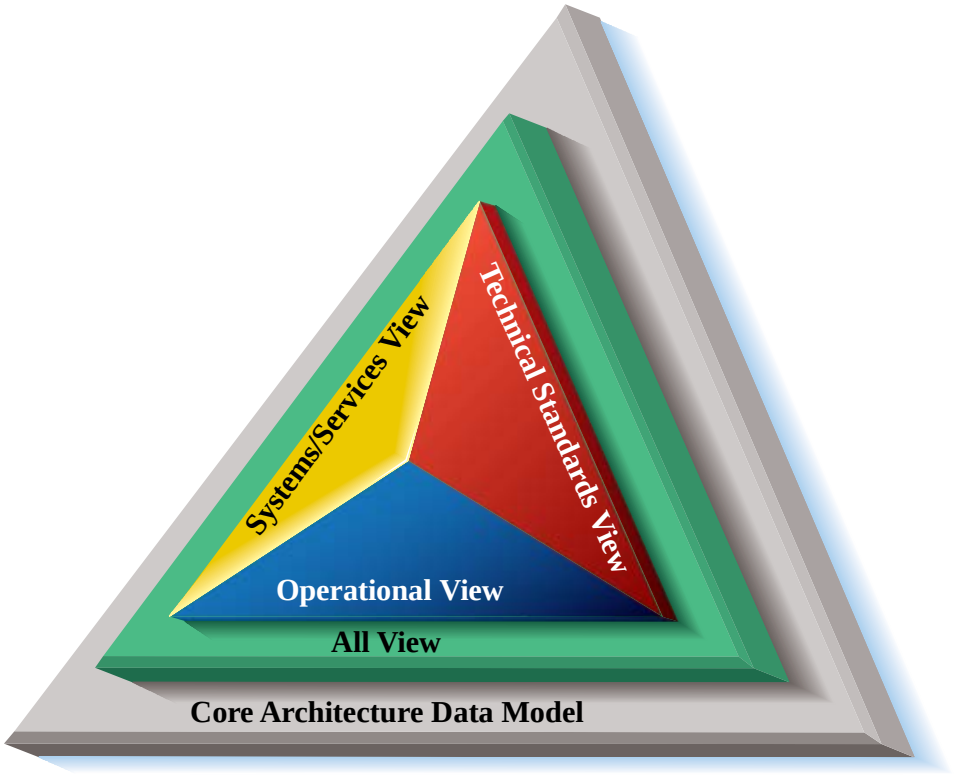
DoD Architecture Framework
Version 1.5



Volume III: Architecture Data Description

23 April 2007

TABLE OF CONTENTS

SECTION **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

LIST OF TABLES

TABLE **PAGE**

EXECUTIVE SUMMARY

> **Architecture**: the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.
>
> DoD Integrated Architecture Panel, 1995, based on IEEE STD 610.12

Architectures within the Department of Defense (DoD) are created for a number of reasons. From a compliance perspective, the DoD's development of architectures is compelled by law and policy (i.e., Clinger-Cohen Act, Office of Management and Budget (OMB) Circular A-130). From a practical perspective, experience has demonstrated that the management of large organizations employing sophisticated systems and technologies in pursuit of joint missions demands a structured, repeatable method for evaluating investments and investment alternatives, implementing organizational change, creating new systems, and deploying new technologies. Towards this end, the DoD Architecture Framework (DoDAF) was established as a guide for the development of architectures.

The DoDAF provides the guidance and rules for developing, representing, and understanding architectures based on a common denominator across DoD, Joint, and multinational boundaries. It provides insight for external stakeholders into how the DoD develops architectures. The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and ultimately, the enterprise, thus, establishing the foundation for analyses that supports decision-making processes throughout the DoD.

As the Department takes appropriate strides to ensure advancement of the Information Technology (IT) environment, it becomes essential for the DoDAF to transform to sufficiently support new technologies. A significant evolution occurring today is the Department's transformation to a new type of information intensive warfare known as Net-Centric Warfare (NCW). NCW focuses on generating combat power from the effective linking or networking of the warfighting enterprise, and making essential information available to authenticated, authorized users when and where they need it. This ability is at the heart of net-Centricity and essential to achieving Net-Centric Operations (NCO).

DoDAF v1.5 is a transitional version that responds to the DoD's migration towards NCW. It applies essential net-centric concepts[1] in transforming the DoDAF and acknowledges that the advances in enabling technologies – such as services within a Service Oriented Architecture (SOA) – are fundamental to realizing the Department's Net-Centric Vision[2]. Version 1.5 addresses the immediate net-centric architecture development needs of the Department while maintaining backward compatibility with DoDAF v1.0.

In addition to net-centric guidance, DoDAF v1.5 places more emphasis on architecture data, rather than the products, introduces the concept of federated architectures, and incorporates the Core Architecture Data Model (CADM) as an integral component of the DoDAF. These aspects

---

[1] Reference DoDAF v1.5 Volume II for further information on the following net-centric concepts and their application to DoDAF: 1) Populate the Net-Centric Environment , 2) Utilize the Net-Centric Environment , 3) Accommodate the Unanticipated User, 4) Promote the Use of Communities Of Interest (COI), 5) Support Shared Infrastructure

[2] 2005 National Defense Strategy

prepare the way for more efficient and flexible use and reuse of architecture data, enabling broader utility for decision makers and process[3] owners.

The DoDAF is a three-volume set that inclusively covers the concept of the architecture framework, development of architecture descriptions, and management of architecture data.

- Volume I introduces the DoDAF framework and addresses the development, use, governance, and maintenance of architecture data.
- Volume II outlines the essential aspects of architecture development and applies the net-centric concepts to the DoDAF products.
- Volume III introduces the architecture data management strategy and describes the pre-release CADM v1.5, which includes the data elements and business rules for the relationships that enable consistent data representation across architectures.

An Online Journal, hosted on the DoD Architecture Registry System (DARS) website (https://dars1.army.mil/IER/index.jsp), replaces the DoDAF v1.0 Desk Book and is designed to capture development best practices, architecture analytical techniques, and showcase exemplar architectures.

The DoDAF will continue to evolve to meet the growing needs of decision makers in a Net-Centric Environment (NCE). Going forward, architectures will need to capture the development of a new generation of net-centric capabilities stemming from operational insights gained in Afghanistan and Iraq. As the maturation of the Global Information Grid (GIG) continues through GIG Capability Increments (an incremental timeframe approach to the delivery of GIG-enabling capabilities), architectures will be a factor in evaluating increment investments, development, and performance at the mission portfolio levels. As the DoD increases its use of architecture data for decision making processes, architects will need to understand how to aggregate the data for presentation purposes at the enterprise level. The DoDAF plays a critical role in the development of architectures and will continue to improve its support for the increasing uses of architecture data.

[3] Chairman Joint Chiefs of Staff Instruction (CJCSI) 3170.01E, Joint Capabilities Integration and Development System (JCIDS); DoD Directive 7045.14, Planning, Programming, Budgeting, and Execution (PPBE); DoD Directive 5000.1, The Defense Acquisition System (DAS); DoD Directive 8115.01, Information Technology Portfolio Management (PfM)

1 INTRODUCTION

1.1 VOLUME III PURPOSE AND INTENDED AUDIENCE

The purpose of this volume is to define, describe, and provide a use for Architecture Data. This volume is organized with various readers in mind.

a. For the **manager** who needs to lead architecture development projects and who may need to use architecture data and products to make acquisition, budgeting, or resourcing decisions, **product definition and purpose** subsections are provided in each product section to:

    1) Help these managers to understand the architecture components or products.
    2) Provide an appreciation of the potential level of effort involved in developing such architectures.
    3) Assist them to discern the potential needs or uses of such an architecture effort.

b. For the **architect and engineering team** who need to present architecture products to the high-level decision makers for use in decision support analysis, **a detailed description of the "data layer" as defined in CADM,** and a **data element table** subsection are provided in each product section to:

    1) Enable the architect and engineering team to identify products to be included in the architecture based on the architecture's intended use (see Use Matrix).
    2) Determine architecture data needs.
    3) Identify sources for the architecture data.
    4) Analyze and relate the data gathered.
    5) Compose the data into architecture products.

For the **architecture data modelers, tool developers, and engineers** who are involved with implementing a data repository to store and manipulate Framework data elements, a **CADM support** subsection is provided in each product section.

This document is organized in the following manner:

Section	Content
Section 1	*Introduction* – Provides an overview of the DoDAF and the benefits of architecture data.
Section 2	*Architecture Data Management Strategy* – Describes the overall DoD architecture data management strategy, in particular how it fits and supports the Department's Net-Centric Data Strategy (NCDS).
Section 3	*Architecture Metadata and Federated Architecture Registry Approach* – Provides a description of architecture metadata, in particular, discovery metadata and how data can be discovered and accessed using a federated repository system coupled with a centralized registry system, the DARS.

Section 4	*Core Architecture Data Model v1.5* – Describes architecture product "data layer" as defined and supported in CADM v1.5.

Table 1-1: Organization of Volume III

1.2 OVERVIEW

The DoDAF, v1.5 provides the guiding principles for modeling and designing architectures via a set of products that support the DoD environment. The Framework is intended to ensure that architecture descriptions can be compared and related across organizational boundaries to support multiple stakeholder perspectives.

An architecture description is a representation of a defined domain, as of a current or future point in time, in terms of its component parts, how those parts function, the rules and constraints under which those parts function, and how those parts relate to each other and to the environment. Within the DoDAF, architectures are described in terms of four views: All View (AV), Operational View (OV), Systems and Services View (SV), and Technical Standards View (TV). An architecture description is composed of architecture products that are interrelated within each view and are interrelated across views. Architecture products are those graphical, textual, and tabular items that are developed in the course of gathering architecture data, identifying their composition into related architecture components or composites, and modeling the relationships among those composites to describe characteristics pertinent to the architecture's purpose.

Underlying both the Framework and supporting architecture tools is the goal of a common specification of the data planned to be incorporated in architecture data repositories and databases. Such a specification needs to be tool independent, constrained only to the degree required for exchanging and reusing data underlying architectures developed for the Department. The CADM provides such a specification.

Architectures are typically developed as a set of products based on an underlying data structure. Merging the underlying data of these products and the architecture in general into a database or other kind of data repository enables architecture data to be maintained in a consistent way and to be reused by other versions of the architecture and by other architects. The benefits of additionally developing, validating, and maintaining architectures in an agreed-upon data repository structure include the following:

a. Consistently expressing the commonality of data underlying architecture products and integrated architectures
b. Enabling the potential for reuse of data underlying all DoD architectures
c. Ensuring consistent data across multiple architectures and architecture products
d. Enabling flexible re-partitioning for different points of view
e. Supporting taxonomies of key reference data
f. Exchanging data among architecture data repositories and multiple modeling and analysis tools
g. Supporting data maintainability by use of standard import mechanisms from authoritative data sources

h. Providing a basis for enterprise-level decision support systems in which architecture data can be queried and analyzed and reports generated for a variety of decision support analyses.

Benefits of using the CADM are best realized if architects go beyond the barriers of proprietary tool formats by using tool templates, data tagging, and export formats aligned with the CADM. The current breed of architecutre tools are generally methodology dependent, which often results in archiecture data that are critical for analysis using those methodologies, but is not readily aligned with the current DoDAF view set or CADM specification. The result is that some tools and methodologies will be challenged in meeting the objectives of DoDAF and CADM conformance. This issue is being addressed through DoDAF data modeling work groups. The primary goal the DoDAF data modeling work groups is to develop modeling guidelines and profiles for popular methodologies along with view specifications and data structures for the next version of the DoDAF and CADM that minimize data element alignment issues and better supports architecture data reuse across different modeling tools and methodologies.

1.3 BENEFITS OF A DATA-CENTRIC APPROACH FOR ARCHITECTURE DEVELOPMENT

Architecture data is typically collected and represented via graphs (i.e., nodes connected by edges) and matrices. The benefits of additionally producing, exporting, and validating CADM conformant data are:

a. Consistency. Developing and expressing products using CADM conformant data ensures consistency through the use of common data elements and taxonomies. Two types of product consistency can be achieved:

   1) Across Levels of Abstraction within the same product. Architectural descriptions at a detailed level of abstraction ensure consistent assertions at higher levels of abstraction through the taxonomic structure of the elements.

   2) Across Products. Many architecture products can be specified using some of the same set of data elements. Product data developed, maintained, and generated in conformance with the CADM ensure consistency in data elements common to multiple products.

b. Data re-use and flexible partitioning. Data can be re-used by different teams, perhaps looking at the architecture from different mission area, functional area, capability, or task force points of view. The data can be ‘sliced and diced’ in a CADM conformant repository however needed, but all the data comes from the same CADM conformant dataset – ‘develop once, use many.’ This provides efficiency, flexibility, and reduces the need for complex, costly, and sometimes infeasible reconciliations.

c. Inter-agency architecture data interoperability. Interfaces to other architecture data repositories can be used to assess inter-organizational interoperability, gaps, or redundancy issues. Inter-organizational interoperability is one of the major reasons for employing architectural techniques.

d. Ability to use multiple tools and perform adhoc analyses. Commercial off-the-shelf software (COTS), Government off-the-shelf software (GOTS), and adhoc reports, diagramming, executable modeling, and other modeling and simulation (M&S) tools can

be interfaced to the data repository, so architecture developers and users are not restricted to the functionality of one tool.

e. Interfaces to other enterprise authoritative data sources. The authoritative data source for much architecture data is actually non-architecture data sources, e.g., the Universal Joint Task List (UJTL), DoD IT Standards Registry (DISR), IT Systems Registry list, organizations, occupational specialties, ships, aircraft, facilities, units, costing, and budget data. Ideally, these would be interfaced to the architecture repository rather than manually input, parsed, or imported by each architecture developer.

f. Maintainability. 'Develop once, use many' and interfaces to authoritative data sources promote maintainability and validity of CADM conformant data.

g. Rapid Decision Support. The integrated architecture data repository becomes an enterprise Decision Support System (DSS). The data can be queried and analyzed, and reports can be generated for decision support from a CADM conformant repository. Some data may be required to augment the decision aid, but, to the extent architectural data are involved, pull from the data repository supports faster decision support and reduces redundant data calls.

h. Integration with Enterprise Taxonomies. Enterprise taxonomies are important components of enterprise knowledge management, providing a basis for the enterprise ontologies. Employing consistent taxonomies in the architecture data repository links knowledge management ontologies with Enterprise Architecture (EA).

Using Architecture Repository Data to Assess a Proposed Architecture. The use of architecture data in conjunction with M&S, performance analysis, and assessment tools is an area of expanding interest because of its importance for capabilities-based assessments and analysis of alternatives.

The potential value to an enterprise of a proposed architecture may not be obvious. Measures of merit can include cost; performance; interoperability; satisfaction of requirements; manpower and training; logistics, deployment, and asset allocation; schedule, and many others. The formulae for computing measures of merit may be quite complicated, as in a complex M&S program. An important ingredient in these measures is quality input data.

In addition to supporting the data requirements of the DoDAF, the CADM was originally developed to support the needs of the M&S community for architecture and interoperability analyses. For example, NETWARS is a GOTS/COTS tool that estimates communications throughput requirements from Information Exchange Requirements (IERs). NETWARS uses IER attributes for information element size, frequency, timeliness, security, required format, etc., along with operational node to physical node mappings to estimate bandwidth requirements at physical nodes and predict throughput bottlenecks.

IER attributes, at the operational, functional, and system levels, across time periods or "as-is" and "to-be," are not the only architecture data elements that can be used to compute measures. Task and process-activity staffing levels, technical standards such as communications protocols, network architectures, scenario information, and performance data, can all be input to M&S and analysis tools for performance measures computation. The advantage of using CADM structures for developing and maintaining measures of merit data is that M&S, analysis, and assessment tools developed or modified to compute the measures based on architecture repository data are

standardized. This means multiple M&S, analysis, and assessment tools can use the same data sets (data reuse) and that, over time, these tools can evolve to provide a fuller set of measures needed for decision support.

2 ARCHITECTURE DATA MANAGEMENT STRATEGY

The DoD NCDS lays out a new approach for data management that focuses on making data visible, available, understandable and trusted in a Net-Centric Operating Environment (NCOE). The strategy applies to all *data assets* on the GIG, including architecture data. Data assets are defined to include system or application output files, databases, documents, or web pages. For the architecture community, data assets include integrated architectures and individual architecture products produced and stored in architecture tools and data repositories. Implementation of the NCDS throughout the DoD architecture community will enable architecture producers and end users to discover, share, understand, and use architecture data and products created and stored in independent architecting environments across the Department.

This section presents the recommended approach to implementing NCDS for management of DoDAF architecture data for DoD Commands/Services/Agencies (C/S/A). Key aspects of the strategy are to 1) make data visible, available and usable, 2) "tag" data with metadata to enable discovery (see section 2.3), 3) post data to shared spaces, and 4) move away from point-to-point interfaces to "many-to-many" exchanges within a net-centric data environment. **Figure 2-1** shows the scope of the NCDS.

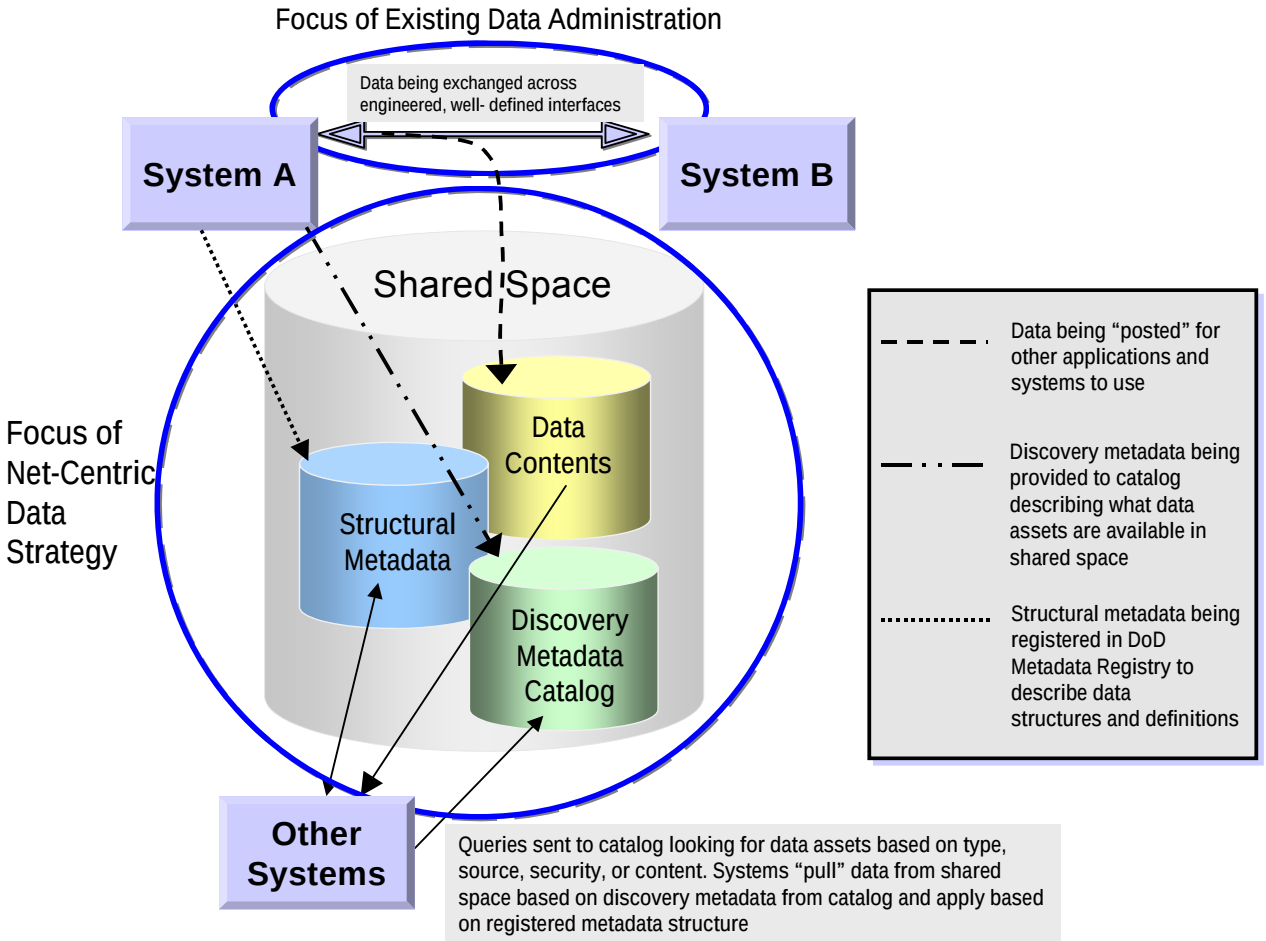


Figure 2-1: Scope of the NCDS

In the context of the DoDAF architecting community, the Systems A and B may represent any architecting tool environment or any system requiring access to architecture data, including

unanticipated users. Architecture data assets are posted in a virtual shared space (*Data Contents*) and accessible to anyone with appropriate access permissions, in formats that are commonly useable. Data are structured in accordance with structural metadata registered in the DoD Metadata Registry (DMR) (*Structural Metadata*) – providing understanding and usability through definitions of the data elements and content structure. Content metadata is registered in catalogs (*Discovery Metadata Catalog*) accessible using enterprise discovery search services. Users search for architecture content by executing a discovery search in the Discovery Metadata Catalog (also known as the Enterprise Catalog) and pull content of interest from repositories forming the content virtual shared space (*Data Contents*).

2.1 DoD Net-centric Policy and Directives

The DoD Chief Information Officer (CIO) has outlined a vision for managing data in the NCE in the NCDS memorandum [NCDS 2003]. In 2004, DoD issued Department of Defense Directive (DoDD) 8320.2, "Data Sharing in a net-Centric DoD," that established policies and responsibilities for implementing the NCDS throughout the DoD.

The NCDS data management vision is based on three key elements: 1) collaborative groups of users, called Communities of Interest (COIs), who must exchange information in pursuit of their shared goals, interests, missions, or business processes, 2) metadata standards (for describing information about data), and 3) GIG Enterprise Services that enable discovery and sharing of data.

NCDS identifies seven specific data goals (**Table 2-1**) by which the strategy will be achieved – be visible, be accessible, be institutionalized (incorporated into DoD processes and practices), be understandable, be trusted, be interoperable, and be responsive to user needs.

Table 2-1 DoD net-nentric Data Goals

Goal	Description
Goals to increase Enterprise and community data over private user and system data	
Visible	Users and applications can discover the existence of data assets through catalogs, registries, and other search services. All data assets (intelligence, non-intelligence, raw, and processed) are advertised or "made visible" by providing metadata, which describes the asset.
Accessible	Users and applications post data to a "shared space." Posting data implies that (1) descriptive information about the asset (metadata) has been provided to a catalog that is visible to the Enterprise and (2) the data are stored such that users and applications in the Enterprise can access it. Data Assets are made available to any user or application except when limited by policy, regulation, or security.
Institutionalize	Data approaches are incorporated into Department processes and practices. The benefits of Enterprise and community data are recognized throughout the Department.
Goals to increase use of Enterprise and community data	
Understandable	Users and applications can comprehend the data, both structurally and semantically, and readily determine how the data may be used for their specific needs.
Trusted	Users and applications can determine and assess the authority of the source, because the pedigree, security level, and access control level of each data asset is known and available.
Interoperable	Many-to-many exchanges of data occur between systems, through interfaces that are sometimes predefined or sometimes anticipated. Metadata are available to allow mediation or translation of data between interfaces, as needed.
Responsive to User Needs	Perspectives of users, whether data consumers or data producers, are incorporated into data approaches via continual feedback to ensure satisfaction.

2.2 COMMUNITY OF INTEREST

The NCDS uses the term COI to describe collaborative groups of users who must exchange information in pursuit of their shared goals, interests, missions, or business processes and who therefore must have shared vocabulary for the information they exchange. NCDS relies on COIs to achieve net-centric data goals by establishing COI agreements on common semantics and structural metadata, cataloging data and metadata, and having members post data assets and metadata to a virtual "shared" space.

Each DoD C/S/A can be considered a COI for the purpose of establishing common *reference data* (i.e., common sets of terms) for the values of data elements to be used for architecture description within a C/S/A domain. However, they all share a common interest in developing DoDAF architectures and, thus, must be considered part of a larger and inclusive DoD Architecture COI for the purpose of establishing agreements on common semantics and structure of DoDAF architecture data elements. Office of the Assistant Secretary of Defense/Networks and Information Integration (OASD/(NII), Architectures and Interoperability Directorate

(A&ID) has the responsibility to govern this larger and inclusive DoD Architecture COI so that the NCDS vision can be realized.

The DoD architecture community has been actively engaged in COI activities for over 10 years. For example:

- The Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework, and later, the DoDAF, was developed to provide a common understanding of the core data content for various views of architecture.
- The CADM was developed by the DoD architecting community to provide an agreement on common structural metadata and semantics for DoDAF architecture data elements.
- DARS was initially established to provide a community shared space for posting and searching architecture data and now also serves as the community's content metadata registry – integrated with the Net-Centric Enterprise Services (NCES) Enterprise Catalog via a federated discovery service.
- The EA Summit provides a forum for senior C/S/A CIOs to discuss issues and coordinate activities for the common benefit of the community.
- The Federated Joint Architecture Working Group (FJAWG) was established to develop solutions, guidance, and implementation plans for federating and integrating architectures.
- The DoD Architecture Configuration Control Board (ACCB) was established to maintain configuration management authority over the DoDAF, CADM, DARS, and reference taxonomies.

