

Product	**Architecture Product**
SV-1	**Systems Integration Description**
SV-2	Systems Communication Description
SV-3	Systems Matrix
SV-4	***Systems Function Description***
SV-5	***OA to System Function Traceability Matrix***
SV-6	System Data Exchange Matrix
SV-7	Systems Performance Parameter Matrix
SV-8	System Evolution Description
SV-9	System Technology Forecast
SV-10a	Systems Rules Model
SV-10b	Systems State Transition Description
SV-10c	Systems Event/Trace Description
SV-11	Physical Data Model

Table 2. Operational and System View Products

Integrated Architectures

However, before you can use architecture descriptions for any type of analysis purposes you must first start with an architecture that is integrated, unambiguous, and consistent. An architecture description is defined to be an *Integrated Architecture* when DoDAF products and their constituent architecture objects are developed such that architecture objects defined in one view are aligned with architecture objects referenced in another view. A subset of these framework 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 the Activity Base Methodology, 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 Operational and System Views within a single architecture. Integrated architectures can also be defined between and among multiple architectures when each single architecture is based on the same set of DoDAF integrated products and constituent architecture aligned objects.

Integrated architectures being developed in accordance with the DoDAF 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***, was developed to establish a common means to express integrated architecture information consistent with intent of DoDAF, CADM 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. The Activity Based Methodology 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 core set of architecture building block elements and enables several architecture objects to be automatically generated and several architecture products to be automatically rendered. The Activity Based Methodology 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 five principles:

1) There exists a set of aligned Operational and System Architecture objects divided into 3 object classes: entities, relationships, and attributes
2) Four Operational and four System Architecture entities provide core foundation building blocks of an an integrated architecture
3) Architecture entities are manually entered once when creating a specific Framework product
4) Several DoDAF relationship and attribute architecture object classes (e.g., Information Exchanges) can be automatically formed from core entities
5) Two DODAF products can be rendered graphically (e.g., OV-2, SV-1) and two can be rendered as report documents (e.g., OV-3, SV-6)

Principle #1 – Alignment of Operational/System Architecture Objects
The set of Operational and System constituent architecture objects are aligned with each other. For example, Operational Activities are aligned with System Functions, Operational Nodes with System Nodes, etc. These architecture objects are 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.