All of these ongoing activities, along with regular community-wide conferences, meetings, and working groups conducted under the sponsorship of OASD(NII) A&ID, are essential for the architecture community to carry out the intended role of COIs in achieving the NCDS vision.

2.3 METADATA

The NCDS calls for COIs to establish agreements on semantic and structural *metadata* needed to provide an understanding of the content of COI data assets. *Metadata* is descriptive information about the meaning of other data or data processing services (e.g., web services). For architecture data assets, metadata are used to provide information about the content of an integrated architecture, individual architecture products, or about the use of data processing and analysis web services. Extensible Markup Language (XML) provides a means of identifying data asset metadata elements in the form of a structured XML document using name tags for each element.

NCDS requires COIs to tag all COI data assets with discovery metadata conforming to the DoD Discovery Metadata Specification (DDMS) to document information about the content of the data assets. NCDS also calls for discovery metadata to either be posted to searchable community metadata catalogs (DARS for DoDAF architectures) or produced in the form of a DDMS-conformant XML document in response to a discovery search request from the NCES discovery service. Archiecture metadata can be manually posted to DARS to provide visibility via NCES or DARS discovery services. Alternarively, the implication of the later option is that

GOTS and COTS tools must implement either the NCES or DARS federated search web service to provide visibility by implementing automated discovery services. The ability to discover all DoDAF architecture content throughout DoD via a federated search is a core element of the Department's Architecture Data Management Strategy.

NCDS discovery metadata standards are provided in the DDMS. DDMS defines metadata elements to be associated with data assets posted to shared spaces so that they can be discovered via DDMS-conformant search services. DDMS combines a Core Layer and an Extensible Layer. The Core Layer consists of four sets of element categories, each targeted to a specific functional area. The Extensible Layer supports unique COI metadata requirements by extending the Core Layer element categories and is used by the DoD architecture COI to record discovery metadata elements unique to the DoD architecture community.

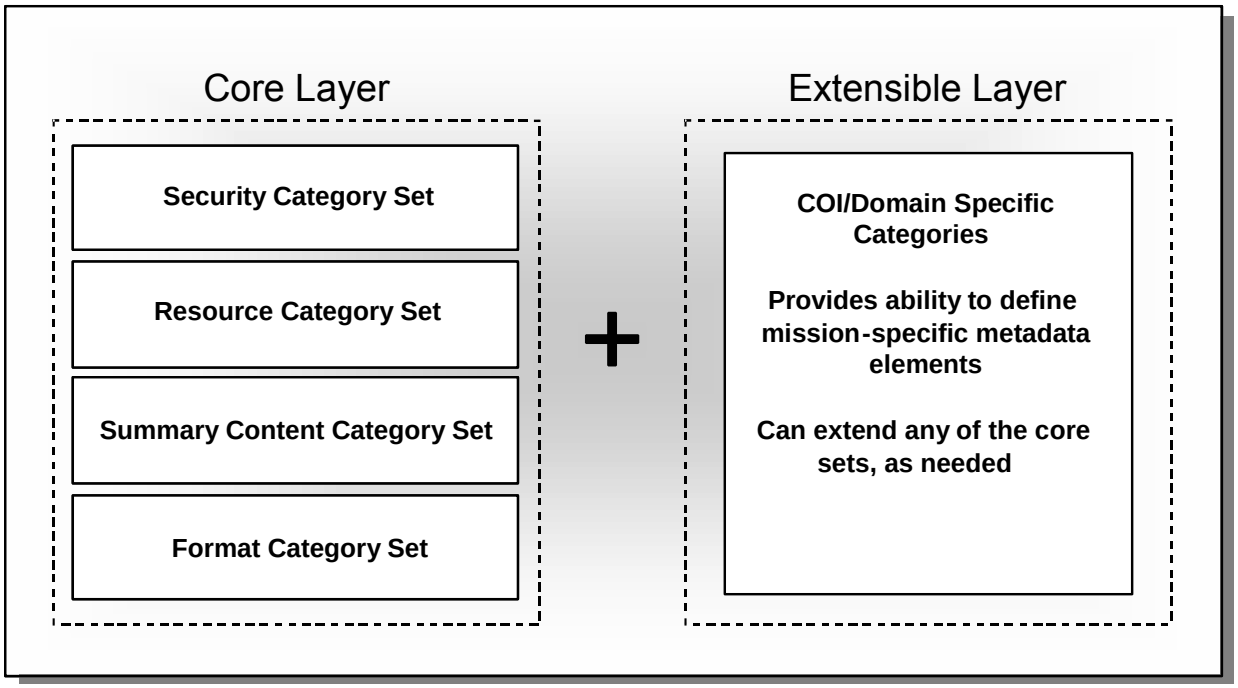


Figure 2-2: DDMS Logical Model

A set of DDMS extension metadata elements is defined in Section 3 to support the unique requirements of the DoDAF architecture community. The initial extension set was developed by FJAWG and is being configuration managed by the DoD ACCB. The discovery metadata extension set is based on the descriptive content elements of the AV-1 and serves several purposes. It enables searching and filtering architecture content based on the values of AV-1 metadata elements. The metadata elements also enable EA federation by linking architectures to elements of the DoD Business Reference Model (BRM) as a means of classifying architectures and building a DoD EA navigation map in DARS. Architecture federation and linking are further described in Section 3.

2.4 NET-CENTRIC OPERATING ENVIRONMENT (NCOE)

The NCOE is the environment that enables the NCDS and Architecture Data Management Strategy. It consists of the set of NCES Core Enterprise Services (CES), GIG infrastructure, and all other services available to users on the GIG. NCOE resources directly supporting the

Architecture Data Management Strategy include the DMR, the Enterprise Service Registry, the Enterprise Catalog, and the NCES Federated Search and Security CES. **Figure 2-3** depicts how NCOE enterprise resources support the Architecture Data Management Strategy.

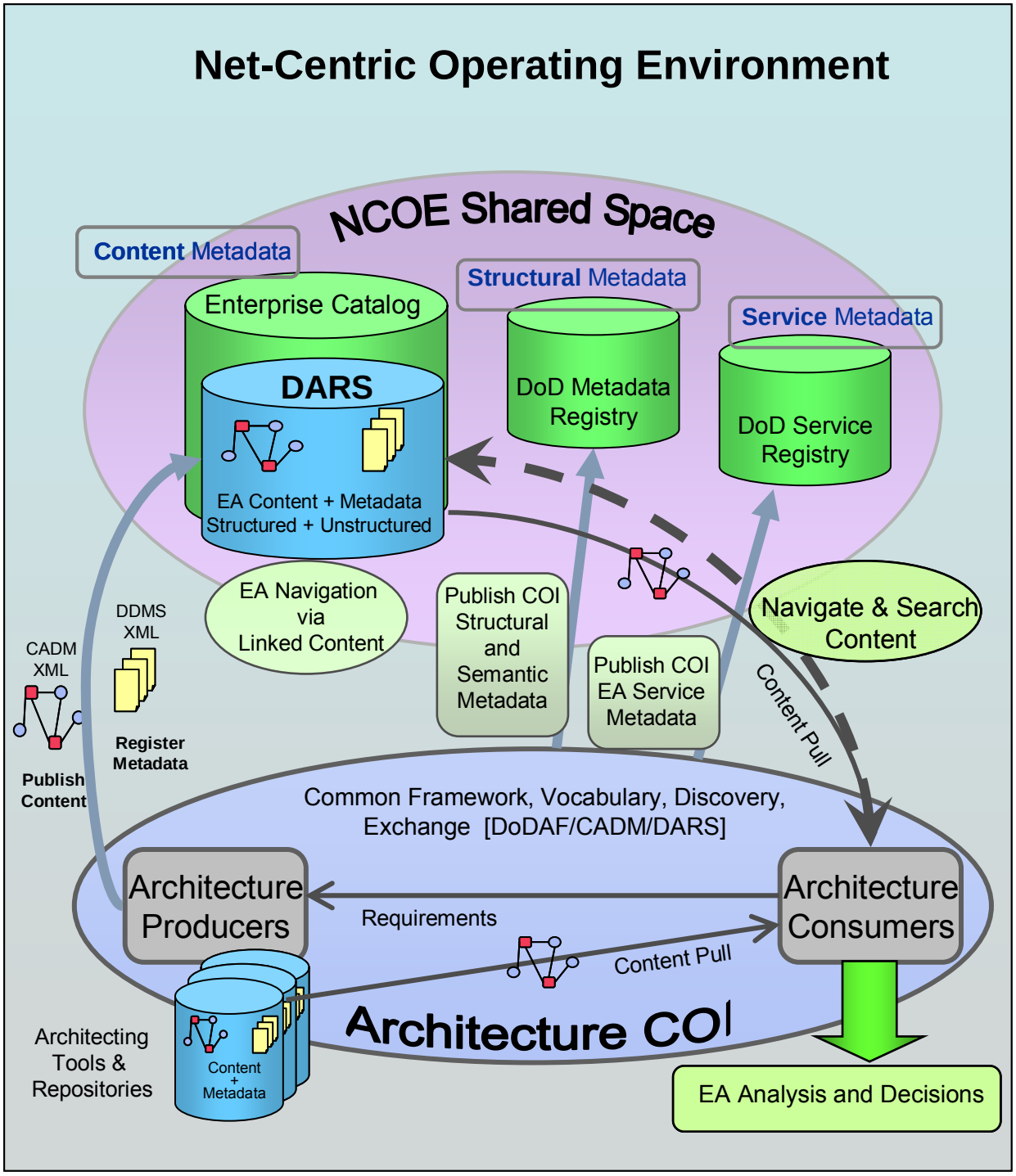


Figure 2-3: NCOE Enterprise Resources

Enterprise web services include the CES and other web services made available for general use by DoD by publishing the services' metadata to the Enterprise Service Registry, where it can be discovered and executed, and where it is subject to security and policy constraints. A Web

Service is defined by the World Wide Web Consortium (W3C) as a software system designed to support interoperable machine-to-machine interaction over a network. Web services are modular application components built in conformance with a set of XML standards that enable applications to access the services of other applications via a standard set of communications protocols – independent of the application implementation environment. Enterprise web services include the CES and other web services made available by publishing the services' metadata to the Enterprise Service Registry, where it can be discovered.

The CES provide the foundational web services for achieving the goals of the NCDS. The architecture community will provide EA web services of interest to architects and architecture end-users across DoD. EA services include services for architecture data management, quality assessment, and analysis developed by OASD(NII) and C/S/A components of the architecture community. EA services designed and implemented to achieve the goals of the architecture data management strategy will be integrated with the CES.

2.4.1 DoD Metadata Registry

The DMR provides access to structural and semantic metadata describing architecture. OASD(NII) Architectures and Interoperability (A&I) has registered the CADM and CADM XML schema as the semantic and structural specifications for DoDAF architecture data elements. The DoD architecture community has also registered a set of DDMS extension metadata elements that are available for reference. Potential users of architecture data can access CADM and DDMS specifications in the DMR to understand the structure and meaning of architecture products and metadata made available throughout the community in the form of CADM XML or DDMS XML documents. The DMR can be accessed at:

https://metadata.dod.mil/mdrPortal/appmanager/mdr/mdr

2.4.2 Enterprise Service Registry

The Enterprise Service Registry is the component of the NCOE, where web service metadata is registered to enable service discovery and use. EA services developed by OASD(NII) and C/S/A components will be registered in the Enterprise Service Registry, as they are developed by the community.

2.4.3 Enterprise Catalog

The Enterprise Catalog is a virtual catalog of all DoD data assets. It supports the NCDS by enabling searches on DDMS and COI-extension metadata registered in federated metadata registries. It functions as a federated catalog of DoD data assets via a federated search web service – providing the ability to search content metadata throughout the federation.

DARS metadata registration and federated discovery services provide the architecture community's extension to the Enterprise Catalog. It enables searching for architecture content based on DDMS metadata elements. A detailed description of architecture metadata and the DARS federated search is provided in Section 3.

2.5 VISIBILITY, ACCESSIBILITY, UNDERSTANDABILITY, AND TRUST

2.5.1 Visibility

Architecture data producers should make their data visible to all potential consumers either by posting architecture products and content metadata to the DARS shared space or by implementing the DARS federation web services in local data repositories. The DARS federated

search service provides visibility by enabling discovery of architecture content in all architecture repositories implementing the federated search service. Acquisition program managers (PMs) and C/S/A Chief Architects should ensure that procedures are implemented to make all architecture content visible as soon as development begins.

For DoDAF architectures, within security and policy constraints of the component, visibility should be provided as soon as the scope of the architecture and all mandatory DDMS data elements can be specified, regardless of the state of development, maturity, or approval. DDMS extension metadata elements for the architecture community should be used to identify the completion and approval status, so that potential users can assess suitability for use. Mechanisms for enabling visibility, and maintaining currency and relevance of discovery metadata, are detailed in Section 3. Acquisition PMs and C/S/A Chief Architects should ensure that procedures are implemented to make all architecture data visible.

2.5.2 Accessibility

Accessibility is provided when users and applications can both: 1) discover architecture content by browsing or searching publicly accessible metadata registries conforming to the DDMS, and 2) access data in commonly understood formats using standard protocols – subject to policy and security restrictions. For the DoDAF architecture community, architecture products can be made accessible by registering content metadata in DARS or a DARS federated content provider and providing users and applications a role-restricted capability to extract data and product files in common tool formats and as CADM XML. Role restriction implies that information assurance procedures are followed for providing role-based access based on approved security policies. Acquisition PMs and C/S/A Chief Architects should ensure that procedures are implemented to make all architecture data accessible.

2.5.3 Understandability

Data can be understood when a consumer can use the data, as discovered and made available from the shared space, for the intended purposes. Understandability implies that users can interpret the structure and semantics of the content consistent with COI-established structural and semantic agreements. A common understanding requires that both producer and consumer agree on the meaning of their shared data. For the DoDAF architecture community the structure and semantics for architecture data are provided by the CADM. Acquisition PMs and C/S/A Chief Architects should ensure that procedures are implemented to make all architecture content available in conformance with the structure and semantics of the CADM to enable anticipated and unanticipated users to understand and use architecture data.

2.5.4 Trust

The trust aspect of the NCDS is concerned with ensuring that the source and quality of data, and data access controls can be identified and trusted. The three key aspects of the Architecture Data Management Strategy for ensuring trust: 1) controlled access to architecture data assets based on user roles, 2) practices for creating and managing high quality architecture data, and 3) metadata that identifies the source, quality, and access control of data.

Architecting environments should provide capabilities for setting up user groups and allowing access to, as well as release of, "approved" documents at the discretion of the group sponsor or approval authority, thereby imparting a measure of trustworthiness in “approved” documents. Group administrators should be able to manage multiple product versions, thereby enabling users to identify and track version changes. At the data level, all record changes should

be logged to identify the user, time, and type of change. A data auditing feature should enable rolling back any changes made. Data should be tagged to control access based on user roles and need to know.

Acquisition PMs and C/S/A Chief Architects should ensure that procedures are implemented to 1) control access to architecture data based on user roles, 2) create and verify the use of authoritative taxonomies and reference data in architectures, and 3) use the DDMS and architecture community extension metadata elements to document the source, quality, and access control aspects of architecture data.

2.6 DATA QUALITY

Data quality affects the ability to analyze architecture models and the ability to compare or integrate independently developed architectures. Architecture data quality can be characterized and measured at various levels of granularity. At the data element level there are two key aspects of data quality. One is conformance with established structural and semantic specifications (i.e., the definitions of fundamental data entity types or object classes and their attribute data type specifications). The CADM provides such a specification for architecture data. Another aspect is conformance with preferred or mandated entity or object instance values (referred to here as reference data) established by recognized authorities, or *authoritative sources*. An *authoritative source* is a designated or recognized authority for specifying the acceptable or allowable data instance values (e.g., domain values) and their taxonomies. A *reference data set* refers to a set of element values that are approved or designated for use by a recognized authoritative source.

An example of an authoritative source for reference data is the Joint Staff for the names and definitions of Joint Capability Areas (JCA). Since it is the designated authority for defining JCAs, the Joint Staff serves as the authoritative source for "JCA_Name" and "JCA_Description" and for the taxonomic composition of JCAs. Other sources of definitions for JCAs would not be considered authoritative and should not be used in developing architecture descriptions within the DoD. DoD architects should use reference data from recognized authoritative sources wherever possible. The use of authoritative reference data in architectures eliminates ambiguity, provides consistency, and facilitates analysis and integration.

When architecture data elements are combined to form an architecture description, another aspect of data quality becomes important – that is, the degree to which an architecture model accurately represents an existing "as-is" architecture, or the proper association of components in a notional "to-be" architecture. This aspect of data quality is dependent on the knowledge of the architecture team about the capability domain being modeled and the reliability of the architects in accurately representing facts about the domain. This aspect of architecture data quality is difficult to measure, but can be controlled through subject matter expert (SME) review and architect training.

Data quality is ultimately dependent on, and should be assessed based on, the intended use. Intended use may vary from communicating general information about a mission scenario to providing a system engineering requirements baseline. Since a single quality metric or set of metrics is not practical for all intended uses, objective measures of quality should be specified based on intended use by C/S/A components. For example, C/S/A Chief Architects may establish requirements for architecture data quality based on any or all of the following criteria:

- Use of specific authoritative taxonomies and reference data

- Use of specific product description (i.e., view) templates with mandatory elements
- Use of specific description methodologies

These criteria and potentially many others may then be used as the basis for quality assessment of Component archtiectures.

To enhance quality in architecture descriptions, acquisition PMs and C/S/A Chief Architects should ensure: 1) architecture data elements created and used in architecting tools are derived from or mapped to elements of the CADM, 2) authoritative reference data are used wherever possible and practical, 3) architects are thoroughly familiar with the DoDAF and trained in the use of architecting tools, and 4) SMEs are involved in the development, review, and approval of architecture descriptions.

2.7 ROLE OF DARS AS AN EXTENSION OF THE ENTERPRISE (METADATA) CATALOG

DARS provides EA content cataloging and discovery by registering content discovery metadata and implementing a modified version of the Enterprise Catalog federated search. Metadata registration and the federated search enables DARS to function as a virtual extension of the Enterprise Catalog. DARS implements a set of federation standards for cataloging and linking architecture content from any content repository implementing the federation standards. The standards include the DDMS plus the architecture community discovery metadata extension and web service specifications for content provider metadata registration and federated search.

2.8 ROLE OF DARS AS AN AUTHORITATIVE SOURCE/REPOSITORY FOR DODAF REFERENCE DATA

Authoritative reference data should be made visible and accessible to all DoD architects and unanticipated users to support the objectives of the NCDS. DARS supports this goal by associating discovery metadata with reference data sets and by providing services for loading, extracting, and version management of reference data.

Authoritative data sources can provide access to reference data sets via DARS communities. A DARS user group community may be identified as the authoritative source for a particular type of reference data. The Joint Staff, for instance, is identified as the authoritative source for the UJTL elements. The UJTL elements are available for export from DARS as a CADM XML record set. This record set can be used in any architecting tool environment to ensure that instances of process activities modeled in that tool environment are authoritative and will be consistent with process activities based on the same reference data in models created in other tool environments. This enables architecture data integration, since each independently developed model using the same reference data can be integrated via those common reference data elements.

Since reference data can change over time, it needs to have an associated version identifier, management authority identifier, and release or approval date to support the “Trust” aspect of the NCDS. These elements of metadata are associated with the reference data to enable discovery searches and assessment of the suitability for use. A “Data Configuration Manager” role provides exclusive access to edit user group reference data records. DARS user group community Data Configuration Managers are responsible for maintaining the version identity of all structured data

in DARS. The DARS community "Approval Authority" is responsible for releasing reference data with appropriate discovery metadata and access controls.

2.9 ROLE OF DARS AS AN AUTHORITATIVE SOURCE/REPOSITORY FOR DODAF REFERENCE ARCHITECTURES

Reference architectures provide approved C/S/A views of mission areas or subject domains and may also be approved for use by other architects in developing interfaces or extensions. DARS provides version management and release control for reference architectures to support the objectives of the NCDS and the needs of the DoD architecture community. It provides a shared space for posting authoritative reference architectures and data, thereby making accessible and trusted data available to the DoD community. DoD architects, analysts, and unanticipated users can search DARS for reference architectures and download products of interest in native tool formats or as CADM XML for analysis or reuse.

To support accessibility and trust, DARS community "Approval Authorities" can assign access control levels to reference architectures. DARS provides three levels of access control. A "Public" access control level provides metadata visibility and data access to all registered DARS users. A "Protected" access control level provides metadata visibility to all registered DARS users, but restricts data access only to users granted "Protected" access within a user group community by a community "Administrator." A "Private" access control restricts metadata visibility and data access only to users granted "Private" access within a user group community by a community "Administrator." The community "Administrator" role is granted by a user group community "Sponsor."

Having robust access controls and metadata ensure that both the producer and consumer can trust the data they are sharing.

2.10 DARS APPROACH TO REALIZING NCDS GOALS

2.10.1 Visible

DARS captures and stores discovery metadata for all products and data loaded into its database. Discovery metadata are cataloged and tagged using the DDMS. DDMS specifies the set of metadata elements needed to identify and locate data assets in a net-centric data environment and provides a standard set of XML tags for tagging the metadata. DARS provides a federated metadata search web service that can be called to expose all metadata cataloged for the architecture data assets held in DARS or in any federated architecture repository using DDMS XML tags. This web service enables discovery of all data assets in DARS and federated repositories, thereby satisfying the visibility requirement of the NCDS.

2.10.2 Accessible

Once data assets are located in DARS, the assets are made accessible via the DARS web site or its data extraction web service. DARS provides both data storage and extraction via web pages or web services. Presently, architecture products may be loaded into DARS in native file formats or as XML documents conforming to the DoD CADM XML schema. The DARS CADM XML loader provides an additional data management service by assigning globally unique enterprise identifiers (EIDs) to all records loaded into its database. EIDs are used in DARS to allow subject domain managers to create and maintain authoritative reference data that can be distributed to

independent domain architects for use in developing architecture models that will ensure consistent data, thereby enabling correlation and integration of the resulting products.

2.10.3 Understandable and Interoperable

Presently, structured architecture data is made understandable and interoperable via database conformance with the CADM standard. The CADM standard includes an XML schema specification and provides a complete description of the semantics of conforming architecture data. DARS also provides a structured data extractor that generates XML documents conforming to the CADM XML schema from structered data loaded into its database. DARS provides a structured data loader that loads and stores structured architecture data in conformance with the CADM. To enhance understandability and interoperability of architecture data developed using government and commercial architecture tools, OASD(NII) sponsors an Architecture Interoperability Program (AIP) that has established an interoperability certification program to assist tool developers in implementing the CADM XML specification for data exchange. Specifications for DARS interoperability are available in the AIP community in DARS. Tool developers interested in participating in the AIP and getting certified as DARS conformant can subscribe to the AIP community in DARS and contact OASD(NII) for more information.

An architecture federation strategy is presented in section 3.0 that details how architecture repositories will federate archtiectures across DoD. Architecture data are expected to be developed and maintained in a federation of repositories. The challenge for DoD is to ensure that tool vendors have the standards and support needed to implement means of creating and exposing both metadata and data conforming to the data standards of the architecture community to be able to interoperate so that architecture data, once discovered, can be interchanged and understood between the various tools in a federated environment.

2.10.4 Trusted

The trust aspect is concerned with ensuring that the sources of data can be identified and trusted. Trust requires that a potential user is able to identify the source, authority, release or approval status, access control, and quality aspects of the data. To ensure trust, DARS captures and catalogs metadata needed to establish the source and quality of data. Responsibility for approval should be in accordance with tiered accountability established within each C/S/A component. DARS offers a robust set of capabilities for setting up user groups and allowing access to, as well as release of, "approved" documents at the discretion of the community Approval Authority or Administrator, thereby imparting a high degree of trustworthiness in “approved” documents. Community Administrators are able to manage multiple product versions, thereby enabling users to identify and track version changes. At the data level, all record changes are logged to identify the user, time, and type of change. A data auditing feature can be used to roll back any changes made. DARS also tags data to control access based on user roles and need to know.

2.10.5 Responsive to User Needs

DARS provides a scalable, web-based capability that directly supports distributed EA development and analysis. DARS provides an interactive graphical environment and the ability to navigate the underlying CADM database by drilling down on graphic elements to extract details from the database. DARS supports version management of architecture products and reference data, thereby providing traceability of version changes and the pedigree of architectures and data, enabling determination of the applicability of architecture data for further

analysis and reuse across the DoD enterprise. Reuse of authoritative reference data sets and architecture products stored in DARS will enhance interoperability by enabling correlation and aggregation of architectures across DoD, thereby enabling interoperability assessments across multiple missions, systems, and theaters of operation. Data stored in DARS provides a suitable base for supporting many types of management decisions including capabilities assessment, gap analysis, portfolio management, system engineering, facilities management, and capital investment planning.

3 ARCHITECTURE METADATA AND FEDERATED ARCHITECTURE REGISTRY APPROACH

This section presents EA federation concepts and guidance based on both the NCDS and Federal policy on the development and use of EAs.

Architectures are currently developed independently by many organizations across DoD. This situation raises several issues for architects and architecture end users. First, both architects and architecture end users require the capability to globally search for architectures that may be relevant for analysis or specific architecture development efforts. Second, a consistent set of standards for architecture version management is needed to enable users to determine the development status, quality, and authority of architecture data. Third, a standard methodology is needed for specifying the alignment or linkages between architectures developed using different tools and maintained in independent repositories. A methodology for federating architectures must address these issues. The first two issues are addressed via metadata tagging and federated search. The third issue is addressed via content metadata linking.