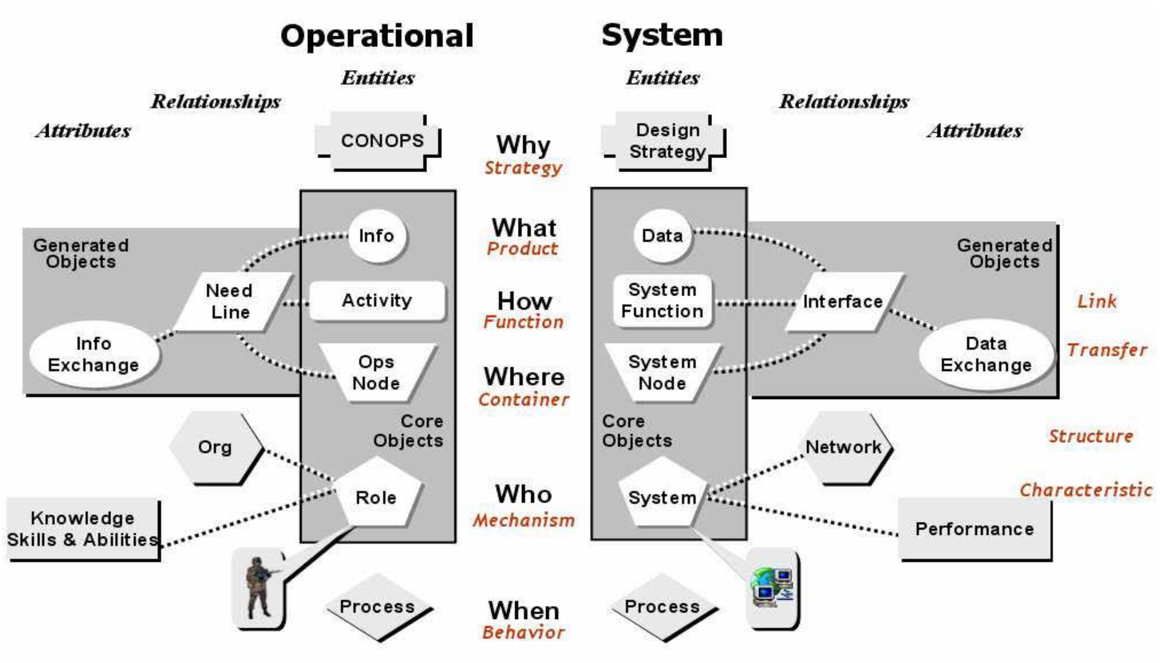


Figure 1: Aligned Operational and System Architecture Objects

Thus, on the Operational side, Information, Activities, Nodes, Roles, Processes, and CONOPS all represent objects that are manually entered. 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 attributes of the Roles. Similar associations exist on the System side.

Principle #2 – Four Core Operational/System Architecture Entity Objects
Four object entities in each view are considered as *core* – i.e., those building block entities that make up the foundation of integrated operational and system architectures.. They are:

- Activities (*System Functions*) represent the means (transformation) by which they act on specific Information (*Data*) input (I) to produce a specific Information (*Data*) output (O)
- Operational (*System*) Node represents 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. Note that Nodes do not represent operational/ human roles - Roles represent Roles
- Role (*System*) is 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
- Information (*Data*) are formalized representations of data subject to a transformation process and are the inputs and outputs of Activities (*System Functions*)

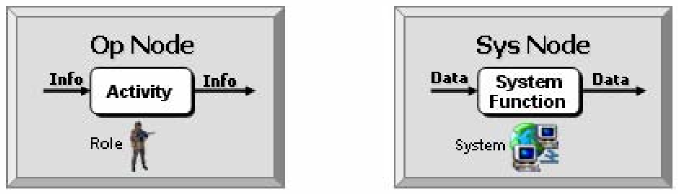


Figure 2: Core DoDAF Architecture Entities

The four core entities 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 the core architecture elements can be represented by three sets of triple relationships

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

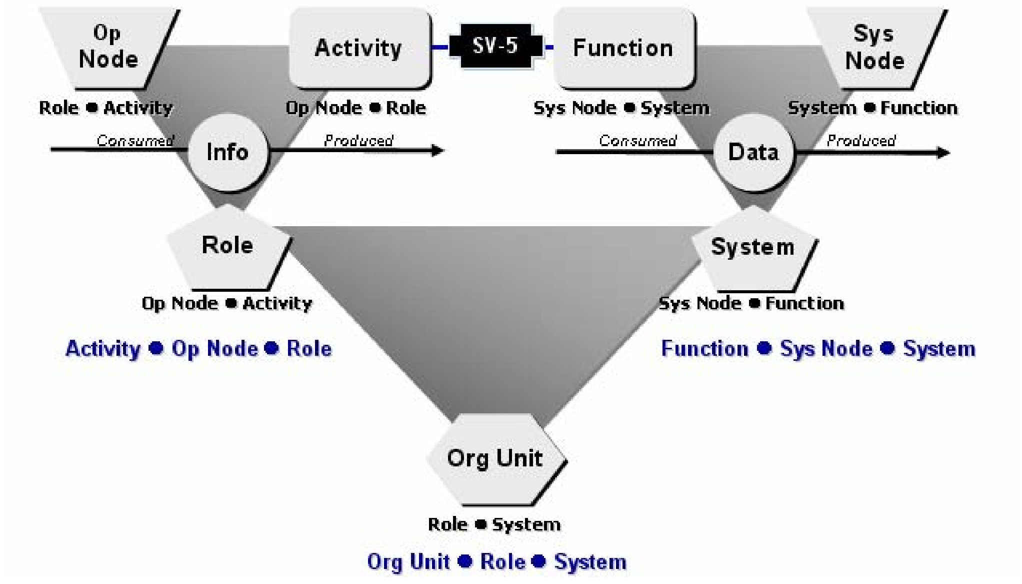


Figure 3: Relationship of the Core Architecture Artifacts

The intersection of the association of an Operational Activity with an Operation Node is a Role. Likewise, the intersection of the association of an System Function with a System Node is a System. The association of Organizational units with Roles already exists in the DoDAF OV-4 product. The Activity Based Methodology establishes an additional association of Organizational Units with Systems.