The key net-centric principles that must be adhered to by the EA community are that data assets must be made visible, accessible, understandable, trusted, and must support interoperability. These principles do not assume or prescribe any requirements for physical data storage. Data may be stored in any format using relational, object oriented, or hybrid technologies based on any kind of data model. These principles do, however, require that agreements be reached within the DoD EA COI on the structure and semantics of data elements used for data asset discovery, linking, exchange, and integration. Metadata elements needed to support the EA user services described herein are defined and prescribed for the DoD EA COI as the standard for EA services.

3.1 CONCEPT SUMMARY

The federation approach detailed herein is based on the assumption that architectures will continue to be developed and maintained in independent repositories throughout DoD for the foreseeable future. A set of federation standards must be implemented by autonomous repositories to enable discovery, linking, and consistent version management of architecture data assets. The standards specified herein include a set of metadata to be maintained for all data assets in DoD architecture repositories and Web Services for discovery, registration, and quality assessment.

EA federation services will enable architects, analysts, planners, and unintended users to search for and access architecture data assets of interest and assess their interrelationships and suitability for use. The federated EA will be constructed via classification of data assets according to the DoD BRM at a minimum.

Architecture producers will use EA federation services to register minimum metadata required for describing architecture content, access restrictions, and approval status. The metadata registration service will also enable content linking to show alignment between EA components. This metadata will enable potential consumers to search for, evaluate suitability for intended use, and retrieve data of interest.

A Federated Discovery service is specified to enable users to execute federated searches for architectures meeting specified search parameters.

A Metadata Registration service is specified to enable cataloging and linking of architectures in federated repositories. The Registry service will enable architects using independent tool and repository environments to specify linkages between architecture data assets developed and maintained in local repositories and nodes of approved classification taxonomies, which will show alignment of component architectures and enable users to navigate a federated EA.

Mechanisms for quality assessment services are recommended to ensure data consistency and facilitate aggregation, integration, and assessment.

3.2 USE CASES

Potential architecture users include combatant commanders, operations planners, acquisition managers, systems engineers, and budget analysts. In many cases, architecture data users are both producers and consumers of architecture data. All can benefit from an *awareness of* and *access to* relevant architecture assets that may support planning or decision processes. Equally beneficial is the ability to identify the source, quality, and authority of architecture data assets and their interrelationships and dependencies.

3.2.1 Data Consumer

The federated EA services will support the following *data consumer* requirements:

- Ability to find all architecture data that may be relevant to a decision-making process or potentially applicable to a new architecture project
- Ability to evaluate the sufficiency, quality and authority of the data,
- Ability to identify dependencies and relationships between various architecture data assets

Users may first want to ascertain what architecture data assets are available that may be relevant to their problem or situation. Users should have the following options for finding architecture data assets of interest:

- Ability to execute a search for data meeting specific search criteria
- Ability to browse a catalog of data assets available in all architecture repositories

3.2.2 Architecture Search

The architecture search capability must enable a user to specify a set of criteria for architecture data assets of interest. That set of search criteria needs to be propagated to all architecture data repositories. The user must receive a consolidated response that:

- Provides sufficient metadata on data assets meeting the search criteria to enable the user to ascertain their relevance
- Provides links to the sources

The federated search service must be able to accept the following user search criteria at a minimum:

- Subject key words for a category (i.e., classification taxonomy)
  - Joint Capability Area = Joint C2
  - DoDAF product type = OV-5
- Releasing organization name [=/like]

- Approval authority organization name [=/like]
- Approval date [before/after]
- Effective Start date [before/after]
- Effective End date [before/after]
- Architecture Scope = [Mission, Functional, Enterprise, Program]
- Temporal Scope = [As-is, To-be]
- Completion Status = [Under development, Review draft, Complete]
- Use Type = [Baseline, Actual, Target]
- View Type = [All, Operational, System, Technical, Other]

The federated search service must enable users to specify each of these search criteria in a search interface using pick lists for the allowable values for each of the criteria. The search interface should allow specifying Boolean operations on multiple criteria (e.g., Releasing Organization Name = SOUTHCOM *OR* SOCOM *AND* Temporal Scope = "To-be").

The search response must provide consistent metadata from all sources. The response should include all available architecture metadata, including any available metadata not specified as search parameters. For example, a user may want to find all OV-5s for Joint C2. The user should only need to specify the subject key words "DoDAF product type = OV-5 *AND* Joint Capability Area = Joint C2." However, the response should provide all available architecture metadata for the matching results from each source. The additional metadata may be evaluated by the user for determining the potential interest value of the matching results (e.g., products with Completion status = "Under development" may be of lesser interest than products with Completion status = "Complete").

Once a user determines which architecture products are of interest, a link associated with each result will enable the user to access the content in the source repository. Depending on the security policy of the source repository, the link may either provide direct access to the selected product in the native repository format or visualization environment, or it may direct the user to a role subscription service on the native repository for requesting access to the product. User credentials should be passed by the search service to the source repository for authentication to enable automated access based on access roles/privileges associated with the user in the source repository.

3.2.3 Registry Browsing

Users may wish to browse a catalog of data asset holdings in federated repositories to find assets of interest. Cataloging of repository holdings via metadata will enable users to search for data assets by navigating classification taxonomies similar to browsing item classifications in on-line shopping web sites. Architecture classification schemes suitable for the DoD EA community include categorization by JCA, Joint Mission Areas (JMA), Missions, UJTL, DoDAF product type, functional area, acquisition program, transformation architecture, as well as others. Multiple classifications may apply to any single architecture. Catalog navigation trees will use standardized category names from classification taxonomies**,** when available, and may be extended by domain extensions, where needed. Navigation trees will also provide links to detailed metadata on architecture data assets to enable users to determine potential interest value and include links to the products in source repositories.

3.2.4 Data Producer

Various types of architecture repositories based on COTS and GOTS tools and databases are used throughout the DoD. Existing repositories typically collect and maintain metadata on architecture projects. Repository managers must ensure metadata are maintained on all architecture data assets that might be specified as search parameters by potential users. Most pertinent metadata are typically included in the AV-1 for an architecture project. Metadata searching will require AV-1 metadata to be captured and stored as structured data in federated repositories to enable searching. Metadata must also be recorded on architecture project status including creation date, date last modified, approval date, and completion status to enable users to determine currency, authority, and applicability of the project data.

The EA federation services must support the following *data producer* requirements:

- Ability to capture all architecture discovery metadata for each architecture data asset
- Ability to align or link architecture data assets under one or more classification taxonomies, including, at a minimum, the DoD BRM
- Ability to create, identify, manage, and provide access to authoritative Reference Data
- Ability to assess data quality in terms of conformance with the CADM, DoDAF product specifications, and the use of authoritative reference data

3.3 FEDERATION SERVICES

EA federation services described in this section have been implemented in DARS and several other federated architecture repositories. Detailed specifications for the web services, including Web Service Description Language (WSDL) specifications, XML schemas for metadata, and sample client code, are maintained in DARS and are available for download.

3.3.1 Registration

To implement a federated search service, metadata elements must be defined to capture required search parameters. The DDMS v1.3 provides the baseline specification for architecture discovery metadata. Additional discovery metadata specifications provided by the CADM are also used. Several new metadata elements, defined as DARS metadata, are specified to enable version management, architecture registration, and cataloging in DARS. Table 3-1 in section 3.3.5 provides a summary of the applicable architecture metadata elements.

A metadata registration web service has been implemented in DARS to enable DoD architecture data asset producers to register and catalog content metadata in DARS via a web service client application. This service should be used for registration of all available architecture metadata specified in **Table 3-1** as soon as it is created in order to support the “post in parallel with processing” aspect of the NCDS. A metadata registration user interface has been implemented in DARS and several federated repositories to enable manual and semi-automated entry of architecture metadata. However, the registration web service specification may be used for automated machine-to-machine transfer of metadata. Architecture tool developers should use automated processes to extract architecture metadata from the tool environment and invoke the registration web service automatically as soon as the minimum required metadata has been collected for an architecture project.

Each federated repository is responsible for implementing a registration client conforming to federation registration service specifications. A registration client should format and send new registrations or registration update requests to DARS by invoking the registration web service whenever a new architecture is created (saved) or a registered architecture's metadata are modified. Registration requests will provide all available metadata for the architecture. The DARS registration web service will automatically process registration requests from federated repositories, when received. Once processed, DARS sends a registration identifier back to the registration client to use to reference the architecture in future registration update requests.

Registration clients should provide a pick list-driven user interface for selecting the classification taxonomy elements that specify associations for the architectures being registered. An architecture link to a classification taxonomy element should identify the type of association as one of ["Is equivalent to," "Is part of," "Supports," or "Replaces"]. The taxonomy element association is used by the DARS registration service to catalog the data assets.

To support discovery, federation repositories must capture all mandatory DDMS metadata and any additional architecture metadata specified for architectures developed in local tool environments. Each legacy COTS/GOTS tool and repository environment typically maintains a subset of the required architecture metadata. Some of the metadata can be captured automatically, from context and some can be extracted from the AV-1 for the architecture project. Federation repository owners must implement mechanisms to collect any missing metadata by making extensions to the tool/repository metamodel and/or user interface.

3.3.2 Discovery

A federated search user interface has been implemented in DARS and several other federation repositories. Users may input search parameters via either 1) a pick list-driven user interface in DARS, or 2) a federated search web service client interface in the local tool environment if implemented. Subject Area Coverage key words must be selected from DoD BRM Mission Area activity lists to support content classification in accordance with the BRM. The search interface may enable specifying Boolean operations on selected parameters. For federated repositories that implement a local search client, the DARS federated search web service will send result sets back to the local client. The search results will consist of records matching the search criteria and will contain as much of the architecture metadata as is available from federated repositories. Uniform Resource Locators (URLs) in the metadata result set will provide links to the source architectures in the confederate repositories.

3.3.3 Version Management

A standard set of version management metadata are required and specified for federated repositories to provide consistency in search result sets and enable users to determine the development status and authority of architecture data. Federation repository owners must implement role-based controls and processes for creating and editing version identification and status metadata in accordance with the federation standards to ensure adequate control (trust) of the data.

3.3.4 Cataloging and Linking

A registry of data asset holdings has been implemented in DARS to provide architects and architecture users with one-stop shopping for architecture data. The DoD BRM Mission Areas are used as the primary cataloging taxonomy for DoD EA components. Other classification taxonomies may be specified and authorized for classification of architecture data assets.

Authoritative Reference Data Sets should be specified for use in architecture descriptions for data elements that are needed for consistency in establishing links between EA artifacts. Optional classification taxonomies may include the following:

- Combatant Command (COCOM) Architectures: List of COCOMS with Joint Task Force (JTF)/Mission Architectures as second tier
- Program Architectures: List of PEOs and PMs with core program names as second tier
- Transformation Architectures: List of core transformation architectures

The DARS registration service catalogs and links registered architectures in federated repositories via classification taxonomy elements specified in the Subject Area Coverage metadata element. Architecture registration and linking enables users to browse for architecture data assets in DARS and federated repositories by navigating architecture classification trees.

3.3.5 Metadata Elements

Table 3-1 lists prescribed EA metadata elements and their specification sources. All elements listed are candidate search parameters for the federated search and should be captured by all confederate repositories. To provide consistency, federation conventions must be followed for date and version formats. Complete and current specifications for all federated EA metadata elements, including XML schemas, are available for download from DARS.

Architecture metadata example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) -->
<ddms:Resource xmlns:ddms="http://ddms.dod.mil"
xmlns:ICISM="urn:us:gov:ic:ism:v2" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dars="http://dars.ddms.dod.mil"   xmlns:cadm="http://cadm.ddms.dod.mil"
xsi:schemaLocation="http://dars.ddms.dod.mil DDMS_DARS_Extension.xsd">
      <ddms:identifier ddms:qualifier="docId"
ddms:value="72064646457648748"/>
      <ddms:identifier ddms:qualifier="URL"
ddms:value="https://dars1.army.mil"/>
      <ddms:identifier ddms:qualifier="version" ddms:value="1.01"/>
      <ddms:title ICISM:classification="U" ICISM:ownerProducer="USA">T-AKE
C4ISP Final.doc</ddms:title>
      <ddms:description ICISM:classification="U"
ICISM:ownerProducer="USA">Lewis And Clark (T-AKE) Class Auxiliary Cargo and
Ammunition Ship C4I Support Plan (C4ISP).</ddms:description>
      <ddms:language
ddms:qualifier="http://metadata.dod.mil/mdr/ns/ExtStd/iso_639-2b.owl#en"
ddms:value="en"/>
      <ddms:dates ddms:posted="2004-05-13Z"  ddms:created="2006-07-
03T14:36:33Z"  ddms:infoCutOff="2004-05-13Z" />
      <ddms:rights ddms:copyright="true" ddms:privacyAct="false"
ddms:intellectualProperty="false"/>
      <ddms:type ddms:qualifier="fileCategory"  ddms:value="XML" />
      <ddms:creator ICISM:classification="U" ICISM:ownerProducer="USA">
            <ddms:Person >
                  <ddms:name>John Doe</ddms:name>
                  <ddms:surname>Doe</ddms:surname>
                  <ddms:userID>72064646457648700</ddms:userID>
                  <ddms:affiliation>SPAWAR</ddms:affiliation>
```

```
                  <ddms:phone>619-524-3674</ddms:phone>
                  <ddms:email>john.doe@navy.mil</ddms:email>
            </ddms:Person>
      </ddms:creator>
      <ddms:publisher ICISM:classification="U" ICISM:ownerProducer="USA">
            <ddms:Organization  ddms:orgId= "67745383939" >
                  <ddms:name>SOUTHCOM</ddms:name>
                  <ddms:phone>305-234-4567</ddms:phone>
                  <ddms:email>richard.smith@southcom.mil</ddms:email>
                  </ddms:Organization>
      </ddms:publisher>
      <ddms:publisher ICISM:classification="U" ICISM:ownerProducer="USA">
            <ddms:Person>
                  <ddms:name>Alton</ddms:name>
                  <ddms:surname>Levis</ddms:surname>
                  <ddms:userID>72064646457648700</ddms:userID>
                  <ddms:affiliation>southcom</ddms:affiliation>
                  <ddms:phone>305-369-3616</ddms:phone>
                  <ddms:email>alton.levis@southcom.mil</ddms:email>
            </ddms:Person>
      </ddms:publisher>
      <ddms:publisher ICISM:classification="U" ICISM:ownerProducer="USA">
            <ddms:Organization  ddms:orgId= "67745383939"
dars:orgRoleType="COMMUNITY  VERIFICATION/ VALIDATION EXAMINER">
                  <ddms:name>SOUTHCOM</ddms:name>
                  <ddms:phone>305-234-4567</ddms:phone>
                  <ddms:email>katherine.holt@southcom.mil</ddms:email>
                  </ddms:Organization>
      </ddms:publisher>
      <ddms:contributor ICISM:classification="U" ICISM:ownerProducer="USA">
            <ddms:Organization  ddms:orgId= "67745383939"
dars:orgRoleType="COMMUNITY  PRODUCER">
                  <ddms:name>CIO/G6</ddms:name>
                  <ddms:phone>703-234-4567</ddms:phone>
                  <ddms:email>paul.snyder@us.army.mil</ddms:email>
                  </ddms:Organization>
      </ddms:contributor>
      <ddms:pointOfContact>
            <ddms:Person>
                  <ddms:name>James</ddms:name>
                  <ddms:surname>Baker</ddms:surname>
                  <ddms:userID>72064646457648700</ddms:userID>
                  <ddms:affiliation>JCS_J8</ddms:affiliation>
                  <ddms:phone>703-605-3674</ddms:phone>
                  <ddms:email>james.baker@js.pentagon.mil</ddms:email>
            </ddms:Person>
      </ddms:pointOfContact>
      <ddms:format>
            <ddms:Media>
                  <ddms:mimeType>text/html</ddms:mimeType>
                  <ddms:extent
ddms:qualifier="http://metadata.dod.mil/mdr/ns/UnitOfMeasure/0.1/ComputerStor
age.owl#byte" ddms:value="871936"/>
                  <ddms:medium>digital</ddms:medium>
            </ddms:Media>
      </ddms:format>
      <ddms:subjectCoverage>
```

```
            <ddms:Subject>
                  <ddms:category ddms:label="mission area 1"/>
                  <ddms:category
ddms:qualifier="https://dars1.army.mil/JointCapabilityAreaList#capabilityArea
1" ddms:code="capabilityArea1" ddms:label="capability area 1"/>
                  <ddms:category
ddms:qualifier="https://dars1.army.mil/dodafProductTypeList#ov2"
ddms:code="ov2" ddms:label="ov-2"/>
                  <ddms:category
ddms:qualifier="https://dars1.army.mil/dodafProductTypeList#ov3"
ddms:code="ov3" ddms:label="ov-3"/>
                  <ddms:category
ddms:qualifier="https://dars1.army.mil/dodafProductTypeList#ov4"
ddms:code="ov4" ddms:label="ov-4"/>
                  <ddms:category
ddms:qualifier="https://dars1.army.mil/dodafProductTypeList#ov5"
ddms:code="ov5" ddms:label="ov-5"/>
                  <ddms:keyword ddms:value="Lewis and Clark"/>
                  <ddms:keyword ddms:value="C4ISP"/>
                  <ddms:keyword ddms:value="ammunition"/>
                  <ddms:keyword ddms:value="ship"/>
            </ddms:Subject>
      </ddms:subjectCoverage>
      <ddms:virtualCoverage ddms:address="ff" ddms:protocol="ff" />
      <ddms:virtualCoverage ddms:address="ff" ddms:protocol="ff" />
      <ddms:temporalCoverage>
            <ddms:TimePeriod>
            <ddms:name>jhgjhg</ddms:name>
                  <ddms:start>2006-04-13T20:57:51Z</ddms:start>
                  <ddms:end>2004-05-17T10:11:00.990Z</ddms:end>
            </ddms:TimePeriod>
      </ddms:temporalCoverage>
      <ddms:temporalCoverage>
            <ddms:TimePeriod>
            <ddms:name>2010</ddms:name>
                  <ddms:start>Not Applicable</ddms:start>
                  <ddms:end>2004-05-17T10:11:00.990</ddms:end>
            </ddms:TimePeriod>
      </ddms:temporalCoverage>
      <ddms:security ICISM:classification="U" ICISM:ownerProducer="USA"
ICISM:dateOfExemptedSource="2006-12-01"/>
      <dars:darsExtension>
      <dars:documentAccessLevelCode>PUBLIC</dars:documentAccessLevelCode>
            <dars:dates  modified="2004-05-17T10:01:00.999Z"  />
            <dars:cadmExtension>
                  <cadm:DOC_APP_CALDT>2004-05-
17T10:01:00.999Z</cadm:DOC_APP_CALDT>
                  <cadm:ARCH_ID>646479326345</cadm:ARCH_ID>
                  <cadm:ARCH_NM>CID Architecture</cadm:ARCH_NM>
                  <cadm:ARCH_START_DT>56678</cadm:ARCH_START_DT>
                  <cadm:ARCH_END_DT>56678</cadm:ARCH_END_DT>
                  <cadm:ARCH_LVL_CD>2</cadm:ARCH_LVL_CD>
                  <cadm:ARCH_TEMP_SCOPE_CD>2</cadm:ARCH_TEMP_SCOPE_CD>
                  <cadm:ARCH_COMPL_STA_CD>2</cadm:ARCH_COMPL_STA_CD>
                  <cadm:ARCH_USE_TY_CD>2</cadm:ARCH_USE_TY_CD>
                  <cadm:ARCH_VW_TY_CD>2</cadm:ARCH_VW_TY_CD>
            </dars:cadmExtension>
```

```
      </dars:darsExtension>
</ddms:Resource>
```

Table 3-1: Architecture Metadata for Classification, Discovery, and Version Management

Metadata Element	Metadata Specification Source
• Identifier* (URL)	(DDMS Identifier Qualifier = URL, Value)
• File name	(DDMS Identifier Qualifier, Value)
• Security Classification*	(DDMS: Security)
• Title* (Architecture Name)	(DDMS: Title)
• Subject Key Words*	(DDMS: Subject)
• Joint Mission Area	(DDMS: Category Qualifier + Category Label)
• Joint Capability Area	(DDMS: Category Qualifier = "JCA List" + Category Label)
• DoDAF Product type	(DDMS: Category Qualifier = "DODAFPRODTY" + Cat Label)
• Creator*	(DDMS: Creator.Person)
• Publisher (COI)	(DDMS: Publisher)
• Approval Authority	(DDMS: Contributor Organization)
• Creation Date	(DDMS: Date.Created)
• Last Modified Date	(DARS: Date.Modified)
• Approval Date	(CADM: *DOC_APPR_CALDT*)
• Effective Start Date	(DDMS: Temporal Coverage Start)
• Effective End Date	(DDMS: Temporal Coverage End)
• Version	(DDMS: Identifier Qualifier= Version, Value)
• Access Level	(DARS: AccessLevel) [Public, Protected, Private]
• Architecture Scope	(CADM: *ARCH_LVL_CD*) [Enterprise, Mission, Functional, Program]
• Taxonomy Element Association	(DARS: TaxonomyName [e.g., JCA, JMA, COAL, CSFL,] + OrdinateElementName + SubordinateElementName + AssociationType [e.g., Part of, Replaces, etc.])
• Temporal Scope	(CADM: *ARCH_TEMP_SCOPE_CD*) [As is, To be, NA]
• Completion Status	(CADM: *ARCH_CMPLTN_STA_CD*) [Under development, Review draft, Complete]
• Use Type	(CADM: *ARCH_USE_TY_CD*) [Baseline, Actual, Target]
• View Type	(CADM: *ARCH_VIEW_CAT_CD*) [All, Operational, System, Technical, Other]
• Format	(DDMS: format.Media.mimeType)

* Mandatory DDMS metadata

Access Control Level will be used as it is in DARS to control access to architecture data assets and data.

Architecture Scope will be used to identify top-level categories for cataloging taxonomies. Multiple taxonomies may apply for each scope category.

Taxonomy Element Association will be used to identify linkages between classification taxonomy elements and the confederate architecture data assets. Multiple linkages may be specified.

4 CORE ARCHITECTURE DATA MODEL V1.5

4.1 OVERVIEW

The major elements of a "core architecture data model" are described as follows:

"*Core*": The essential elements of architecture information that need to be developed, validated, and maintained and that should be sharable across architecture concerns to achieve architecture goals (e.g., interoperability, investment optimization).

"*Architecture Data*": The possible piece-parts of architecture products and related analytical tools in a rigorous definition of the pieces (object classes), their properties, features, or attributes, and inter-relationships.

"*Data Model*": A data model defines the objects of a domain, their inter-relationships, and their properties, normally for the purpose of a database design. There are three data model levels, from highest to lowest: Conceptual, Logical, and Physical. Conceptual models are the highest level. They model the user concepts in terms familiar to users. Details may be left out to improve clarity and focus with users. Logical models are more formal, often with considerations of unique data representation (non-redundancy or "normalization"), emphasis on semantic well-definedness and exclusivity (non-overlapping entities), and domain-level completeness. Logical models need not commit to a specific Data Base Management System (DBMS). The OV-7 is a logical data model. Physical models are usually the most detailed and the level sufficient for database generation. The Physical model must contain all the information necessary for implementation. The Physical model often addresses performance considerations. The SV-11 is a physical data model.

CADM v1.5 presented in this volume represents the initial baseline. Although major changes are not anticipated, the final release may vary from this version. Conformance is not required until the final release is posted in the DISR. This section describes the CADM at the Logical and Physical level. CADM is the data representation of the DoDAF "product" models defined in Volume II. The Conceptual view describes the principal entities of DoDAF models and the inter-relationship between them. The Logical view goes on to describe the attributes of those entities and provide more detail regarding the inter-relationships. The Physical view goes on to specify table, field, and relationship properties at a level of specificity sufficient to generate and implement a database. The Logical and Physical views also contain business rules.

Data modeling tools allow for the specification of subsets of CADM pertaining to each DoDAF model-based product. CADM "product" subviews are contained in Volume II, alongside the product description to which they pertain.

4.2 CADM DESIGN AND MAINTENANCE PRINCIPLES

CADM may be implemented in multiple target environments, e.g., implementations exist in Microsoft (MS) Access, Structured Query Language (SQL) Server 2000, and Oracle DBMS. CADM conformance means the following:

a. The conforming model is based on a subset of the CADM (neither all the entities nor all attributes of selected entities have to be part of the chosen subset).

b. Extensions of that subset are allowed (but should not be redundant with elements of the CADM itself); extensions that could apply to the CADM for general use should be proposed for incorporation into the model.

c. Agreed datatypes and coded domains must be used.

d. Points of contact should be identified and consulted when generating instances of keys (to avoid redundancy and non-uniqueness).

e. Primary key attributes for entities taken from the CADM should be identical with or directly derivable from the primary key attributes specified in CADM (alternate keys may be used but CADM keys need to be preserved).

f. Keys for authoritative data source instances should be retained to enable effective updates from those sources. The goal of CADM conformance is to ensure fully faithful information transfer among databases, which cannot happen if the primary keys of one database have no correlation to the primary keys of another database for the same entity.

Architecture data repositories that conform to CADM have the ability to compare and share architecture data across architecture repositories and databases. Non-conforming repositories require translation and data correlation and reconciliation. Translation losses and infeasible reconciliations can occur. Translation, data correlation, and reconciliation costs and impacts are typically underestimated. For these reasons, development of architecture data in non-conforming data repositories, databases, or tools should be carefully considered and avoided whenever possible.

4.3 DESCRIPTION OF THE CADM V1.5 LOGICAL MODEL

4.3.1 The CADM v1.5 Superstructure

As discussed in Volume II, the new version of CADM consists essentially of two components:

- A superstructure made of a set of high-level entities, namely, Object, ObjectVersion, ObjectVersionStructure, ObjectVersionStructureDetail, and ObjectVersionStructureAssociation, and five subtypes of ObjectVersion: ObjectItem, ObjectType, ObjectVersionAssociation, ObjectByReference, and ArchitectureElement
- A series of subtype hierarchies under ObjectItem, ObjectType, and ArchitectureElement

Figure 4-1 shows the IDEF1X representation of the CADM v1.5 superstructure.

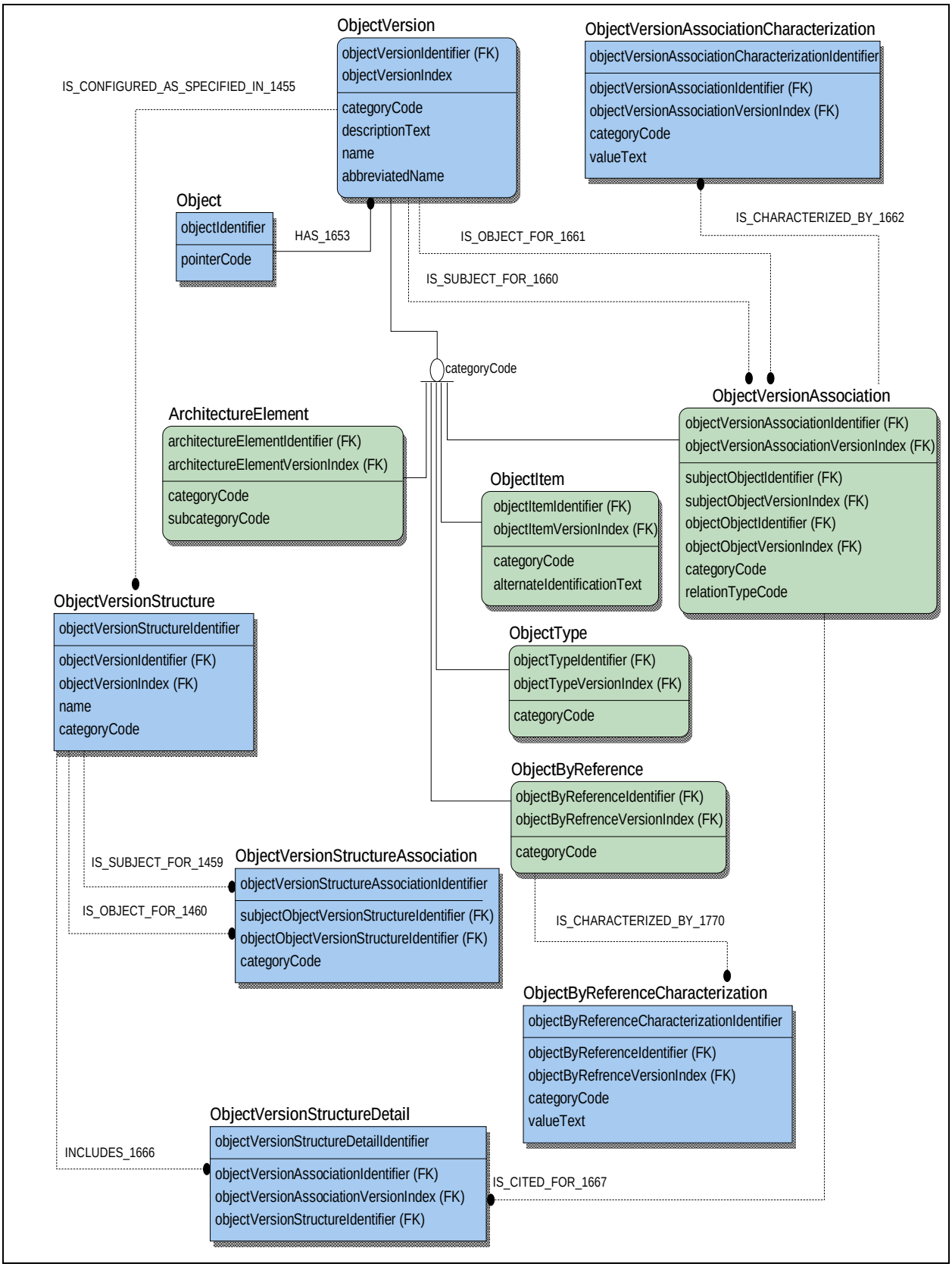


Figure 4-1: High-level Representation of CADM v1.5

4.3.2 The CADM v1.5 Subtype Hierarchies

The subtypes under ObjectItem are shown at the entity level in **Figure 4-2**.

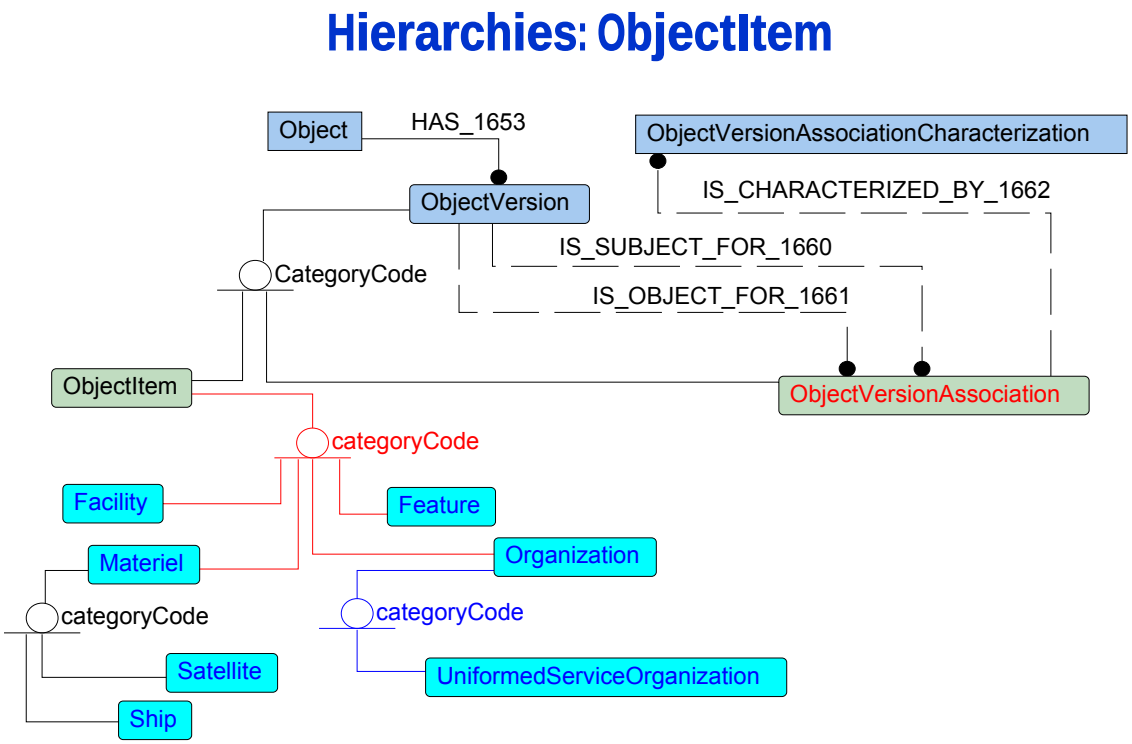


Figure 4-2: The ObjectItem Subtype Hierarchy in CADM v1.5

The subtypes under ObjectType are shown in **Figure 4-3**.

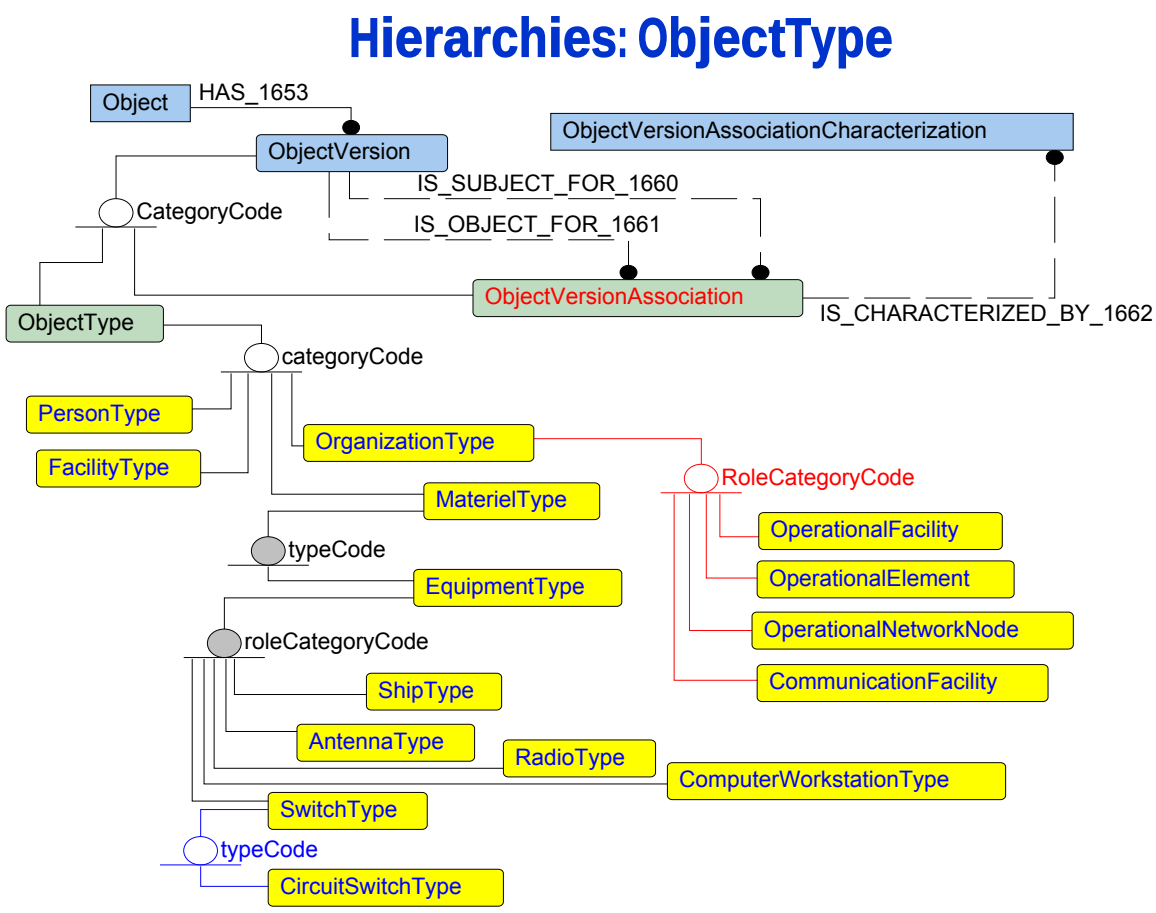


Figure 4-3: The ObjectType Subtype Hierarchy in CADM v1.5

All the remaining, explicitly modeled, CADM v1.5 entities are part of the subtype hierarchy under ArchitectureElement. **Figure 4-4** shows a partial set of those subtypes.

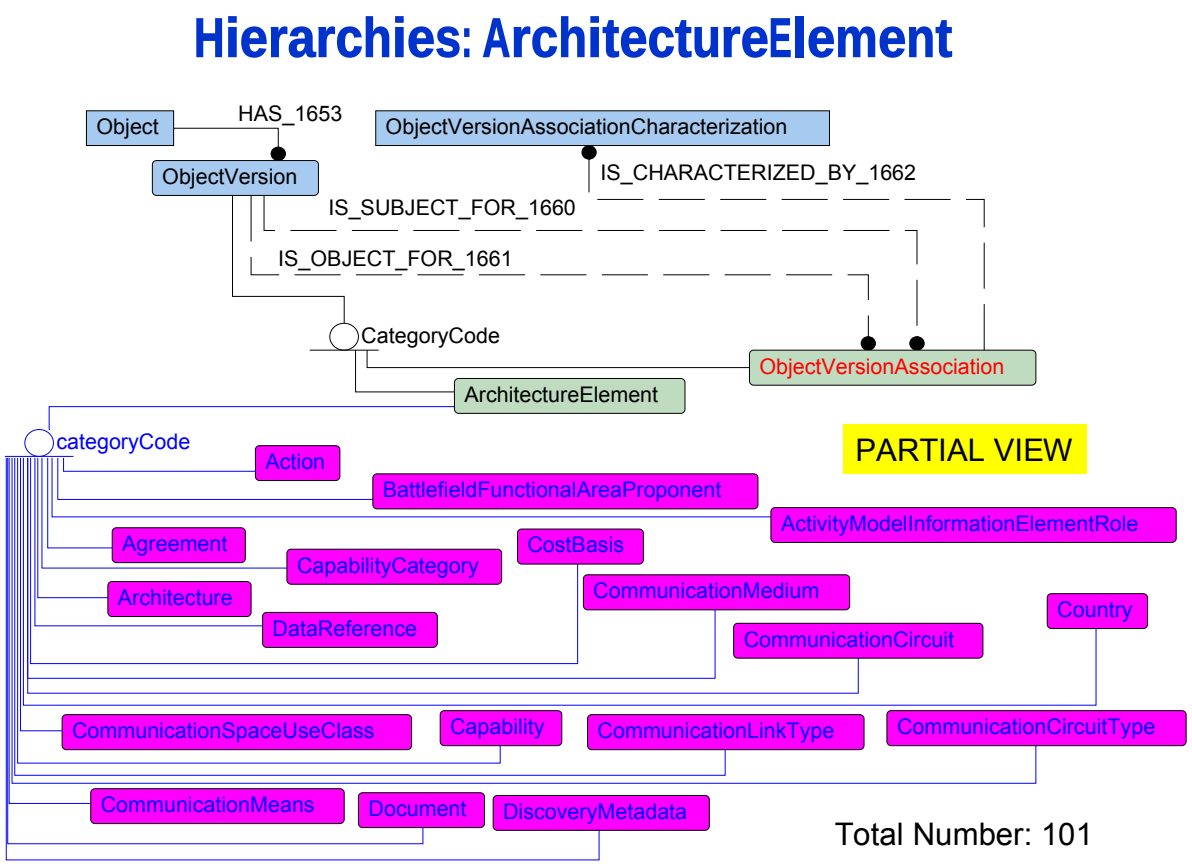


Figure 4-4: A Partial View of the ArchitectureElement Subtype Hierarchy in CADM v1.5

4.4 USE OF THE CADM V1.5 SUPERSTRUCTURE

4.4.1 Creation of the Physical Schema in a Relational Database

Both the entities in the superstructure as well as those in the hierarchies shown in Figures 4-2, 4-3, and 4-4 above are specified in CADM v1.5 at a level of detail sufficient to generate the physical schema within a relational database management system (RDBMS). **Figure 4-5** shows the "physical" view of the CADM v1.5 superstructure corresponding to the entity-attribute view shown in Figure 4-1.

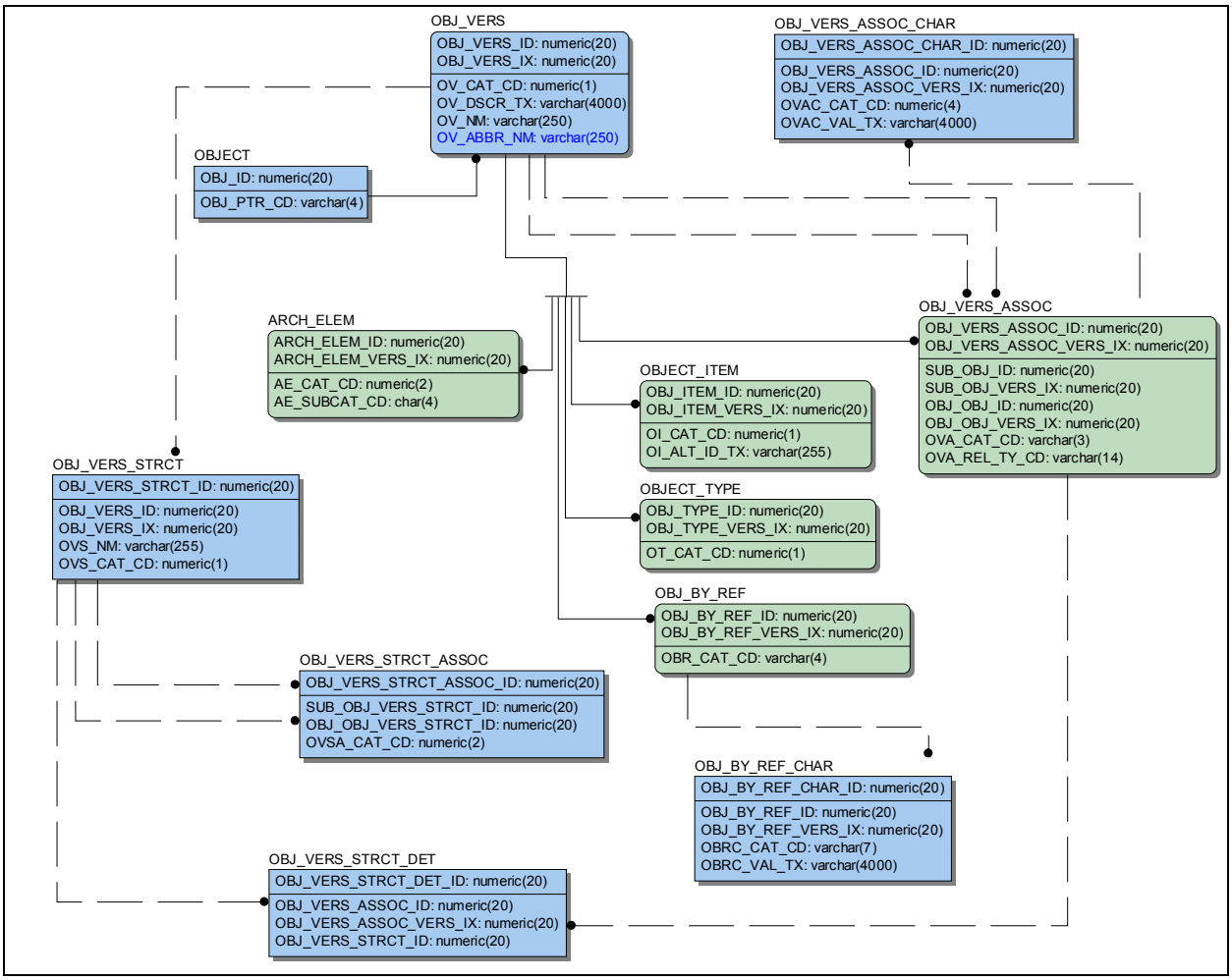


Figure 4-5: Physical Specification of the CADM v1.5 Superstructure Components

Each of the boxes represents a table in the physical schema of the target RDBMS, and each of the entries in the tables corresponds to columns in the respective tables. The transformation of the graphical representation into the actual schema is accomplished by generating an SQL script that expresses the Integrated Definition for Data Modeling (IDEF1X) notation into corresponding statements of the Data Definition Language (DDL) of the target RDBMS. When such a script is executed, the RDBMS creates a data structure that corresponds to the table in question. **Figure 4-6** shows the three stages described above: (a) IDEF1X physical specification, (b) SQL Script, and (c) Instantiated table using the RDBMS MySQL.

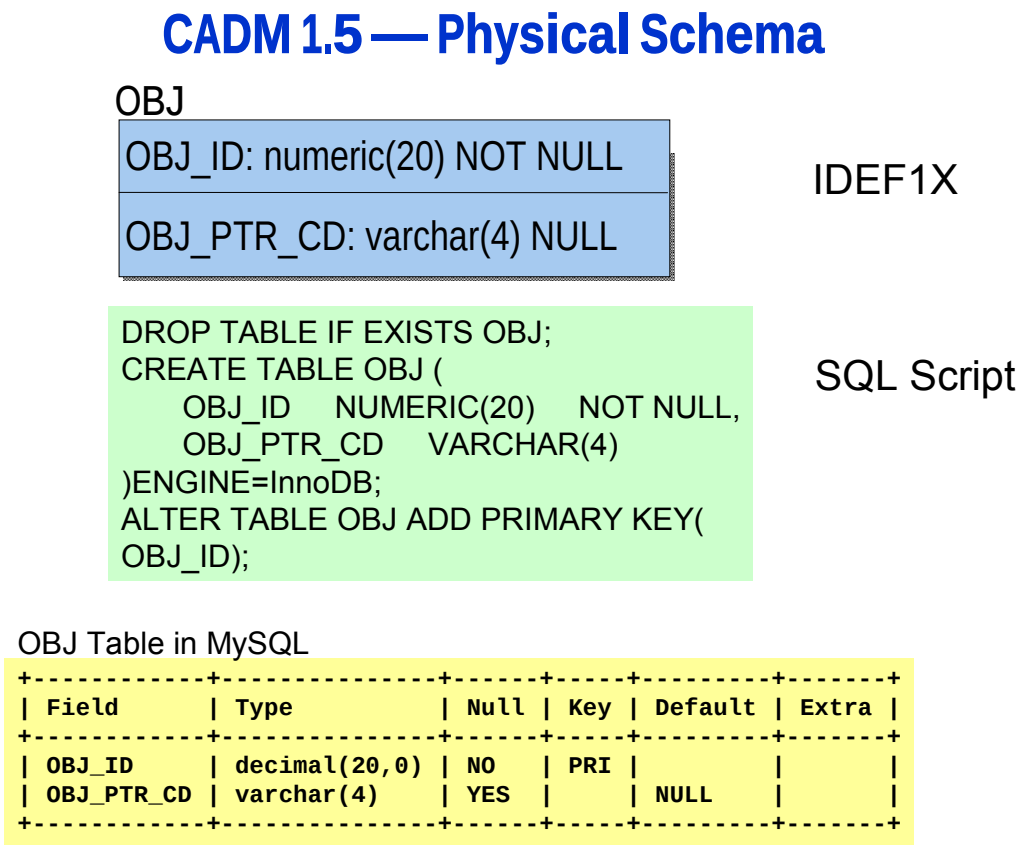


Figure 4-6: Transformation Stages for RDBMS Use

4.4.2 Data Loading

Once the physical tables have been created, one can load data in them. This is accomplished through the use of data manipulation language (DML) statements. **Figure 4-7** shows how two new instances of Object are created in MySQL via the INSERT statement. The column OBJ_ID contains the value required for the RDBMS engine to retrieve the data. This value, the so-called record key, must be, at a minimum, unique within each table. If one uses enterprise-wide globally unique identifiers, then no two record keys are the same for any table.

Figure 4-7 also shows that in CADM v1.5 that when one creates an instance of Object, one can already indicate the class of Object that it corresponds to. The example shows the creation of a record corresponding to Architecture and another corresponding to Document, both subtypes of ArchitectureElement.

All records in an RDBMS that implement CADM v1.5 must start as entries in the table OBJ, since all entities in CADM v1.5 depend directly or through one or more intermediate entities on the entity Object.