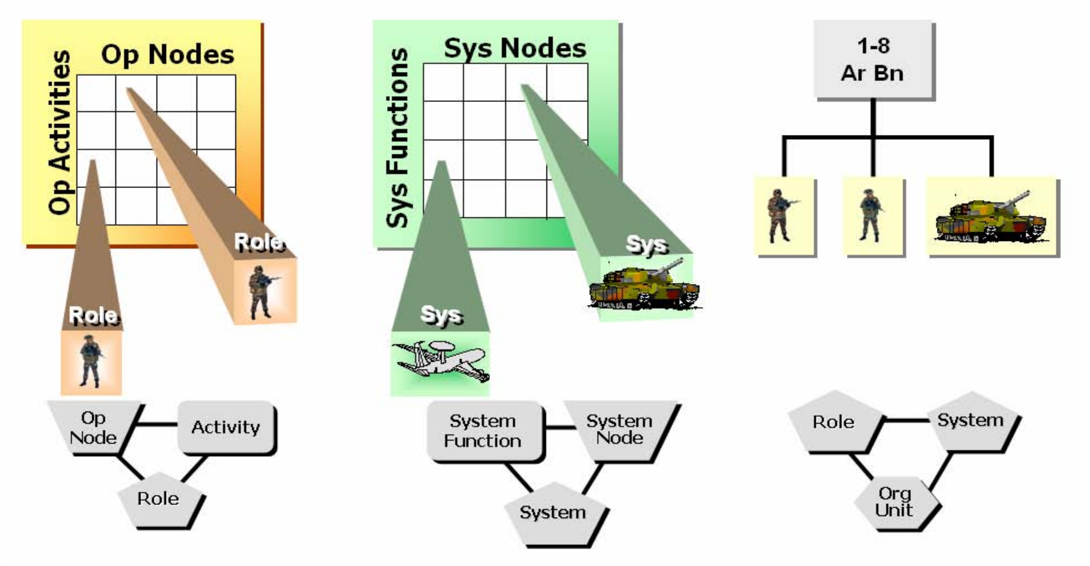


Figure 4 Triple Associations of Core Architecture Objects

DoDAF Architecture Description Specification Model (DADSM)

The alignment of DoDAF data centric architecture objects and their three sets of triple associations/ relationships are all modeled in what is called the *DoDAF Architecture Description Specification Model* (DADSM). DADSM consists of a set of formal object class specification models for each of the twenty seven DoDAF products and all their constituent objects. It serves to resolve DoDAF omissions and remove inconsistencies, ambiguities, and any architecture methodology dependencies.