```
INSERT INTO OBJ(OBJ_ID,OBJ_PTR_CD) VALUES
      (337,'E038'),
      (876,'E148');
```

```
mysql> SELECT * FROM OBJ;
+--------+--------------------+
| OBJ_ID | OBJ_PTR_CD         |
+--------+--------------------+
|    337 | E038[Architecture] |
|    876 | E148[Document]     |
+--------+--------------------+
```

Figure 4-7: Creation of Records in CADM v1.5

4.4.3 Versioning

As the logical view of the CADM v1.5 depicted in **Figure 4-8** shows, every instance of Object has a one-to-many relationship to ObjectVersion.[4] This allows for the retention of the "semantic identity" of an instance of Object while, at the same time, allowing for modifications in the values of the attributes of said instance. One can think of this as analogous to the fact that for a person its social security number does not change even though its weight may change.

In CADM v1.5, if one needs or wishes to keep track of changes in an instance of Object while retaining any previous information linked to the same instance, every modification is treated as a new version of the same Object.

Figure 4-8 shows a notional case pertaining to the instance of Object with OBJ_ID = 337. The table OBJ_VERS shows two entries both linked to the same record belonging to the class of Architecture. In the second version of it, there is a change in the name, but otherwise everything else is meant to be the same. Because the key of the table OBJ_VERS is a composite key made up of the concatenation of the values of OBJ_ID and OBJ_VERS_IX, the records can be distinguished by the RDBMS engine when executing a SELECT statement.

4.4.4 Expressing Relationships in CADM v1.5 – Mapping of Foreign Keys

With the exception of the 10 one-to-many relationships among the entities of the CADM v1.5 superstructure, shown in Figure 4-5, all other relationships in the model are subtype relationships.

To link an instance of Object to another instance of Object in CADM v1.5, one must use ObjectVersionAssociation. There are three cases. The first is when the relationship between the

[4] The values of the **categoryCode** in **ObjectVersion** are: 1 = **ObjectItem**; 2 = **ObjectType**; 3 = **ObjectVersionAssociation**; 4 = **ArchitectureElement**; 5 = **ObjectByReference**

instances of Object has a fixed semantic meaning and there is no further characterization of the relationship. For example, an instance of CommunicationMedium may be linked to an instance of InformationExchangeRequirement with the fixed role "*is used to transport.*"

In previous versions of the model, this was expressed adding an extra column in the table InformationExchangeRequirement to capture the key of the corresponding CommunicationMedium serving that purpose. To capture this fact in CADM v1.5, one first creates in Object and ObjectVersion both the corresponding entries of the CommunicationMedium and InformationExchangeRequirement, as well as an instance of ObjectVersionAssociation. The tables below show a notional sample instantiation.

CADM 1.5 — Versioning

```
OBJ
+--------+--------------------+
| OBJ_ID | OBJ_PTR_CD         |
+--------+--------------------+
|    337 | E038[Architecture] |
|    876 | E148[Document]     |
+--------+--------------------+
```

```
OBJ_VERS
+-------------+-------------+-----------------+--------------------------+
| OBJ_VERS_ID | OBJ_VERS_IX | OBJ_VERS_CAT_CD | OBJ_NM                   |
+-------------+-------------+-----------------+--------------------------+
|         337 |           1 |               4 | Transcom Communications  |
|             |             |                 | Systems Architecture     |
|         337 |           2 |               4 | Transcom Communications  |
|             |             |                 | Systems Architecture/B   |
+-------------+-------------+-----------------+--------------------------+
```

Figure 4-8: Versioning of Records in CADM v1.5

Object

objectIdentifier	pointerCode
315	E102[CommunicationMedium]
316	E234[IER]
317	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
315	1	Satellite Link AFX-176	4[ArchElem]
316	1	Node A102 to Node A345	4[ArchElem]
317	1	CommMedium for IER 316	3[OVA]

As stated above, since there are no migrated keys in CADM v1.5 to relate the instance of CommunicationMedium to the instance of InformationExchangeRequirement, one now must use the ObjectVersionAssociation table. The instance table below shows the entries required.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
317	1	315	1	316	1	999	E102-R-E234

Note that the following convention is used: The parent of the relationship (in the example shown above, this is the instance of CommunicationMedium) must be entered as the *subject* in the ObjectVersionAssociation table. The child entity (the instance of InformationExchange-Requirement) in the relationship must be entered as the *object* in the ObjectVersionAssociation table.

It should also be noted that all relationships of this type, i.e., the equivalent of all the foreign keys modeled in CADM v1.03, have the categoryCode = 999.[5] The meaning of the relationship is stated via the relationTypeCode. The code E102-R-E234 stands for "*is used to transport.*"[6]

At present the values of the relationTypeCode in ObjectVersionAssociation are those defined in CADM v1.03 with the augmentation required to support the net-centric requirements of DoDAF v1.5.

4.4.5 Expressing Associative Entities in CADM v1.5

The second case of relationships involving foreign keys is a generalization of the first case discussed in the preceding section. The main difference is that the semantics of the relationship are augmented through additional attribution. For example, the relationship may have a begin and an end date; it may also have a textual description of the rationale for establishing such a relationship, etc. An example of such a case in CADM v1.5 is the relationship between organizations and mission areas.

In CADM v1.5, all the entities that serve the purpose of relating instances of one type of Object to another type of Object are not modeled explicitly, i.e., there are no entities corresponding to the associative entities that existed in CADM v1.03. Instead, there is a single data structure, namely, ObjectByReference, which allows the instantiation of this type of entity.

Therefore, to express the construct A—related to—AB—related to—B one needs to establish two entries in the ObjectVersionAssociation table, one for the A—related to—AB part and another for the B—related to—AB half. In addition, both entries must use the instance of ObjectByReference corresponding to AB as their *object*. The instance tables below show how one handles the case of an organization supporting a mission area.

As before one must begin by creating all the instances of Object required, namely, the instances of Organization and MissionArea, the two instances of ObjectVersionAssociation that handle the relationships, and an instance of ObjectByReference for the augmented characterization of the association, that, in CADM v1.03, was the entity *OrganizationMissionArea*.

---

[5] In addition to the value '999' the categoryCode can also take the value '997' which stands for the generic association of "OTHER". This value can be used to state that there is a linkage between instances of ObjectVersion but without stating what the linkage means. See Section 4.4.7 below on the use of this type of association for the creation of structures and lists in CADM 1.5.

[6] There is a difference between CADM 1.03 and CADM 1.5 regarding the cardinality of this type of relationships. In the former, instances of the child entity (e.g., InformationExchangeRequirement) could be related only to one instance of the parent entity (e.g., CommunicationMedium). In CADM 1.5 instances of the child entity can be related to more than one instance of the parent entity. In other words, they are implicitly converted into many-to-many relationships. If strict conformance to the original semantics is needed it must be implemented at the application level.

Object

objectIdentifier	pointerCode
415	E312[MissionArea]
416	E432[Organization]
417	E678[OVA]
418	E678[OVA]
419	E679[OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
415	1	Future Land Combat	4 [ArchElem]
416	1	10th Infantry Division	4 [ArchElem]
417	1	10th ID-FLC Support	3 [OVA]
418	1	FLC Supported by 10th ID	3 [OVA]
419	1	OrgMA-419	5 [OBR]

Once these records are created, one can relate the instances corresponding to the Organization and the MissionArea to the instance of ObjectByReference corresponding to *OrganizationMissionArea*.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
417	1	415	1	419	1	999	E312-R-E443
418	1	416	1	419	1	999	E432-R-E443

The relationTypeCode values used have the following meanings:

E312-R-E443 = is supported by
E432-R-E443 = supports

Lastly, one needs to instantiate the ObjectByReference corresponding to *OrganizationMissionArea* so that the augmented attribution for this relationship can be expressed. The tables below show the entries in ObjectByReference and ObjectByReferenceCharacterization.

ObjectByReference

*Identifier	*Index	categoryCode
419	1	E443[OrganizationMissionArea]

ObjectByReferenceCharacterization

*Identifier	OBR Identifier	OBR Index	OBRC categoryCode	OBRC valueText
665	419	1	E443.A01	1[PRIMARY]
667	419	1	E443.A02	SECDEF Memo 9876-A
669	419	1	E443.A03	20070901000000.000
671	419	1	E443.A04	1[HL OP READINESS]

The categoryCode values used have the following meanings:

E443.A01 = PriorityCode
E443.A02 = ReasonText
E443.A03 = RequiredReadinessCalendarDatetime
E443.A04 = RequiredReadinessCode

Reading these tables, one can see that, for the link between the instance of Organization corresponding to the 10th Infantry Division and the instance of MissionArea corresponding to "Future Land Combat," the priority is ranked as "primary," the purpose to support this mission area is provided in the SECDEF Memo 9876-A, and that the ability to support this mission area, at a high level of operational readiness, is expected to be in place by September of 2007.

Note that in addition to linking the two parent entities, the associative entities can be both the source as well as the recipient of additional relationships. In CADM v1.03, one could establish a relationship between Guidance and the associative entity *OrganizationMissionArea*. In CADM v1.5, this is also possible, as it would be simply another instance of ObjectVersionAssociation, and the instantiation would follow the same pattern as the one discussed in Section 4.4.4 above.

4.4.6 Expressing Double Associative Relationships in CADM v1.5

The third case of relationships involving foreign keys is a specialization of the second case discussed in the preceding section. Whereas in the generic case one has A—related to—AB—related to—B, in the specialization both parents are the same (A—related to—A-Assoc—related to—A) and the roles of the two instances of the parent entity are also fixed, i.e., they are either parent-child, or ordinate-subordinate.[7]

In CADM v1.5, the various names given to the role of the parent entity have been normalized to the role of *subject* in the entries of the ObjectVersionAssociation table, and, similarly, the role names of the child entity now all map to the role of *object*. As a result, there is no need to have two records in ObjectVersionAssociation to express relationships of the type A—related to—A-Assoc—related to—A.

For the reasons given above, double associative entities are treated differently from the regular associative entities. Specifically, the categoryCode in ObjectVersionAssociation has a value that corresponds to the name of the original CADM v1.03 double associative entity, and the relationTypeCode is always set to NULL.[8]

Where the double associative entity has augmented characterization, this is captured via ObjectVersionAssociationCharacterization, which works in the same way as the ObjectByReferenceCharacterization discussed in the preceding section.

The following instance tables show a notional case for how to relate an instance of Organization to another instance of Organization, which in CADM v1.03 corresponded to entries in the table *OrganizationAssociation*.[9]

As before, one must start by creating the entries in Object and ObjectVersion.

---

[7] Double-associative entities of the type A-Assoc are essential for the description of taxonomic and hierarchical decompositions. In CADM 1.03 there were 49 such entities, ranging from *OrganizationAssociation*, *FacilityAssociation* to *ProcessActivityAssociation* and *TechnicalInterfaceAssociation*. They are now expressible through ObjectVersionAssociation.

[8] Currently, the categoryCode values reflect the CADM 1.03 set of double associative entities. The suggested use of NULL is only an indication that there should be no entry in the relationTypeCode when expressing double associations in CADM 1.5 through ObjectVersionAssociation.

[9] Throughout this document the following convention is used: entities which are expressed in CADM 1.5 through ObjectByReference or ObjectVersionAssociation are shown in italics. Where they are identical to CADM 1.03 entities the text will normally say so. Entities that are explicitly modeled in CADM 1.5 are shown in bold face.

Object

objectIdentifier	pointerCode
815	E432[Organization]
816	E432[Organization]
817	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
815	1	1st Army Logistics BN	1 [OI]
816	1	82nd Airborne Division	1 [OI]
817	1	TACON-Link 01	3 [OVA]

The linkage between the two instances of Organization is done through ObjectVersionAssociation as follows:

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	categoryCode	relationType Code
817	1	816	1	815	1	E436 [OrgAssoc]	NULL

Finally, the augmented semantics of relationships among instances of Organization is expressed by instantiating the respective entries in the ObjectVersionAssociationCharacterization table as shown below.

ObjectVersionAssociationCharacterization

*Identifier	OVA Identifier	OVA Index	OVAC categoryCode	OVAC valueText
965	817	1	E436.A01	2[OPS SPECIFIC]
967	817	1	E436.A03	20050101000000.000
969	817	1	E436.A04	20081231000000.000
971	817	1	E436.A05	2[NO EFFECT]
973	817	1	E436.A06	10[LOG SERVICES]

The categoryCode values used have the following meanings:

E436.A01 = ConfigurationCategoryCode
E436.A03 = EffectiveCalendarDatetime
E436.A04 = EndCalendarDatetime
E436.A05 = ReinforcementCategoryCode
E436.A06 = TypeCode

Reading these tables, one can see that, for the link between the instance of Organization corresponding to the 82nd ABN Division and the instance of Organization corresponding to 1st Army Logistics Bn, the configuration is listed as "operations specific" (as opposed to a baseline or garrison configuration), the begin and end dates of the association are 01 JAN 05 and 31 DEC 08, respectively, there is no augmentation effect caused by this relationship, and the type of relationship is one in which the subject (1st Army Logistics Bn) *provides logistics services to* the object (82nd ABN Division).

As with all the other coded domains discussed before, the set of valid values currently contained in CADM v1.5 is identical to the one present in CADM v1.03, plus the additional values identified in DoDAF v1.5 for supporting NCO.

4.4.7 Disambiguation of ObjectVersionAssociation Instances in CADM v1.5

The preceding sections have discussed the way in which the new set of CADM v1.5 superstructure entities enable the capture of instances of architecture-relevant data and the relations among them.

As shown in CADM v1.5, once the instances of Object and ObjectVersion are created, the bulk of the work is done through the ObjectVersionAssociation entity. Storing large numbers of DoDAF architecture artifacts in a CADM v1.5-conformant repository will result in a very large set of ObjectVersionAssociation instances.

Traversing this table every time one needs or wants to extract information about the pairs associated or the characteristics of their associations can lead to time-consuming and complex queries.

CADM v1.5 contains a set of entities designed to create data subsets of associations contained in the ObjectVersionAssociation table so that they can be disambiguated and more efficiently retrieved. The first entity is ObjectVersionStructure. This entity can link to a given instance of ObjectVersion one or more sets of data that may have the nature of either tree-like, hierarchical decompositions or just simple lists.

Each instance of ObjectVersionStructure is uniquely identified and can be named to provide a means for separating it from any other list or structure related to the instance of ObjectVersion.

The actual composition of the structure or list is done via the entity ObjectVersionStructureDetail. As shown in Figure 4-8, this entity simply collects the pertinent associations that one wishes to document for the specific instance of ObjectVersionStructure.

The tables below show an example of how this part of the CADM v1.5 can be used. This case involves a hierarchical decomposition as it may exist, for example, in an OV-4 diagram having the generic form shown in **Figure 4-9** below.

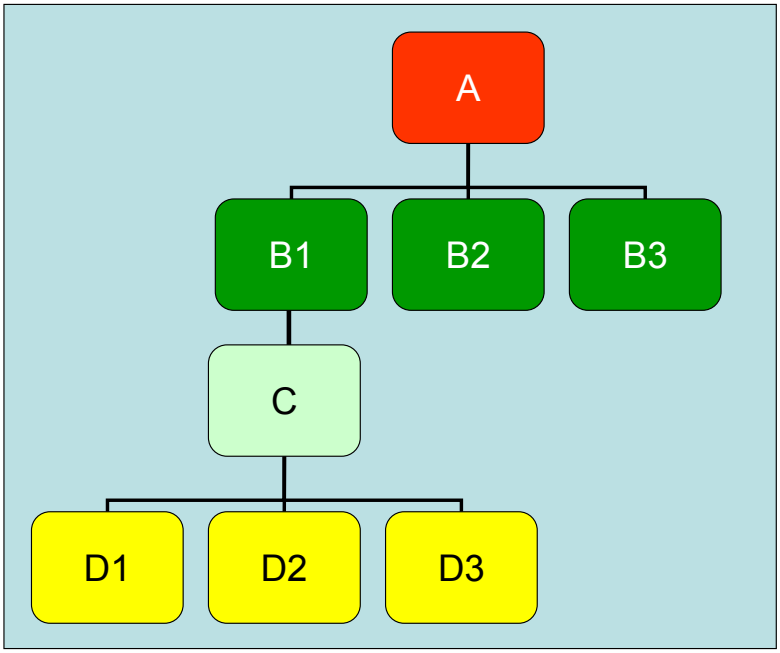


Figure 4-9: Notional Example of Organization associations for an OV-4 type of Architecture Product

As before, we start with the creation of the required instances in Object and ObjectVersion and build the entries in the ObjectVersionAssociation table.

Object

objectIdentifier	pointerCode
515[A]	E432 [Organization]
516[B1]	E432 [Organization]
517[B2]	E432 [Organization]
518[B3]	E432 [Organization]
519[C]	E432 [Organization]
520[D1]	E432 [Organization]
521[D2]	E432 [Organization]
522[D3]	E432 [Organization]
857	E678 [OVA]
858	E678 [OVA]
859	E678 [OVA]
860	E678 [OVA]
861	E678 [OVA]
862	E678 [OVA]
863	E678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
515	1	A	1 [OI]
516	1	B1	1 [OI]
517	1	B2	1 [OI]
518	1	B3	1 [OI]
519	1	C	1 [OI]
520	1	D1	1 [OI]
521	1	D2	1 [OI]
522	1	D3	1 [OI]
857	1	A to B1 link	3 [OVA]
858	1	A to B2 link	3 [OVA]
859	1	A to B3 link	3 [OVA]
860	1	B1 to C link	3 [OVA]
861	1	C to D1 link	3 [OVA]
862	1	C to D2 link	3 [OVA]
863	1	C to D3 link	3 [OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	categoryCode	relationType Code
857	1	515[A]	1	516[B1]	1	E436[OrgAssoc]	NULL
858	1	515[A]	1	517[B2]	1	E436[OrgAssoc]	NULL
859	1	515[A]	1	518[B3]	1	E436[OrgAssoc]	NULL
860	1	516[B1]	1	519[C]	1	E436[OrgAssoc]	NULL
861	1	519[C]	1	520[D1]	1	E436[OrgAssoc]	NULL
862	1	519[C]	1	520[D2]	1	E436[OrgAssoc]	NULL
863	1	519[C]	1	520[D3]	1	E436[OrgAssoc]	NULL

The next step is to create the instance of ObjectVersionStructure that will collect one or more of the decompositions or lists that one wants to assign to a given instance of ObjectVersion. In this example, the ObjectVersionStructure is for the instance of ObjectVersion corresponding to the organization A in Figure 4-9 above.

ObjectVersionStructure

*Identifier	OV Identifier	OV Index	Name	categoryCode
157	515[A]	1	Draft OV-4 High Level	2[Structure]

After the creation of the ObjectVersionStructure, one can specify the content by populating the ObjectVersionStructureDetail table as shown below.

ObjectVersionStructureDetail

*Identifier	OVA Identifier	OVA Index	OVS Identifier
257	857	1	157
258	858	1	157
259	859	1	157
260	860	1	157
261	861	1	157
262	862	1	157
263	863	1	157

Inspection of the table indicates that for the structure with ID = 157 there are 7 *OrganizationAssociations* (specified in the ObjectVersionAssociation table) linked to it. Going back one level, one can see this structure is the one declared for the instance of ObjectVersion corresponding to organization A, the root of the notional OV-4 shown in Figure 4-9 above. Use of the ObjectVersionStructure and ObjectVersionStructureDetail, therefore, makes it easier to understand the relationships among the objects in the data store, and to write specific queries to retrieve data from the ObjectVersionAssociation table.

If additional relationships are established between organization A and other organizations, the structure created in the example above would allow the user to select those with the specific meaning of being part of a high level draft OV-4, as opposed to being part of some other type of organization association specified for organization A.

As mentioned in footnote 7 of Section 4.4.4 above, one can set the categoryCode = '997' in order to capture a generic type of relationship ('OTHER') in ObjectVersionAssociation. This kind of relationship simply states that there is a linkage between the instances without further characterization.

The following uses may be of interest:

1) In repository management where one may want to keep track of all the records that make up an entire architecture so that it can be efficiently retrieved from the repository as a single data set.

2) To create lists. This may be the case when an organization is interested in tracking a bundle of Object instances either for administrative or other purposes.

3) To create particular information exchange artifacts tailored to satisfy specific types of user information requests.

4) To perform architecture analysis. The generic relationship code can be used to create associations that reflect a particular perspective on the architecture data. For example, one can create an architecture view that presents the content of a given architecture from the perspective of the UJTL taxonomy by linking all the organizations, systems, SOA services, etc., that are connected to the UJTL tasks specified in that architecture.

4.5 SUMMARY

The CADM v1.5 superstructure discussed in the preceding section supports all the kinds of relationships specified in the IDEF1X notation that result in a foreign key. **Figure 4-10** shows the graphical representation for these relationships that are now handled via the ObjectVersionAssociation entity in CADM v1.5.

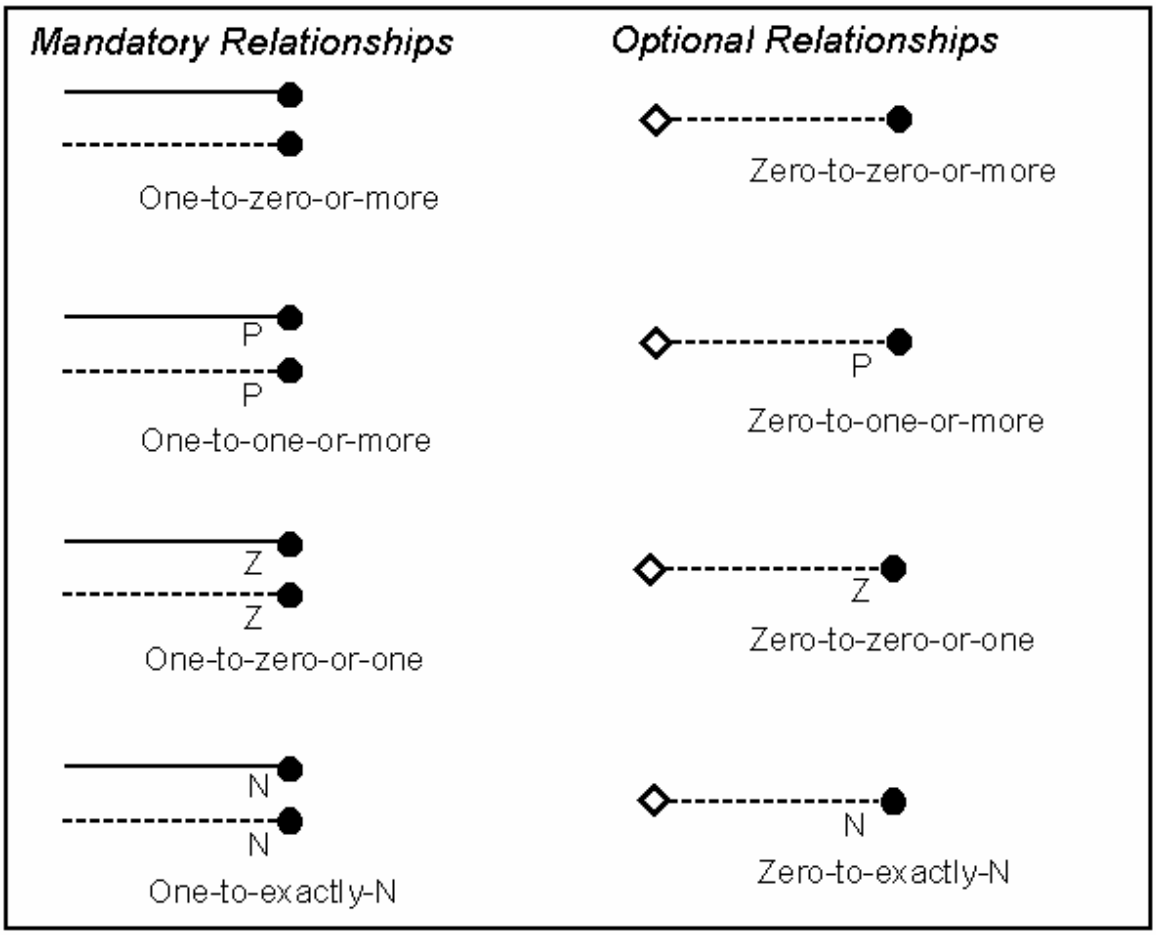


Figure 4-10: Summary Depiction of IDEF1X Notation for Relationships that Result in a Foreign Key

The following things should be noted regarding the modeling approach employed in CADM v1.5 when dealing with these kinds of relationships.

1) The cardinality of the relationships is not explicitly stated in CADM v1.5. Because relationships are instantiated only when there is a need to do so, CADM v1.5 does not restrict the number of relationships that a given instance of Object

can have. If an architecture product does not need to establish a relationship, there is no entry in the ObjectVersionAssociation table. In contrast, CADM v1.03 contains additional columns for all the foreign keys in each one of the tables that may potentially instantiate such a relationship.

2) As noted in the sections above, some relationships in CADM v1.03 restricted the number of instances that could be linked to a given record. Thus, for example, in CADM v1.03, one can only assign one instance of Guideline to a given record in *OrganizationMissionArea*. In CADM v1.5, it is possible to link more than one instance of Guideline to a record of *OrganizationMissionArea*. The consequence of this is that whereas every data set created in conformance to the CADM v1.03 specifications can be mapped to the pertinent structures in CADM v1.05, a data set that does not impose the same restrictions may not be faithfully translated to a CADM v1.03-conformant implementation. In other words, the *backward compatibility* between CADM v1.03 and CADM v1.5 is assured only in one direction, namely from the older specification to the new one. A data set built in CADM v1.5 that does not impose the same cardinality restrictions cannot be expressed using CADM v1.03 structures.

3) Finally, there is no differentiation in CADM v1.5 between so-called "*identifying relationships*" (a.k.a. *mandatory relationships*, or *no nulls allowed*) and "*non-identifying relationships*" (a.k.a. *optional relationships*, or *nulls allowed*). Again, if there is no need to create the relationship, there will be no entry in the ObjectVersionAssociation table. Or, put in a different way, in CADM v1.5, all relationships are treated as optional. Because the relationships are not part of the model structure but are modeled as data entries in the ObjectVersionAssociation table, the model does not enforce any creation of a relationship. If a relationship must exist, its creation has to be enforced at the application level.

An additional type of relationship specified in IDEF1X is the subtype relationship. This type of relationship is similar to the notion of a generalization in object oriented modeling languages such as Unified Modeling Language (UML). **Figure 4-11** shows the corresponding graphical representation of the two kinds of subtype notation that exist in IDEF 1X.

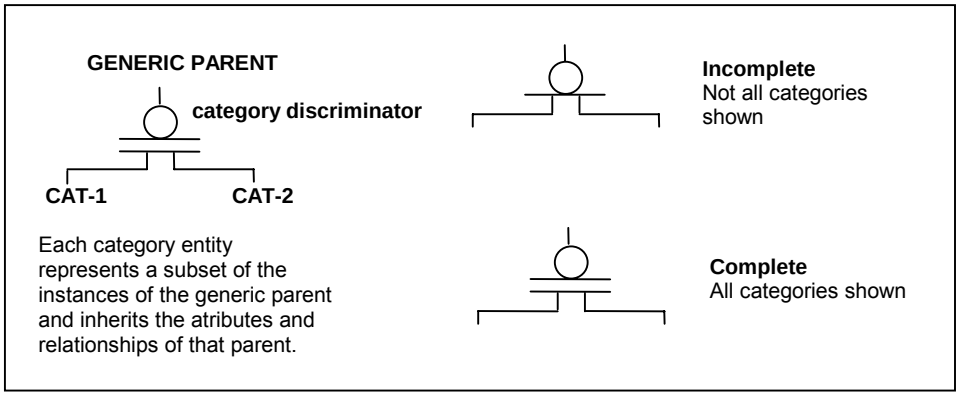


Figure 4-11: Summary Depiction of IDEF1X Notation for Relationships that Result in a Foreign Key

The following things should be noted regarding the modeling approach employed in CADM v1.5 when dealing with subtype relationships.

1) The subtype notation in IDEF1X is purely a logical notation. From the point of view of the key structure, it is equivalent to an identifying Z-relationship, with the added constraint that the subtypes must be mutually exclusive. The latter constraint is only enforceable at the application level. In other words, without additional code, an INSERT statement relates the same instance of entity A to its subtype B1, and its subtype B2 will be executed without error by most, if not all, RDBMS engines.

2) In CADM v1.5, subtype relationships that are not explicitly contained in the model (i.e., where the subtype is an instance of ObjectByReference) are handled in the same way as any of the other relationships shown in Figure 4-10 above. In other words, to establish a subtype relationship, one creates the entries in Object and ObjectVersion for the supertype and the subtype as well as the instance for ObjectVersionAssociation. The supertype always has the role of *subject* and the subtype the role of *object*. The categoryCode must be set equal to '999,' and the relationTypeCode has the value for the specific subtyping relationship. The notation employed differs from the other codes shown in the preceding sections, whereas a foreign key relationship has "R" as part of the code (e.g., E312-R-E443), a subtyping relationship has an "S" (e.g., E047-S-E043).

4.5.1 Mapping Business Rules

All the mapping Business Rules for expressing CADM v1.03 data sets via the CADM v1.5 is provided in the document "MappingRules CADM 103-CADM 15.html," distributed in electronic form with the DoDAF v1.5 volumes. The document is also available at the DARS site (https://dars1.army.mil/IER/index.jsp).

The applicable coded domains are also provided in the "CADM v1.5 Domain Specification.doc" document.

4.6 CADM V1.5 SUPPORT FOR DODAF PRODUCTS

The richness and expressiveness of an information model is a function of two components, (a) the number and kind of relationships that are defined among the entities of the information model and (b) the robustness of the entity attribution.

The preceding section discussed the first component. That section showed how all the kinds of relationships that existed in CADM v1.03 continue to be supported in CADM v1.5. That section also showed that CADM v1.5 provides for additional mechanisms to disambiguate data sets, as well as means to create ad hoc relationships that go beyond what CADM v1.03 specified. Four potential uses of the latter were also presented therein.

This section addresses the second component, namely, the entity attribution specified in CADM v1.5. The description will be done in the context of the architecture products. DoDAF v1.5 Volume II contains the description of all the entities necessary to capture the data underlying each of the DoDAF products. The following section will follow the same approach. The main difference is that now the attribution will be explicitly discussed, and instance table examples will be provided.

4.6.1 CADM v1.5 Support for Overview and Summary Information (AV-1)

4.6.1.1 Product Definition

As stated in DoDAF v1.5 Volume II, the purpose of AV-1 is to provide sufficient textual information to enable a reader to select one architecture from among many to read in more detail executive-level summary information in a consistent form that allows quick reference and comparison among architectures. AV-1 includes assumptions, constraints, and limitations that may affect high-level decision processes involving the architecture.

AV-1 serves two additional purposes. In the initial phases of architecture development, it serves as a planning guide. Upon completion of an architecture, AV-1 provides summary textual information concerning the architecture.

The information pertinent to an architecture is captured in CADM v1.5 in the entity Architecture, and the DoDAF products contained in it are represented as instances of Document. Both these entities are subtypes of ArchitectureElement. **Figure 4-12** shows the specification of these data structures.

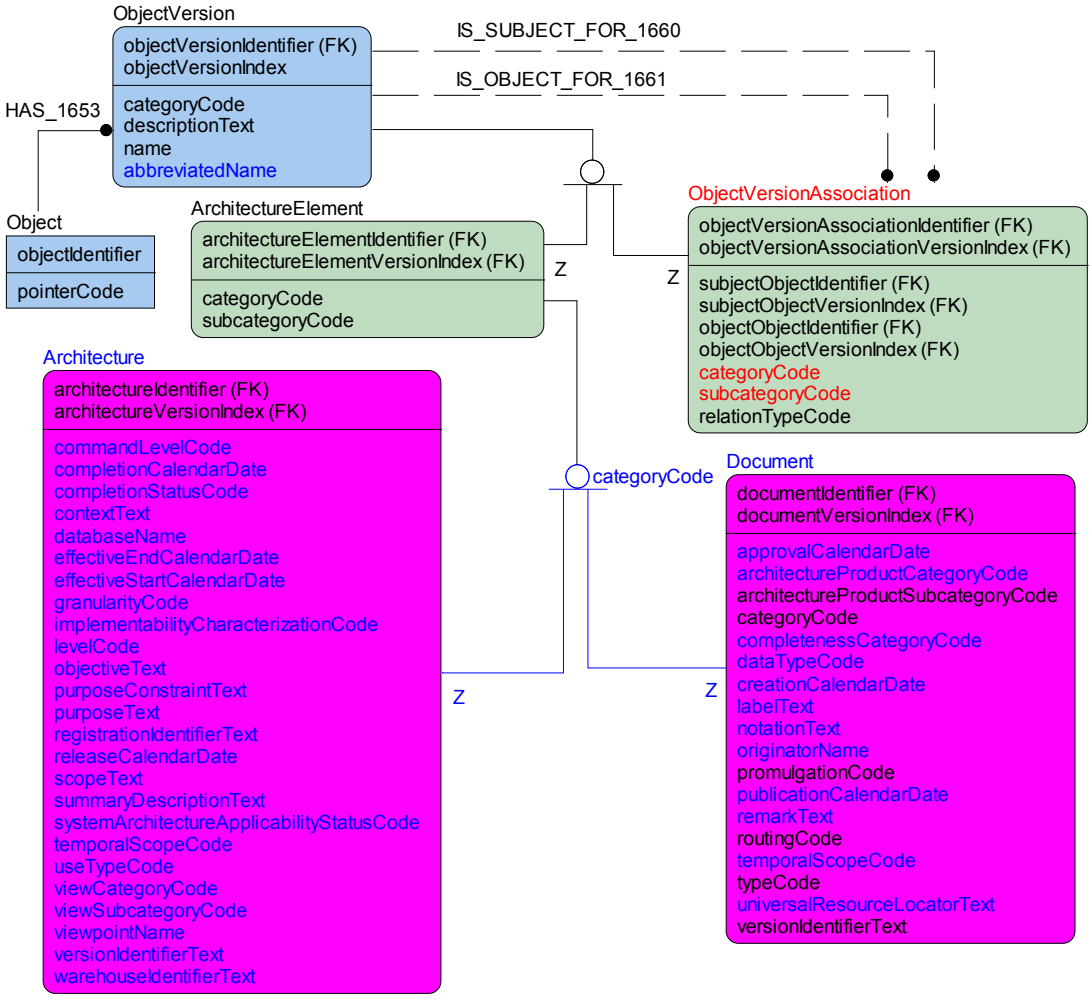


Figure 4-12: Attribute-Level Depiction of Document and Architecture Data Structures in CADM v1.5