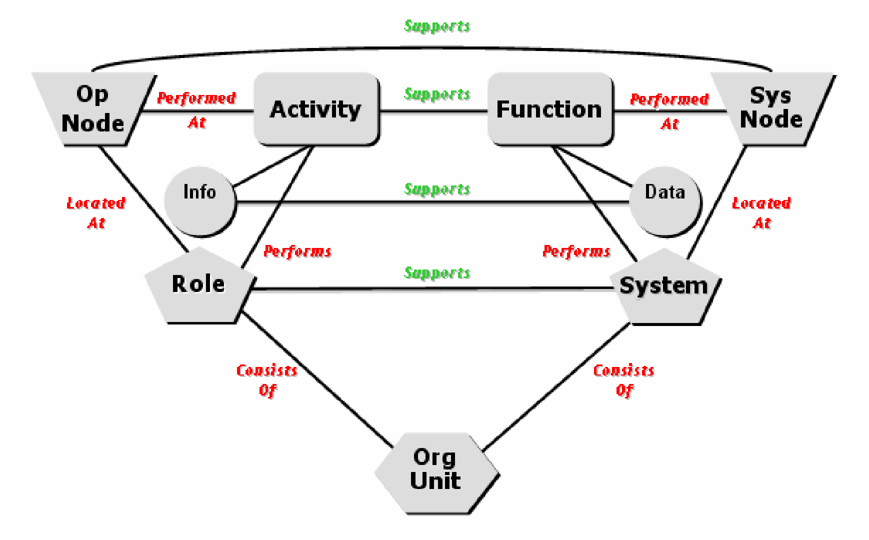


Figure 5 DOD Architecture Description Specification Model

Mapping DADSM to DOTMLPF

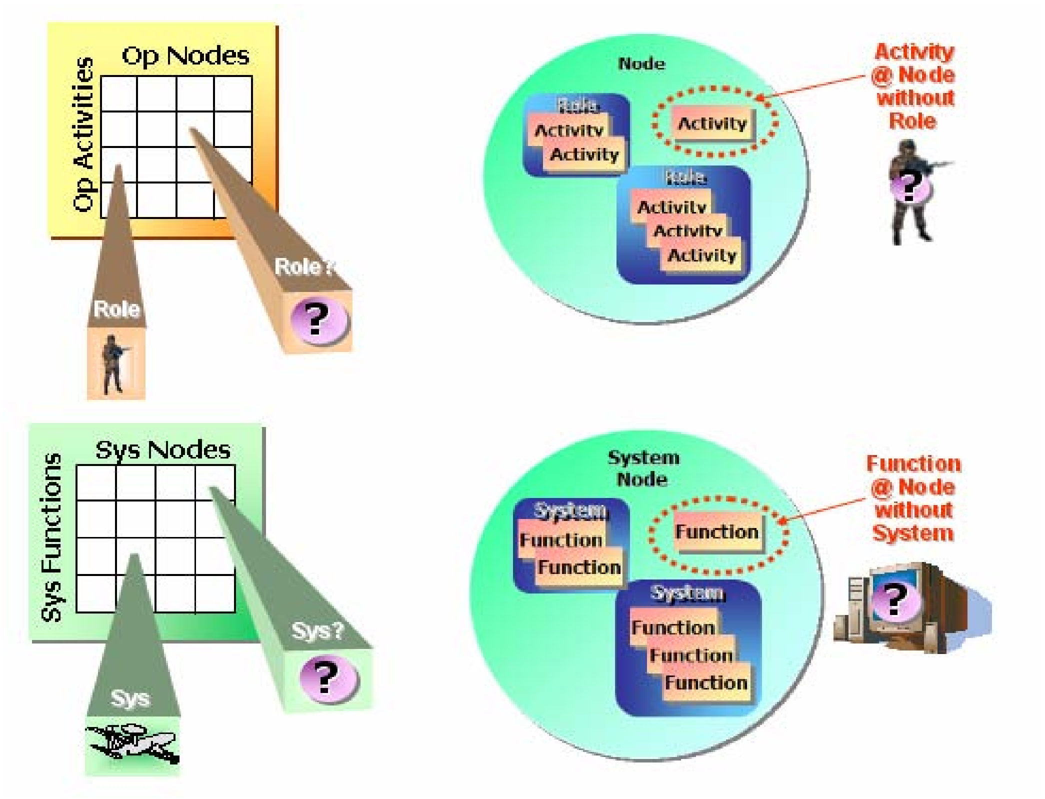
Based on DADSM, the military DOTMLPF maps to specific architecture objects

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

Gap Analysis

Note that for the operational 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 system 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 should reveal these gaps:

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



Based on this "to-be" gap analysis, by clustering and aggregating these orphaned activities and orphaned systems, one could derive a set of requirements for a needed organizational structure and a needed system or, depending on how one clusters orphaned system function, multiple systems

Principle #3 - Core architecture entity objects manually from Specific DoDAF Products
Each core architecture entity object is only manually entered from one specific Framework product. For example, activity entities are only entered when creating an OV-5 activity model. Where there are associations between, say nodes and activities, these associations are only created and managed from a two dimensional matrix (spreadsheet) editing facility.

Principle #4 – Automatically Forming Relationship/Attribute Architecture Objects
Several relationship and attribute architecture class objects can be automatically formed from the core building block entities. This is what enables "object-classes" of architecture data to be defined and insures data consistency in the resultant architecture products. Auto-generating architecture data 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 Operational side, Information Exchanges and Need Lines are formed from OV-5 activities, their information inputs and outputs, and their associations to OV-2 Operational Nodes. On the System side, System Data Exchanges and Interfaces are formed from SV-4 Functions, their data inputs and outputs, and their associations to SV-1 System Nodes.