4.6.1.2 High-Level Description

Figure 4-13 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an AV-1.

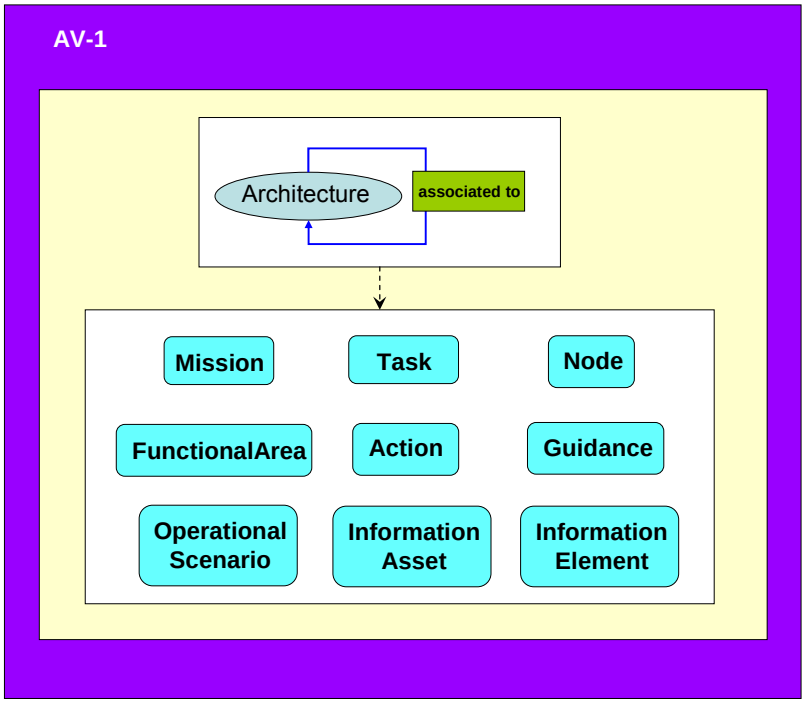


Figure 4-13: High-Level Depiction of CADM v1.5 Data Structures for AV-1 Representation

Each AV-1 is specified by use of (1) Architecture, its association to other types of architectures (operational, systems and services, and technical) using ObjectVersionAssociation with categoryCode = E041 [ARCHITECTURE-ASSOCIATION] and instances of ObjectByReference corresponding to the CADM v1.03 entity *ArchitectureFinding* (together with subtypes for issues, constraints, impacts, and recommendations); (2) the linkages to Mission, OperationalScenario, Action, and System; together with (3) associations that relate a specific Architecture to other data structures such as Agreement, Document, FunctionalArea, Node, Action, Task, and InformationTechnologyRequirement.[10]

4.6.1.3 CADM v1.5 Instantiation

Figure 4-14 shows the template for AV-1 content. An example of the instantiation of a record in the Architecture table that captures AV-1 data along the lines of the template is shown below the figure.

[10] For a complete list of all the possible associations supported both in CADM 1.03 and CADM 1.5 see the "MappingRules CADM 103-CADM 15.html" document.

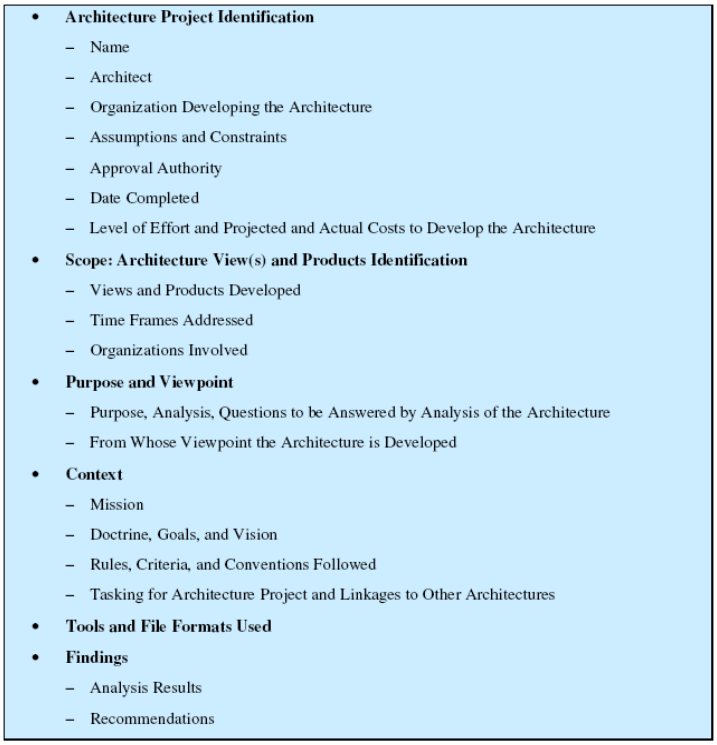


Figure 4-14: Summary Depiction of AV-1 Content

Object

objectIdentifier	pointerCode
20000021	E038[Architecture]

ObjectVersion

*Identifier	*Index	name	description Text	categoryCode
20000021	1	Combat Identification Architecture	Includes Phase I (Air-to-Surface and Surface-to-Surface) and Phase II (Air-to-Air and Surface-to-Air) products for Joint Staff J8, OUSD(AT&L), and ASD(NII)C3.	4[ArchElem]

ArchitectureElement

*Identifier	*Index	categoryCode	subcategoryCode
20000021	1	E038	NULL

Architecture

*Identifier	*Index	completion CalendarDate	summary DescriptionText	objective Text
20000021	1	20040213	A significant number (34%) of the 2007 As-Is critical gaps can be addressed by accelerated deployment of existing systems; may achieve FOC by 2007 if begun immediately.	Provide (1) a future joint CID vision and operational concept; (2) integrated operational and systems architectural views in accordance with the DoD AF; and (3) a capabilities roadmap and investment strategy (CRIS) for future CID capabilities.

Architecture (Cont'd)

scopeText	contextText	purposeText	purposeConstraintText
The integrated CID architecture addresses all four operating environments identified in the CID CRD—surface-to-surface (S-S), air-to-surface (A-S), air-to-air (A-A), and surface-to-air (S-A). First priority was given to S-S and A-S.	Description of an analytic methodology specifically developed to use operational and systems architectures as a basis for identifying capability gaps and shortfalls and potential system solutions to address critical capability needs. Includes examples of …	1. Identification of "high-urgency" capability gaps: (a) by individual platform and platform class; and (b) by operating environment/ mission area. 2. Identification of improvement options to directly address "high-urgency" capability gaps.	The integrated CID architecture addresses all four operating environments identified in the CID Capstone Requirements Document (CRD)—surface-to-surface (S-S), air-to-surface (A-S), air-to-air (A-A), and surface-to-air (S-A). First priority was given to S-S and A-S since historically most fratricide incidents have occurred in these two environments. IDA completed the development and coordination of S-S and A-S architecture and CRIS products in 2002. Additionally, IDA identified high-priority, high-payoff CID technology and system investments to address critical CID capability shortfalls for those two operating environments. IDA completed the development of CID architecture and CRIS product for the A-A and S-A operating environments in 2003.

Architecture (Cont'd)

temporal ScopeCode	view TypeCode	levelCode	completion StatusCode	useTypeCode	view PointName	granularity Code
1 (As Is)	1 (Operational)	1 (Enterprise)	2 (Draft)	9 (Other)	Designer	3 (Operational)

Architecture (Cont'd)

effective Start Calendar Date	effective End Calendar Date	implementability Characterization Code	command LevelCode	Version Identifier Text	Release CalendarDate
20011228	20040220	R (Real)	03 (Military Dept.)	Version 2	20040220

Architecture (Cont'd)

Warehouse IdentifierText	database| Name
2001 (DARS, Planned)	CADM Example Architecture Database

An example of the instantiation of Document describing architecture products related to the instance of Architecture shown in the preceding tables is shown below.

Document

DOC_ID	DOC_ ABBRV_ NM	DOC_NM	DOC_ ARCHPROD_ TY_CD	DOC_ PUB_DT
501	AV-1 CID	AV-1 CID Overview and Summary	98	20031119
502	AV-2 CID	AV-2 CID Integrated Dictionary	7	20031119
503	OV-2 CID	OV-2 CID Operational Node Connectivity Diagram	24	20031119
504	OV-3 CID	OV-3 CID Operational Information Exchange Matrix	16	20031119
506	OV-5 CID	OV-5 CID Activity Model	1	20031119
507	OV-7 CID	OV-7 CID Logical Data Model	8	20031119
508	SV-1 CID	SV-1 CID Systems Interface Description	39	20031119
509	SV-2 CID	SV-2 CID Systems Connectivity Diagram	35	20031119
510	SV-3 CID	SV-3 CID Systems-Systems Matrix	40	20031119
512	SV-5 CID	SV-5 CID Operational Activity to Systems Function Traceability Matrix	38	20031119
514	SV-7 CID	SV-7 CID Systems Performance Parameters Matrix	25	20031119
516	SV-9 CID	SV-9 CID Systems Technology Forecast	41	20031119
517	SV-11 CID	SV-11 CID Physical Schema	26	20031119
518	TV-1 CID	TV-1 CID Technical Standards Profile	42	20031119
519	TV-2 CID	TV-2 CID Technical Standards Forecast	34	20031119

To express the fact that the instance of Architecture is described by the architecture products indicated in the preceding Document, table one needs to create the appropriate entries in the ObjectVersionAssociation using instances of ObjectByReference corresponding to the CADM v1.03 entity *ArchitectureDocument* to connect the products to the architecture. For simplicity only, a pair of those relationships are shown since the method is exactly the same for each of the rows in the Document table.

Object

objectIdentifier	pointerCode
20000021	E038[Architecture]
501	E148[Document]
502	E148[Document]
503	E148[Document]
117	E679[OBR]
118	E679[OBR]
119	E679[OBR]
2522	E678[OVA]
2523	E678[OVA]
2524	E678[OVA]
2525	E678[OVA]
2526	E678[OVA]
2527	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
20000021	1	Combat Identification Architecture	4[ArchElem]
501	1	AV-1 CID Overview and Summary	4[ArchElem]
502	1	AV-2 CID Integrated Dictionary	4[ArchElem]
503	1	OV-2 CID Operational Node Connectivity Diagram	4[ArchElem]
117	1	ArchitectureDocument (AV-1 in Combat ID Arch)	5[OBR]
118	1	ArchitectureDocument (AV-2 in Combat ID Arch)	5[OBR]
119	1	ArchitectureDocument (OV-2 in Combat ID Arch)	5[OBR]
2522	1	Architecture is documented by AV-1	3[OVA]
2523	1	AV-1 documents Architecture	3[OVA]
2524	1	Architecture is documented by AV-2	3[OVA]
2525	1	AV-2 documents Architecture	3[OVA]
2526	1	Architecture is documented by OV-2	3[OVA]
2527	1	OV-2 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
117	1	E045[ArchitectureDocument]
118	1	E045[ArchitectureDocument]
119	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
2522	1	20000021	1	117	1	999	E038-R-E045
2523	1	501	1	117	1	999	E148-R-E045
2524	1	20000021	1	118	1	999	E038-R-E045
2525	1	502	1	118	1	999	E148-R-E045
2526	1	20000021	1	119	1	999	E038-R-E045
2527	1	503	1	119	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

As depicted in Figure 4-13 above, the AV-1 also documents the organizations, missions, tasks, actions, etc., that pertain to the architecture. In CADM v1.5, the details of those relationships are expressed through each of the products, which are described in detail in the subsequent sections. Direct links between Architecture and any of those data structures, where defined, can be expressed in the same fashion presented in the previous instance tables through the use of ObjectVersionAssociation.

The ObjectVersionStructure and ObjectVersionStructureDetail (see discussion above) can also be employed to capture links that were not supported in CADM v1.03 directly, but which would facilitate the retrieval of data pertinent to this product.

4.6.1.4 Net-Centric Requirements

The specification of services can be expressed in CADM v1.5 through SoaService, which is linkable to Architecture through ObjectVersionAssociation using instances of ObjectByReference

corresponding to *ArchitectureSoaService*. To do that, one needs to create an instance of *ArchitectureSoaService* and link it in the ObjectVersionAssociation table to (a) SoaService with the relationTypeCode = E682-R-E683 (*is cited for*) and to (b) Architecture with the relationTypeCode = E038-R-E683 (*cites*).

4.6.2 CADM v1.5 Support for Integrated Dictionary (AV-2)

4.6.2.1 Product Definition

As stated in DoDAF v1.5 Volume II, the AV-2 contains definitions of terms used in the given architecture. It consists of textual definitions in the form of a glossary, a repository of architecture data, their taxonomies, and their metadata (i.e., data about architecture data), including metadata for tailored products, associated with the architecture products developed. A type of metadata is the architecture structured data attributes, possibly expressed in the form of a physical schema. In this document, architecture data types are referred to as architecture data elements.

4.6.2.2 High-Level Description

Figure 4-15 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an AV-2.

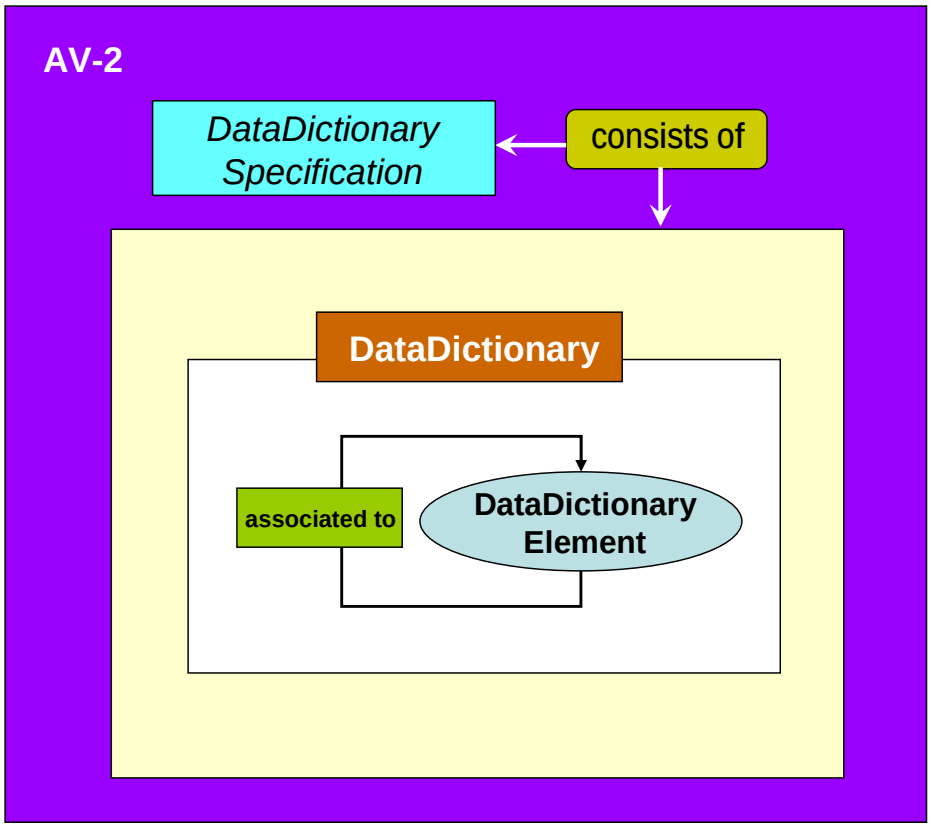


Figure 4-15: High-Level Depiction of CADM v1.5 Data Structures for AV-2 Representation

In CADM v1.5, the DoDAF architecture product AV-2 is an instance of Document with architectureProductCategoryCode = 7 [DATA-DICTIONARY-SPECIFICATION]. Each AV-2 cites a specific instance of DataDictionary. The AV-2 document can be linked to the appropriate

architecture via ObjectVersionAssociation using instances of ObjectByReference corresponding to the CADM v1.03 the associative entity *ArchitectureDocument*.

The actual content of the AV-2 is built by linking it to instances of DataDictionary, a subtype of InformationAsset. The DataDictionary is defined using instances of ObjectByReference corresponding to the CADM v1.03 entities *DataDictionaryElement* and *DataDictionaryElementAssociation* and linking them via ObjectVersionAssociation. These entities provide the details needed for a self-contained Glossary of Terms. Where the AV-2 is considered a database, the schema for the DataDictionary can be specified using the InformationAsset subtype ConceptualDataModel. The entities, attributes, relationships, and other information for the metadata model of the Data Dictionary can be specified in such entities of the CADM as DataEntity, DataAttribute, *DataEntityRelationship*, and *DataDomain*.

4.6.2.3 CADM v1.5 Instantiation

The AV-2 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
91136	E038 [Architecture]
91137	E148 [Document]
91138	E679 [OBR]
91605	E678 [OVA]
91606	E678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
91136	1	Project Charlie-Bravo Architecture	4 [ArchElem]
91137	1	AV-2 Data Dictionary Specification	4 [ArchElem]
91138	1	ArchitectureDocument (AV-2 in Project Charlie-Bravo Architecture [91136]	5[OBR]
91605	1	Architecture is documented by AV-2	3[OVA]
91606	1	AV-2 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
91138	1	E045 [ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
91605	1	91136	1	91138	1	999	E038-R-E045
91606	1	91137	1	91138	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The AV-2 document is linked to instances of DataDictionary, a subtype of InformationAsset. The instance tables for a notional example contaning two instances of DataDictionary are shown below.

Object

objectIdentifier	pointerCode
92136	E123 [DataDictionary]
92137	E123 [DataDictionary]

ObjectVersion

*Identifier	*Index	name	categoryCode
92136	1	AV-2 – Definitions Section	4 [ArchElem]
92137	1	AV-2 – Acronyms and Abbreviations	4 [ArchElem]

ArchitectureElement

*Identifier	*Index	categoryCode
92136	1	33 = INFORMATION-ASSET
92137	1	33 = INFORMATION-ASSET

InformationAsset

*Identifier	*Index	typeCode	versionIdentifierText
92136	1	18 = DATA DICTIONARY	Version 1.0.1
92137	1	18 = DATA DICTIONARY	Version 1.0.1

DataDictionary

*Identifier	*Index	typeCode
92136	1	8 = NOT SPECIFIED
92137	1	8 = NOT SPECIFIED

Each instance of DataDictionary can be related to one or more instances of ObjectByReference corresponding to the CADM v1.03 entity *DataDictionaryElement*. The tables below show a series of instances for the notional example discussed in this section.

Object

objectIdentifier	pointerCode
20000001	E679 [OBR]
20000002	E679 [OBR]
20000003	E679 [OBR]
20000007	E679 [OBR]
20000008	E679 [OBR]
20000009	E679 [OBR]
20000010	E679 [OBR]
20000011	E679 [OBR]
20000012	E679 [OBR]
20000013	E679 [OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
20000001	1	ACCURACY, GEOSPATIAL	5 [OBR]
20000002	1	ACCURACY, KINEMATIC	5 [OBR]
20000003	1	ACTIVITY MODEL	5 [OBR]
20000007	1	A/C	5 [OBR]
20000008	1	A-A	5 [OBR]
20000009	1	AAAV	5 [OBR]
20000010	1	AAD	5 [OBR]
20000011	1	AADC	5 [OBR]
20000012	1	AAMDC	5 [OBR]
20000013	1	AAP	5 [OBR]

ObjectByReference

*Identifier	*Index	categoryCode
20000001	1	E124 [DataDictionaryElement]
20000002	1	E124 [DataDictionaryElement]
20000003	1	E124 [DataDictionaryElement]
20000007	1	E124 [DataDictionaryElement]
20000008	1	E124 [DataDictionaryElement]
20000009	1	E124 [DataDictionaryElement]
20000010	1	E124 [DataDictionaryElement]
20000011	1	E124 [DataDictionaryElement]
20000012	1	E124 [DataDictionaryElement]
20000013	1	E124 [DataDictionaryElement]

For each instance of ObjectByReference corresponding to the CADM v1.03 entity *DataDictionaryElement*, one can specify its attribution via ObjectByReferenceCharacterization. The tables below show this for the first two instances in the table above.

ObjectByReferenceCharacterization

*Identifier	OBR Identifier	OBR Index	category Code	value Text
45671	20000001	1	E124.A01	1 (Approved)
45672	20000001	1	E124.A02	20010401
45673	20000001	1	E124.A03	CID systems must be sufficiently accurate to precisely correlate characterizations among multiple closely spaced surface targets. Geospatial accuracy will be met if all participants can correctly correlate with network tracks. Thus, if participants are not generating dual designations and not miscorrelating tracks, then they must be meeting the CID requirement for geospatial accuracy.
45674	20000001	1	E124.A04	Text
45676	20000001	1	E124.A06	SIAP SETF Analysis System Engineering Team
45677	20000001	1	E124.A07	For use in CID and JTAMD architectures
45678	20000002	1	E124.A01	1 (Approved)
45679	20000002	1	E124.A02	20010401
45680	20000002	1	E124.A03	CID systems are kinematically accurate when the position and velocity of a track agrees with the position and velocity of the associated object.
45681	20000002	1	E124.A04	Text
45683	20000002	1	E124.A06	SIAP SETF Analysis System Engineering Team
45684	20000002	1	E124.A07	For use in CID and JTAMD architectures

The categoryCode values in the table above correspond to the attribution of the CADM v1.03 entity *DataDictionaryElement*. They have the following meaning:

E124.A01 = ApprovalStatusCode
E124.A02 = ApprovalStatusCalendarDate
E124.A03 = DefinitionText
E124.A04 = FormatDescriptionText
E124.A06 = SourceName
E124.A07 = UsageDescriptionText

The linkage between the AV-2 document and the instances of DataDictionary is done through ObjectVersionAssociation. The table below show the instantiation for the example shown in this section (the instances of Object and ObjectVersion are not shown).

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
256801	1	92136	1	91137	1	999	E123-R-E126
256802	1	92137	1	91137	1	999	E123-R-E126

E123-R-E126 = [DataDictionary] is cited for [DataDictionarySpecification – AV-2]

4.6.2.4 Net-Centric Requirements

The specification of services and family of services can be expressed in CADM v1.5 through SoaService. Instances of this entity can be linked to each other through ObjectVersionAssociation with categoryCode set to E684 [*SoaServiceAssociation*]. This allows the expression of decomposition of services as well as other relationships such as "*supports*," "*is alternate for*," etc.

4.6.3 CADM v1.5 Support for High-Level Operational Concept Graphic (OV-1)

4.6.3.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-1 describes a mission and highlights main operational nodes (see OV-2 definition), as well as interesting or unique aspects of operations. It provides a description of the interactions between the subject architecture and its environment, and between the architecture and external systems. A textual description accompanying the graphic is crucial. Graphics alone are not sufficient for capturing the necessary architecture data.

4.6.3.2 High-Level Description

Figure 4-16 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-1.

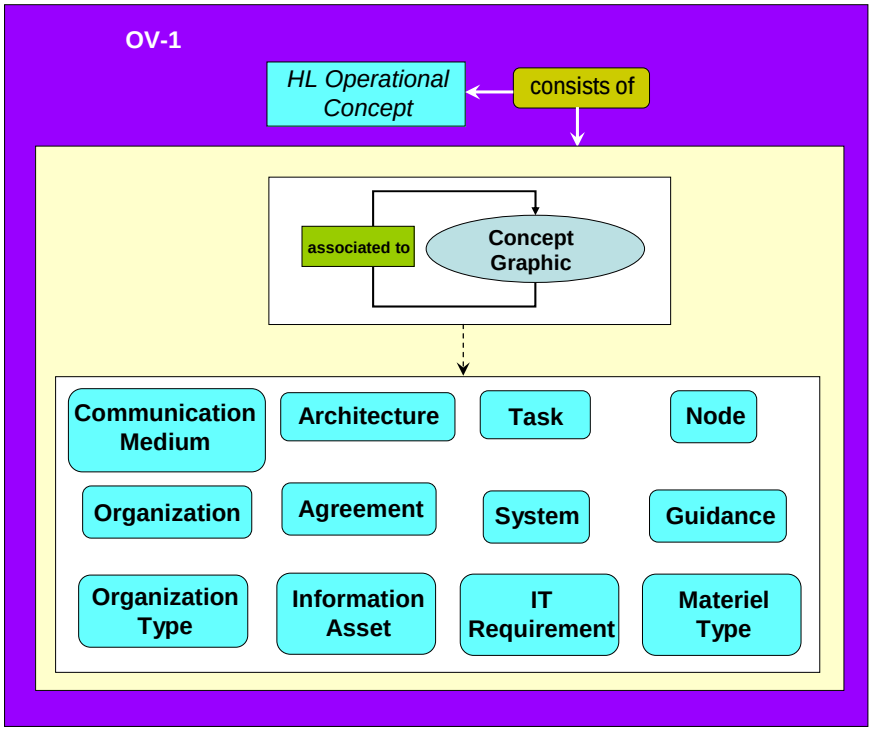


Figure 4-16: High-Level Depiction of CADM v1.5 Data Structures for OV-1 Representation

In CADM v1.5, the DoDAF architecture product OV-1 is an instance of Document with architectureProductCategoryCode = 6 [CONCEPT-GRAPHIC]. Where an OV-1 is made up of discrete components defined in their own right, the overall OV-1 can be built through *DocumentAssociation* (using ObjectVersionAssociation). Each instance of OV-1 or a component can be directly related in the CADM to such entities as Agreement, all other architecture products, Architecture, Guidance (including subtypes of InformationTechnologyRequirement for needlines and information exchange contents), InformationAsset, MaterielType, Organization, OrganizationType, Node, and System through relationships with the parent entity Document. Indirect relationships to Mission, Task, CommunicationMedium, and *PlatformElement* (through ObjectByReference corresponding to the CADM v1.03 entity *SystemElement*) can also be recorded in the CADM. See **Figure 4-17**, USCENTCOM deep operations in the joint operations area.

4.6.3.3 CADM v1.5 Instantiation

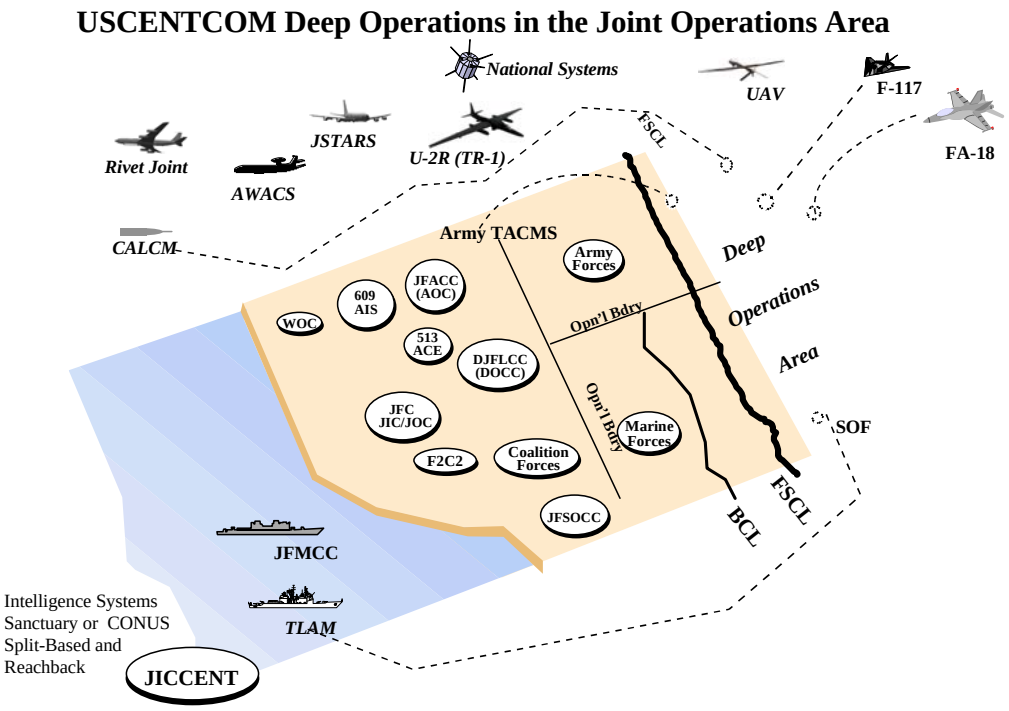


Figure 4-17: USCENTCOM Deep Operations in the Joint Operations Area Example

The OV-1 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
115	E038[Architecture]
116	E148[Document]
117	E679[OBR]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
115	1	Program Architecture C08	4[ArchElem]
116	1	Notional OV-1	4[ArchElem]
117	1	ArchitectureDocument (OV-1 in Program Architecture C08)	5[OBR]
522	1	Architecture is documented by OV-1	3[OVA]
523	1	AV-2 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
117	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	115	1	117	1	999	E038-R-E045
523	1	116	1	117	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The tables below show the instance tables for a notional example using CADM v1.5.

Object

objectIdentifier	pointerCode
116	E148[Document]
117	E148[Document]
215	E679 [OBR]
216	E679 [OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
116	1	OV-1 Electronic Commerce Overview and Summary	4[ArchElem]
117	1	OV-1 CENTCOM Deep Operations in JOA Overview and Summary	4[ArchElem]
215	1	ConceptGraphic for OV-1 [116]	5[OBR]
216		ConceptGraphic for OV-1 [117]	

ObjectByReference

*Identifier	*Index	categoryCode
215	1	E111[ConceptGraphic]
216	1	E111[ConceptGraphic]

Through the use of ObjectByReferenceCharacterization, it is possible to state the specifics of each graphic, for example its physical type, i.e., view graph, bit map, etc.

The relation between the OV-1 and the instances of Agreement is done through ObjectByReference in the usual manner. The instances of OVA in the Object and ObjectVersion tables are not shown.

Object

objectIdentifier	pointerCode
116	E148 [Document]
305	E031 [Agreement]
501	E679 [OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
116	1	Notional OV-1	4[ArchElem]
305	1	OV-1 Agreement	4[ArchElem]
501	1	AgreementDocument(Agreement in OV-1)	5[OBR]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
7701	1	116	1	501	1	999	E031-R-E033
7702	1	305	1	501	1	999	E031-R-E034

The relationTypeCode values used have the following meanings:

E148-R-E033 = specifies
E031-R-E033 = is specified using

4.6.3.4 Net-Centric Requirements

The specification of services at the operational level can be expressed in CADM v1.5 through SoaService, which is linkable to Node through instances of ObjectByReference corresponding to *NodeSoaService*. The applicable codes in the ObjectVersionAssociation table are:

E682-R-E685 = [SoaService] supports the functions of [*NodeSoaService*]

E359-R-E685 = [Node] is supported by [*NodeSoaService*].

4.6.4 CADM v1.5 Support for Operational Node Connectivity Description (OV-2)

4.6.4.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-2 graphically depicts the operational nodes (or organizations) with *needlines* between those nodes that indicate a need to exchange information. The graphic includes internal operational nodes (internal to the architecture) as well as external nodes.

4.6.4.2 High-Level Description

Figure 4-18 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-2.

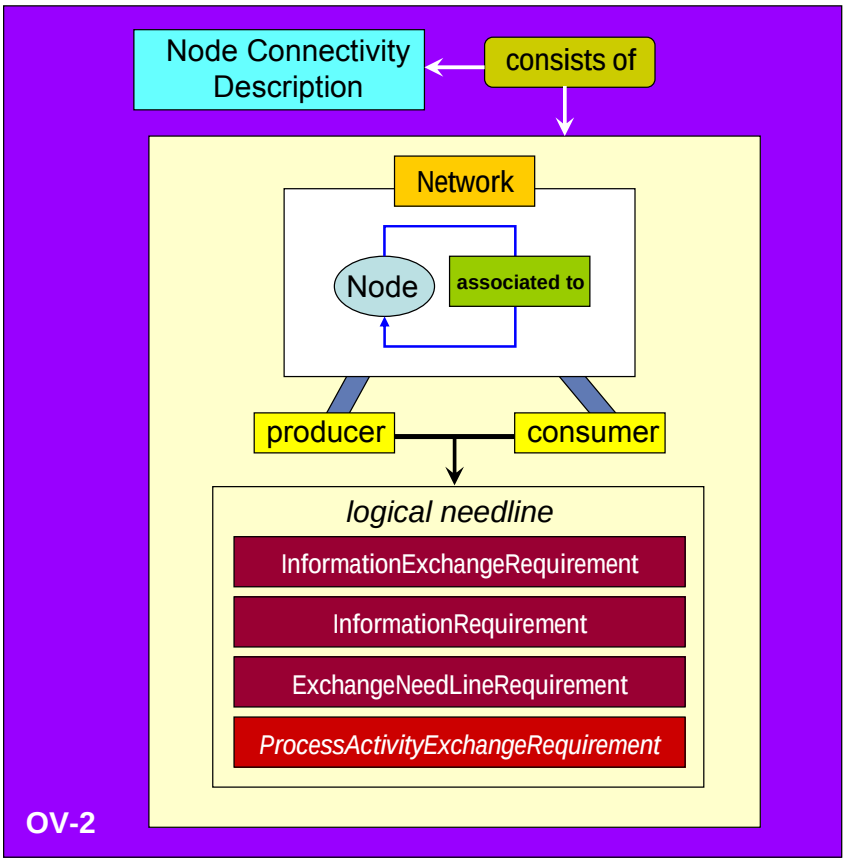


Figure 4-18: High-Level Depiction of CADM v1.5 Data Structures for OV-2 Representation (Notation Independent Style)

In CADM v1.5, the DoDAF architecture product OV-2 as an architecture product is expressed as an instance of Document. This instance can be connected to the appropriate instance of Architecture of which it is part. The instance of Document links to the actual data content of the OV-2 through one or more instances of Network.

The function of these instances of Network is simply to collect the nodes that are part of the OV-2. The nodes, in turn, can be linked to each other to create node associations. As will be shown below, these node associations are related to the instances of InformationExchangeRequirement, the CADM v1.5 entity that serves as the focus for the specification of the logical *needlines*.

InformationTechnologyRequirement has four subtypes, namely, *ProcessActivityExchangeRequirement (PAER)* (modeled via ObjectByReference), InformationExchangeRequirement (IER), ExchangeNeedLineRequirement (ENLR), and InformationRequirement (IR).

In the context of an OV-2, an instance of ObjectByReference corresponding to *ProcessActivityExchangeRequirement* allows the specification of the producer and consumer *activities* with respect to each instance of **IER**. Instances of *PAER* are linked to **IER,** through the relationship "*PAER is cited in **IER.***"

As the description of the OV-2 in DoDAF v1.5 Volume II indicates, *needlines* can be among organizations. The CADM v1.5 data structure **ENLR** can be used to state the organization or organization type that is the source or destination with respect to each **IER**. Instances of **ENLR** are linked to **IER** via the relationship "***ENLR** uses **IER.***"

The characterization of the information content of a logical *needline* is done through the CADM v1.5 data structure **IR**. Instances of **IER** can then be linked to the corresponding **IR** via the relationship "***IR** is used for **IER.***"

The OV-2 product itself (as opposed to its content) is represented in CADM v1.5 as an instance of Document. The linkage of the OV-2 document to its content (as described above) is established via the relationship "*is used to specify*" from Network (which is defined as the specification for the joining of two or more nodes for a specific purpose).

4.6.4.3 CADM v1.5 Instantiation

Figure 4-5 in DoDAF Volume II depicts the template for OV-2 products in a notation neutral form. As shown in Figure 4-5, the *needlines* may flow either in a single direction or both ways between two given nodes. When the *needlines* are intended to be bi-directional, one must instantiate in CADM v1.5 two instances to InformationTechnologyRequirement, as well as the corresponding subtypes to indicate the roles implied by the directionality. Each node may be the site for a number of *activities* according to this template. As mentioned in the preceding subsection, the representation of the OV-2 data content that CADM v1.5 utilizes is based on instances of Network, which in turn may contain as many instances of Node as the OV-2 contains, and as many instances of InformationExchangeRequirement as there are *needlines* in the OV-2.

1. For the OV-2 as a document:

   a) One Document to express the fact that this is a DoDAF product
   b) One Architecture, since products are always viewed as components of architectures
   c) One instance of OBR corresponding to *ArchitectureDocument* to link the two
   d) Two instances of OVA, one for the Document relationship, the other for the Architecture relationship to the OBR above.

Object

objectIdentifier	pointerCode
115	E038[Architecture]
116	E148[Document]
117	E679[OBR]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
115	1	Program Architecture C08	4[ArchElem]
116	1	Notional Node Connectivity Description (Template)	4[ArchElem]
117	1	ArchitectureDocument (OV-2 in Program Architecture C08)	5[OBR]
522	1	Architecture is documented by OV-2	3[OVA]
523	1	OV-2 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
117	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	115	1	117	1	999	E038-R-E045
523	1	116	1	117	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

2. For the Nodes:

   a) One instance of Network, to collect the various node associations
   b) Five instances of Node, three that are internal to the OV-2 and two external to it
   c) Five instances of OBR (NetworkNode) to link the nodes to the network
   d) 10 instances of OVA to relate the network to the nodes (these are always pairwise, one OVA from Network to OBR (NetworkNode) and one OVA from Node to OBR(NetworkNode)
   e) Four instances of OVA (*NodeAssociation*) to handle the associations between the five nodes in the template