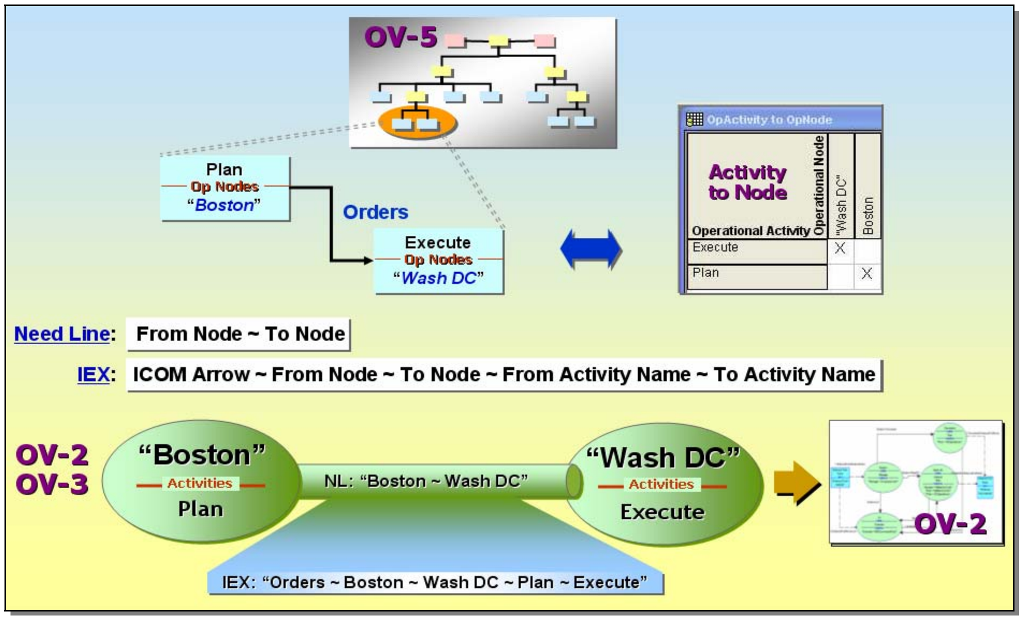


Figure 6 Forming Need Lines and Information Exchanges

Principle #5 – Automatically Generating DoDAF Products

On the Operational 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 as a document report since it consists entirely of the collection of Information Exchanges 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 as a document report since it consists entirely of the collection of System Data Exchanges.

Workflow

Operational and System Architecture development work flow consists of 8 steps – 3 manual data entry, 1 manual association, and 4 automation as follows.

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 Systems
4) Manually triple Associate Activities with Nodes with Roles	4) Manual triple Associate Sys Functions with Sys Nodes with Systems
5) Auto form 3-way associations: Activities, Nodes, and Roles	5) Auto form 3-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

Transition to Executable Architectures
The Activity-based Methodology enables the transition to dynamic executable models and the associated time-dependent behavior analysis of complex, dynamic operations and human and system resource interactions that cannot be identified or properly understood using static models. 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 what conditions information is produced/consumed. Dynamic (over time) Executable Models goes beyond “must be capable of” and define precisely under what conditions Information is actually produced/ consumed.. An Executable Architecture can then be defined as a dynamic model of Activities and their sequencing performed at an Operational Node by Roles (within Organizations) using Resources (Systems) to produce and consume Information.

The transition consists of starting with a model of the leaf activities (set of lowest activities that are not decomposed) to which dynamic processing time (duration) and its statistical time distribution, average wait time before processing, continuation strategy, activity cost, and Input/Output conditions are all filled in for each leaf activity. These become the processes in a dynamic executable process model. Information Exchanges connect the dynamic processes together in that Information Exchanges are the data produced and consumed by each process. Transport times and other time dependent properties including any statistical time distribution, quantity, and cost are also filled in. 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.

The executable models can then generate measures of performance (MOP) and measures of effectiveness (MOE) of resources to function in an operational environment. Additionally, these models provide measures of force effectiveness (MOFE) to determine the overall success of the organization employing a system in accomplishing its mission. By comparing a scenario where a system is used with the same scenario without the system, the system's contribution to overall force effectiveness can be determined. Providing time and costs analysis of executable architectures is the first step in an 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.

Summary

In summary, 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 strategy.

References

1. DOD Architecture Framework, V1.0, Vol. I and II, 15 August 2003.

Author Biographies

STEVEN RING, 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 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 analyzing and validating both static and dynamic DOD architecture models. He currently is leading an OSD and Air Force sponsored effort in developing a tool-independent approach to integrated DOD architectures for assessment and analysis that also facilitates the transition to dynamic executable architectures.

DAVE NICHOLSON, 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

JIM THILENIUS

STANLEY HARRIS