Object

objectIdentifier	pointerCode
207	E333[Network]
217	E359[Node]
218	E359[Node]
219	E359[Node]
220	E359[Node]
221	E359[Node]
247	E679[OBR]
248	E679[OBR]
249	E679[OBR]
250	E679[OBR]
251	E679[OBR]]
524	E678[OVA]
525	E678[OVA]
526	E678[OVA]

objectIdentifier	pointerCode
527	E678[OVA]
528	E678[OVA]
529	E678[OVA]
530	E678[OVA]
531	E678[OVA]
532	E678[OVA]
533	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
207	1	OV-2 Level 1 Decomposition	4[ArchElem]
217	1	Node A	4[ArchElem]
218	1	Node B	4[ArchElem]
219	1	Node C	4[ArchElem]
220	1	External Destination L	4[ArchElem]
221	1	External Source M	4[ArchElem]
247	1	NetworkNodeA (Node A in OV-2[115])	5[OBR]
248	1	NetworkNodeB (Node B in OV-2[115])	5[OBR]
249	1	NetworkNodeC (Node C in OV-2[115])	5[OBR]
250	1	NetworkNodeL (External Destination L in OV-2[115])	5[OBR]
251	1	NetworkNodeM (External Source M in OV-2[115])	5[OBR]
524	1	NetworkNodeA[247] is part of Network	3[OVA]
525	1	NodeA[217] is part of NetworkNodeA	3[OVA]
526	1	NetworkNodeB[248] is part of Network	3[OVA]
527	1	NodeB[218] is part of NetworkNodeB	3[OVA]
528	1	NetworkNodeC[249] is part of Network	3[OVA]
529	1	NodeB[219] is part of NetworkNodeC	3[OVA]
530	1	NetworkNodeL[250] is part of Network	3[OVA]
531	1	ExternalDestinaltionL[220] is part of NetworkNodeL	3[OVA]
532	1	NetworkNodeM[251] is part of Network	3[OVA]
533	1	ExternalSourceM[221] is part of NetworkNodeM	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
247	1	E350[NetworkNode]
248	1	E350[NetworkNode]
249	1	E350[NetworkNode]
250	1	E350[NetworkNode]
251	1	E350[NetworkNode]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
524	1	207	1	247	1	999	E333-R-E350
525	1	217	1	247	1	999	E359-R-E350
526	1	207	1	248	1	999	E333-R-E350
527	1	218	1	248	1	999	E359-R-E350

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
528	1	207	1	249	1	999	E333-R-E350
529	1	219	1	249	1	999	E359-R-E350
530	1	207	1	250	1	999	E333-R-E350
531	1	220	1	250	1	999	E359-R-E350
532	1	207	1	251	1	999	E333-R-E350
533	1	221	1	251	1	999	E359-R-E350

The relationTypeCode values used have the following meanings:

E359-R-E350 = participates in
E333-R-E350 = has as a participant

For the node associations:

Object

objectIdentifier	pointerCode
534	E678[OVA]
535	E678[OVA]
536	E678[OVA]
537	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
534	1	Node A to Node B	3[OVA]
535	1	Node A to External Source M	3[OVA]
536	1	Node B to External Destination L	3[OVA]
537	1	Node B to Node C	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
534	1	217	1	218	1	E362	NULL
535	1	217	1	221	1	E362	NULL
536	1	218	1	220	1	E362	NULL
537	1	218	1	219	1	E362	NULL

3. To relate the instance of Network to the instance of Document:

Object

objectIdentifier	pointerCode
563	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
563	1	Network to OV-2 Document	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
563	1	207	1	116	1	999	E333-R-E369

The relationTypeCode values used have the following meanings:

E333-R-E369 = is used to specify

4. For each of the *needlines*:

    a) One instance of InformationRequirement (a subtype of InformationTechnologyRequirement, which in turn is a subtype of Guidance)
    b) One instance of InformationNeedLineRequirement (a subtype of InformationTechnologyRequirement, which in turn is a subtype of Guidance)
    c) One instance of InformationExchangeRequirement (a subtype of InformationTechnologyRequirement which in turn is a subtype of Guidance)
    d) One instance of InformationTechnologyRequirement, which in turn is a subtype of Guidance to handle the implicit subtype *ProcessActivityExchangeRequirement*
    e) One instance of ObjectbyReference corresponding to *ProcessActivityExchangeRequirement*
    f) Three instances of ProcessActivity to reflect the notional activities shown in the template (Activity 1, Activity 2, Activity 3)
    g) One OVA to link IR to IER
    h) One OVA to link ENLR to IER
    i) One OVA to link PAER to IER
    j) OVAs to link ProcessActivity to PAER (note that in the template there is no way to know whether the ProcessActivity produces or consumes the IER)

Example: Needline 2 in OV-2 Template: Node A sending information Type Y to Node B.

For the characterization of the needline, we need IR, ENLR, IER and PAER. The first three are explicitly modeled in CADM v1.5. PAER is modeled through ObjectByReference. The instance tables for the InformationRequirement subtype hierarchy are shown below. Similar instantiation would take place for all the other subtypes.

Object

objectIdentifier	pointerCode
307	E246[InformationRequirement]
308	E164[ExchangeNeedLineRequirement]
309	E234[InformationExchangeRequirement]
310	E251[InformationTechnologyRequirement]
564	E679[OBR]
565	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
307	1	IR for Needline 2	4[ArchElem]
308	1	ENLR for Needline 2	4[ArchElem]
309	1	IER for Needline 2	4[ArchElem]
310	1	ITR for Needline 2	4[ArchElem]
564	1	PAER for Needline 2	5[OBR]
565	1	ITR[310] is a PAER[564]	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
564	1	E491[ProcessActivityExchangeRequirement]

ArchitectureElement

*Identifier	*Index	categoryCode
307	1	27 = GUIDANCE
308	1	27 = GUIDANCE
309	1	27 = GUIDANCE
310	1	27 = GUIDANCE

Guidance

*Identifier	*Index	categoryCode
307	1	13 = INFORMATION TECHNOLOGY
308	1	13 = INFORMATION TECHNOLOGY
309	1	13 = INFORMATION TECHNOLOGY
310	1	13 = INFORMATION TECHNOLOGY

InformationTechnologyRequirement

*Identifier	*Index	categoryCode
307	1	7 = INFORMATION REQUIREMENT
308	1	3 = EXCHANGE NEED LINE REQUIREMENT
309	1	4 = INFORMATION EXCHANGE REQUIREMENT
310	1	8 = PROCESS ACTIVITY EXCHANGE REQUIREMENT

InformationRequirement

*Identifier	*Index	accuracy Description Text	automated Processing Code	content Size Qty	exchange Frequency Text	graphic Page HighQty	graphic Page LowQty
307	1		1	500000	2/hr	200	10

methodVoiceVideoHigh ElapsedTimeQuantity	methodVoiceVideo LowElapsedTimeQuantity	perishability HighElapsed TimeQuantity	perishability LowElapsedTime Quantity	subscription TypeText
300	75	600	120	

tTransaction TypeText	volume Code
300	H

The special case of the implicit subtype *ProcessActivityExchangeRequirement* requires an entry in the ObjectVersionAssociation table to express the linkage to its supertype InformationExchangeRequirement.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
565	1	310	1	564	1	999	E251-S-E491

The relationTypeCode values used have the following meanings:

E251-S-E491 = is a

In the OV-2 template, there are three activities occurring at Node A and Node B. If, for example, Activity 1 (at Node A) is the producer ProcessActivity and Activity 2 (at Node) is the consumer ProcessActivity of the information, one can use the CADM v1.03 entity *ProcessActivityExchangeRequirement* (instantiated as ObjectByReference—see previous table

above), that supports that type of relationship. The instance table below shows how this is done in CADM v1.5.

Object

objectIdentifier	pointerCode
427	E486[ProcessActivity]
428	E486[ProcessActivity]
566	E678 [OVA]
567	E678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
427	1	Activity 1	4[ArchElem]
428	1	Activity 2	4[ArchElem]
566	1	Activity 1 produces IER	3[OVA]
567	1	Activity 2 consumes IER	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
566	1	427	1	564	1	999	E486-R2-E491
567	1	428	1	564	1	999	E486-R1-E491

The relationTypeCode values used have the following meanings:

E486-R1-E491 = is the consumer for
E486-R2-E491 = is the producer for

In CADM v1.5 the InformationExchangeRequirement can be linked to IR, ENLR, and PAER. The instance table below shows the case for Needline 2 in the OV-2 template.

Object

objectIdentifier	pointerCode
538	E678[OVA]
539	E678[OVA]
540	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
538	1	IR[307] to IER[309]	3[OVA]
539	1	ENLR[308] to IER[309]	3[OVA]
540	1	PAER[564] to IER[309]	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
538	1	307	1	309	1	999	E246-R-E234
539	1	308	1	309	1	999	E164-R-E234
540	1	564	1	309	1	999	E491-R-E234

The relationTypeCode values used have the following meanings:

E246-R-E234 = is used for
E164-R-E234 = uses
E491-R-E234 = is cited in

The relationship of the nodes involved in the IER is done by linking to it the specific node association for which the IER is defined.

Object

objectIdentifier	pointerCode
569	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
569	1	Node A to Node B for IER[309]	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
569	1	534	1	309	1	999	E362-R-E234

The relationTypeCode values used have the following meanings:

E362-R-E234 = is used to represent

4.6.4.4 Net-Centric Requirements

The specification of service functionality provider, service consumer, and unanticipated user at the operational level can be expressed in CADM v1.5 through OperationalRole, which is linkable to an OV-2 *node* through instances of ObjectByReference corresponding to the CADM v1.03 entity *NodeOperationalRole.* This allows the specification of the *role* identified for each *node*. The applicable codes in the ObjectVersionAssociation table are:

E359-R-E389 = [Node] represents [NodeOperationalRole]

E424-R-E389 = [OperationalRole] is represented by [NodeOperationalRole]

4.6.5 CADM v1.5 Support for Operational Information Exchange Matrix (OV-3)

4.6.5.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-3 details information exchanges and identifies "*who* exchanges *what* information, with *whom*, *why* the information is necessary, and *how* the information exchange must occur." [CJCSI 6212.01D] There is not a one-to-one mapping of OV-3 information exchanges to OV-2 needlines; rather, many individual information exchanges may be associated with one needline.

4.6.5.2 High-Level Description

Figure 4-19 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-3. As shown in the figure, the OV-3 is a presentation format for the exchanges that highlights the sender and receiver, the content of the exchange, the constraints imposed on the exchanges, and the rationale for the exchange.

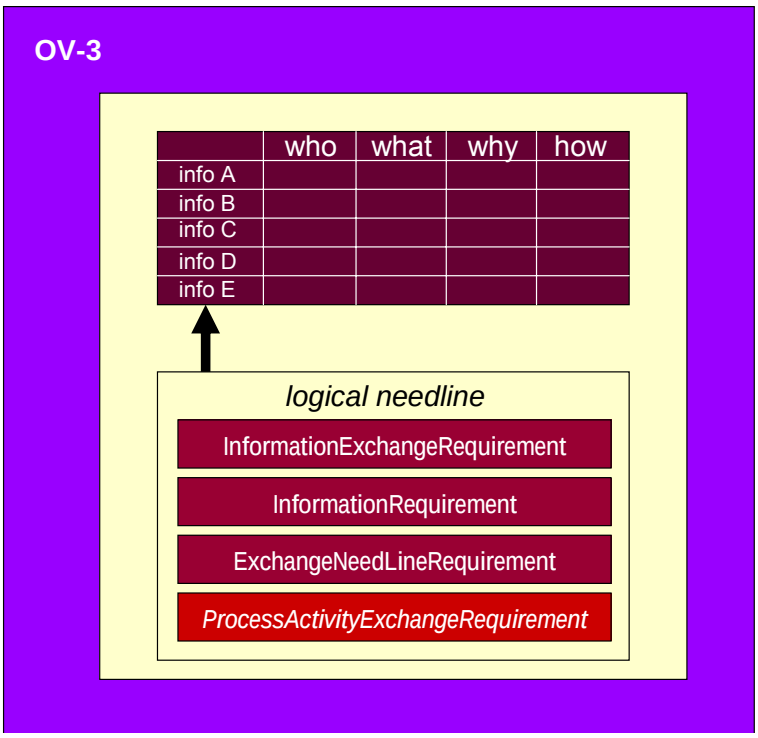


Figure 4-19: High-Level Depiction of CADM v1.5 Data Structures for OV-3 Representation

In CADM v1.5, the DoDAF architecture product OV-3 as an architecture product can be linked to the instance of Document representing the OV-3. In turn the instance of Architecture can be related to it. The actual data content of the OV-3 is built linking it to instances of *InformationExchangeMatrixElement*, which themselves can be linked to instances of InformationElement, ExchangeNeedLineRequirement, InformationExchangeRequirement, and *ProcessActivityExchangeRequirement*.

All this information is, in principle, captured already in the characterization of the logical *needlines* depicted in the OV-2. In CADM v1.5, the OV-3 is supported through the construct *InformationExchangeMatrixElement* from CADM v1.03, instantiated through ObjectByReference.

4.6.5.3 CADM v1.5 Instantiation

The OV-3 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E679[OBR]
601	E678[OVA]
602	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X4567 Architecture	4[ArchElem]
126	1	OV-3 Bravo-3	4[ArchElem]
127	1	ArchitectureDocument (OV-3 in Program Architecture[126]	5[OBR]
601	1	Architecture is documented by OV-3	3[OVA]
602	1	OV-3 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
601	1	125	1	127	1	999	E038-R-E045
602	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The instance of Document representing the OV-3 can now be linked to each of the required instances of *InformationExchangeMatrixElement* (expressed through ObjectByReference).

Object

objectIdentifier	pointerCode
701	679 [OBR]
702	679 [OBR]
703	679 [OBR]
704	679 [OBR]
671	678 [OVA]
672	678 [OVA]
673	678 [OVA]
674	678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
701	1	IER Matrix Element 1	5[OBR]
702	1	IER Matrix Element 2	5[OBR]
703	1	IER Matrix Element 3	5[OBR]
704	1	IER Matrix Element 4	5[OBR]
671	1	OV-3[125] contains IER ME 1	3[OVA]
672	1	OV-3[125] contains IER ME 2	3[OVA]
673	1	OV-3[125] contains IER ME 3	3[OVA]
674	1	OV-3[125] contains IER ME 4	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
701	1	E226[IER Matrix Element]
702	1	E226[IER Matrix Element]
703	1	E226[IER Matrix Element]
704	1	E226[IER Matrix Element]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
671	1	126	1	701	1	999	E225-R-E226
672	1	126	1	702	1	999	E225-R-E226
673	1	126	1	703	1	999	E225-R-E226
674	1	126	1	704	1	999	E225-R-E226

The relationTypeCode values used have the following meanings:

E225-R-E226 = contains

Finally, each *InformationExchangeMatrixElement* can be linked to the pertinent instance of InformationExchangeRequirement. For the purpose of illustration, one can take the instance already created for the OV-2 example discussed in the previous section. The only addition required is the new instance of OVA.

Object

objectIdentifier	pointerCode
309	E234[InformationExchangeRequirement]
901	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
309	1	IER for Needline 2	4[ArchElem]
901	1	IER[309]] to IER Matrix Element 1	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
901	1	701	1	309	1	999	E234-R-E226

The relationTypeCode values used have the following meanings:

E234-R-E226 = [IER] is referenced in [*InformationExchangeMatrixElement*]

4.6.5.4 Net-Centric Requirements

The specification of discovery metadata at the operational level can be expressed in CADM v1.5 through DiscoveryMetadata, which can be linked to the instance of Document corresponding to OV-3. Where discovery metadata needs to be specified individually for each of the instances of InformationExchangeRequirement, the association can be done through ObjectVersionAssociation with relationTypeCode = E147-R-E234 ([DiscoveryMetadata] is specified for [IER].

4.6.6 CADM v1.5 Support for Organizational Relationships Chart (OV-4)

4.6.6.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-4 illustrates the command structure or relationships (as opposed to relationships with respect to a business process flow) among human roles, organizations, or organization types that are the key players in an architecture.

4.6.6.2 High-Level Description

Figure 4-20 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-4 when represented in a notation neutral diagram.

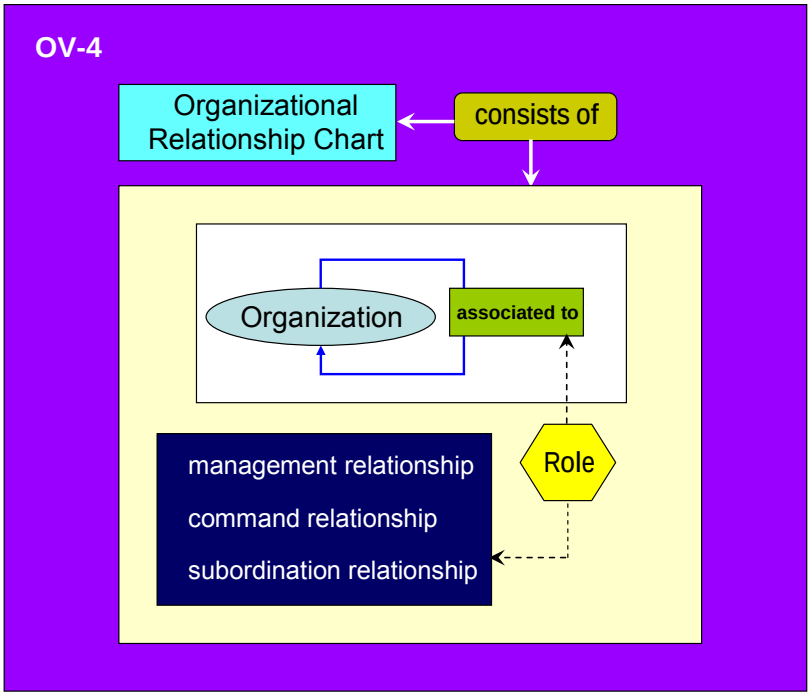


Figure 4-20: High-Level Depiction of CADM v1.5 Data Structures for OV-4 Representation (Notation Neutral)

In CADM v1.5, the DoDAF architecture product OV-4 as an architecture product is expressed as an instance of Document. The OV-4 can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference).

The actual data content of the OV-4 is built linking it to instances of *OrganizationalRelationshipChartElement* from CADM v1.03, instantiated through ObjectByReference, which themselves can be linked to instances of ObjectVersionAssociation corresponding to *OrganizationAssociation*, *OrganizationTypeAssociation*, or *NodeHierarchy*, a subtype of *NodeAssociation*.

4.6.6.3 CADM v1.5 Instantiation

The OV-4 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
105	E038[Architecture]
106	E148[Document]
107	E679[OBR]
822	E678[OVA]
823	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
105	1	Program Architecture C09	4[ArchElem]
106	1	Notional Organizational Relationship Chart	4[ArchElem]
107	1	ArchitectureDocument (OV-4 in Program Architecture C09)	5[OBR]
822	1	Architecture is documented by OV-4	3[OVA]
823	1	OV-4 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
107	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
822	1	105	1	107	1	999	E038-R-E045
823	1	106	1	107	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The instance of Document representing the OV-4 can now be linked to each of the required instances of *OrganizationalRelationshipChartElement* (expressed through ObjectByReference).

Object

objectIdentifier	pointerCode
401	679 [OBR]
402	679 [OBR]
403	679 [OBR]
404	679 [OBR]
405	679 [OBR]
406	679 [OBR]
171	678 [OVA]
172	678 [OVA]
173	678 [OVA]
174	678 [OVA]
175	678 [OVA]
176	678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
401	1	Org-Rel-Chart Element 1	5[OBR]
402	1	Org-Rel-Chart Element 2	5[OBR]
403	1	Org-Rel-Chart Element 3	5[OBR]
404	1	Org-Rel-Chart Element 4	5[OBR]
405	1	Org-Rel-Chart Element 5	5[OBR]
406	1	Org-Rel-Chart Element 6	5[OBR]
171	1	OV-4 [105] comprises ORC Elem 1	3[OVA]
172	1	OV-4 [105] comprises ORC Elem 2	3[OVA]
173	1	OV-4 [105] comprises ORC Elem 3	3[OVA]
174	1	OV-4 [105] comprises ORC Elem 4	3[OVA]
175	1	OV-4 [105] comprises ORC Elem 5	3[OVA]
176	1	OV-4 [105] comprises ORC Elem 6	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
401	1	E435 [Org-Rel-Chart Element]
402	1	E435 [Org-Rel-Chart Element]
403	1	E435 [Org-Rel-Chart Element]
404	1	E435 [Org-Rel-Chart Element]
405	1	E435 [Org-Rel-Chart Element]
406	1	E435 [Org-Rel-Chart Element]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
171	1	106	1	401	1	999	E225-R-E226
172	1	106	1	402	1	999	E225-R-E226
173	1	106	1	403	1	999	E225-R-E226
174	1	106	1	404	1	999	E225-R-E226
175	1	106	1	405	1	999	E225-R-E226
176	1	106	1	406	1	999	E225-R-E226

The relationTypeCode values used have the following meanings:

E225-R-E226 = contains

Finally, each *OrganizationalRelationshipChartElement* can be linked to the pertinent instance of ObjectVersionAssociation corresponding to either *OrganizationAssociation*, *OrganizationTypeAssociation*, or *NodeHierarchy*. To represent the content of the OV-4 template in Figure 4-11 of DoDAF v1.5 Volume II, it is appropriate to use *OrganizationAssociation*. The tables below show their instantiation and linkage to *OrganizationalRelationshipChartElement* from above.

Object

objectIdentifier	pointerCode
515	E432[Organization]
516	E432[Organization]
517	E432[Organization]
518	E432[Organization]
519	E432[Organization]
520	E432[Organization]
857	E678[OVA]
858	E678[OVA]
859	E678[OVA]
860	E678[OVA]
861	E678[OVA]
862	E678[OVA]
863	E678[OVA]
864	E678[OVA]
865	E678[OVA]
866	E678[OVA]
867	E678[OVA]
868	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
515	1	Top-Level Organization A	1 [OI]
516	1	Second-Level Organization B1	1 [OI]
517	1	Second-Level Organization B2	1 [OI]
518	1	Third-Level Organization C1	1 [OI]
519	1	Third-Level Organization C2	1 [OI]
520	1	Working Group D	1 [OI]
857	1	A to B1 link	3[OVA]
858	1	A to B2 link	3[OVA]
859	1	A to D link	3[OVA]
860	1	B1 to C1 link	3[OVA]
861	1	B1 to C2 link	3[OVA]
862	1	D to C1 link	3[OVA]
863	1	ORC Elem 1 to A-B1 Assoc	3[OVA]
864	1	ORC Elem 2 to A-B2 Assoc	3[OVA]
865	1	ORC Elem 3 to A-D Assoc	3[OVA]
866	1	ORC Elem 4 to B1-C1 Assoc	3[OVA]
867	1	ORC Elem 5 to B1-C2 Assoc	3[OVA]
868	1	ORC Elem 6 to D-C1 Assoc	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	categoryCode	relationType Code
857	1	515	1	516	1	E436[OrgAssoc]	NULL
858	1	515	1	517	1	E436[OrgAssoc]	NULL
859	1	515	1	520	1	E436[OrgAssoc]	NULL
860	1	516	1	518	1	E436[OrgAssoc]	NULL
861	1	516	1	519	1	E436[OrgAssoc]	NULL
862	1	520	1	518	1	E436[OrgAssoc]	NULL
863	1	857	1	401	1	999	E436-R-E435
864	1	858	1	402	1	999	E436-R-E435
865	1	859	1	403	1	999	E436-R-E435
866	1	860	1	404	1	999	E436-R-E435
867	1	861	1	405	1	999	E436-R-E435
868	1	862	1	406	1	999	E436-R-E435

The relationTypeCode values used have the following meanings:

E436-R-E435 = is used to define

4.6.6.4 Net-Centric Requirements

The specification COIs at the operational level can be expressed in CADM v1.5 through the use OrganizationType, which can be used to indicate the type of a given instance of Organization. Where the OV-4 is built using instances of OrganizationType directly, the root of the organizational chart can be typed as a COI and the associated organization types are understood as being part of that COI.

4.6.7 CADM v1.5 Support for Operational Activity Model (OV-5)

4.6.7.1 Product Definition

As described in DoDAF v1.5 Volume II, the OV-5 describes the operations that are normally conducted in the course of achieving a mission or a business capability. It describes capabilities, operational activities (or tasks), input and output (I/O) flows between activities, and I/O flows to/from activities that are outside the scope of the architecture. High-level operational activities should trace to (are decompositions of) a Business Area, an Internal Line of Business, and/or a Business Sub-Function as published in OMB's BRM [OMB, 2003].

4.6.7.2 High-Level Description

Figure 4-21 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-5 that is represented as an Integrated Definition for Activity Modeling (IDEF0) diagram.

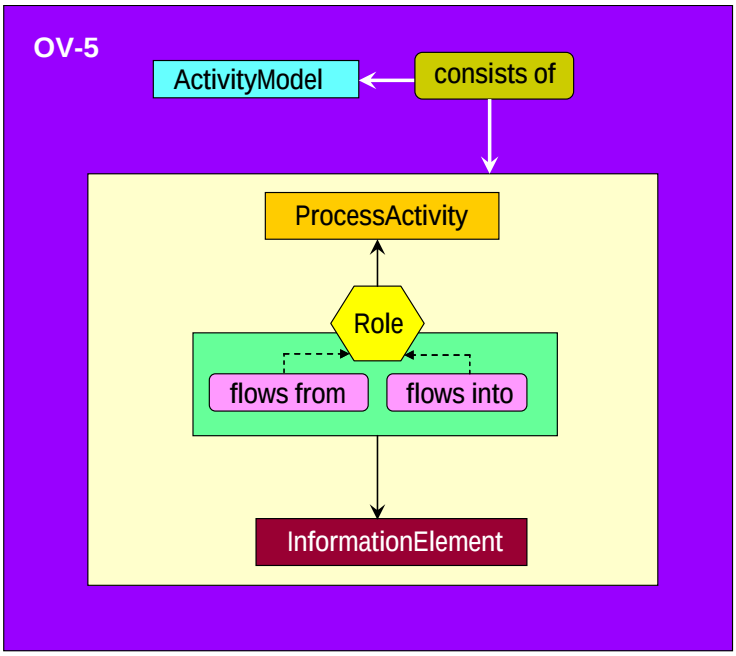


Figure 4-21: High-Level Depiction of CADM v1.5 Data Structures for OV-5 Representation (IDEF0 Style)

In CADM v1.5, the DoDAF architecture product OV-5 as an architecture product is expressed as an instance of Document. The OV-5 can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference). The relationship between InformationAsset and the Document instance corresponding to OV-5 provides access to its content.

The data content of a DoDAF architecture product OV-5 in IDEF0 notation is expressed in CADM v1.5 as an instance of ActivityModel (a subtype of InformationAsset) that is composed of *activities*, represented as instances of ProcessActivity. For each of the instances of ProcessActivity, there may be one or more instances of InformationElement (i.e., the *flows* between activities mentioned above). In the context of IDEF0, these flows may have specific "roles" (input, output, control or mechanism). Every InformationElement *flows from* some activity (either internal to the OV-5 or external to it) and *flows into* another activity (including the activity whence it originated for cases where there is a feedback loop).

4.6.7.3 CADM v1.5 Instantiation

The OV-5 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
305	E038[Architecture]
306	E148[Document]
307	E679[OBR]
611	E678[OVA]
612	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
305	1	Program Architecture C01	4[ArchElem]
306	1	Notional Activity Model	4[ArchElem]
307	1	ArchitectureDocument (OV-5 in Program Architecture[306]	5[OBR]
611	1	Architecture is documented by OV-5	3[OVA]
612	1	OV-5 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
307	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
611	1	305	1	307	1	999	E038-R-E045
612	1	306	1	307	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

Figure 4-14 in DoDAF Volume II depicts the template for OV-5 products using IDEF0 notation. As shown in that Figure, there may be any number of *activities* in an OV-5 built using this template. Each *activity* has a name, notionally depicted as A1, A2, etc., and there are flows starting at some *activity* and ending at another. As mentioned in the subsections above, the representation of this data content in CADM v1.5 utilizes an instance of ActivityModel, as many instances of ProcessAcitivity as there are *activities*, and as many instances of InformationElement as there are *flows* in the OV-5 (e.g., Flow 1, Flow 2, etc.)

The instance tables below show how CADM v1.5 captures the notional activity A2 from Figure 4-14 in DoDAF Volume II, and its associated *flows* (Flow 2 and Flow 3). As discussed in previous examples, the first step is to create all the instances of Object and ObjectVersion required. For this example, one needs (a) an instance corresponding ActivityModel, a subtype of InformationAsset (b) three instances for the *activities* A1, A2 and A3, (c) two instances for the *flows* as instances of InformationElement, (d) four instances for the ActivityModelInformationElementRole [AMIER] (the connector between the OV-5, the activities, the flows, and their respective roles) of ObjectByReference, (e) three instances of ObjectByReference corresponding to the CADM v1.03 *ActivityModelProcessActivity* [AMPA] to relate the instances of ProcessActivity A1, A2, and A3 to the ActivityModel (f) fourteen instances of ObjectVersionAssociation. Six of those are needed to relate the *activities* A1, A2, and A3 to the OV-5, wherein they are defined. The other eight are needed to express the linkage between each of the *flows* to their *activities*, and the roles each of the *flows* play in the context of the OV-5, wherein they reside.

Object

objectIdentifier	pointerCode
116	E009[ActivityModel]
217	E486[ProcessActvity]
218	E486[ProcessActvity]
219	E486[ProcessActvity]
320	E221[InformationElement]
321	E221[InformationElement]
425	E010[ActivityModelInformationElementRole]
426	E010[ActivityModelInformationElementRole]
427	E010[ActivityModelInformationElementRole]
428	E010[ActivityModelInformationElementRole]
522	E678[OVA]
523	E678[OVA]
524	E678[OVA]
525	E678[OVA]
526	E678[OVA]
527	E678[OVA]
528	E678[OVA]
529	E678[OVA]
530	E678[OVA]
531	E678[OVA]
532	E678[OVA]
533	E678[OVA]
822	E022[ActivityModelProcessActivity]
823	E022[ActivityModelProcessActivity]
824	E022[ActivityModelProcessActivity]

ObjectVersion

*Identifier	*Index	name	categoryCode
116	1	Level 1 Decomposition (Template)	4[ArchElem]
217	1	IDEF0 Activity A1	4[ArchElem]
218	1	IDEF0 Activity A2	4[ArchElem]
219	1	IDEF0 Activity A3	4[ArchElem]
320	1	IDEF0 Flow 2	4[ArchElem]
321	1	IDEF0 Flow 3	4[ArchElem]
425	1	AMIER01 for Flow 2	4[ArchElem]
426	1	AMIER02 for Flow 2	4[ArchElem]
427	1	AMIER03 for Flow 3	4[ArchElem]
428	1	AMIER04 for Flow 3	4[ArchElem]
522	1	A1 connected through AMPA01	3[OVA]
523	1	OV-5 connected through AMPA01	3[OVA]
524	1	A2 connected through AMPA02	3[OVA]
525	1	OV-5 connected through AMPA02	3[OVA]
526	1	A3 connected through AMPA03	3[OVA]
527	1	OV-5 connected through AMPA03	3[OVA]
528	1	AMPA[522] is part of AMIER01	3[OVA]
529	1	**Flow 1** in AMIER01 **starts at A1**	3[OVA]
530	1	AMPA[523] is part of AMIER02	3[OVA]
531	1	**Flow 1** in AMIER02 **ends at A2**	3[OVA]
532	1	AMPA[523] is part of AMIER03	3[OVA]
533	1	**Flow 2** in AMIER03 **starts at A2**	3[OVA]
534	1	AMPA[524] is part of AMIER04	3[OVA]
535	1	**Flow 2** in AMIER04 **ends at A3**	3[OVA]
822	1	AMPA01 (A1 in OV-5[116])	5[OBR]
823	1	AMPA02 (A2 in OV-5[116])	5[OBR]
824	1	AMPA03 (A3 in OV-5[116])	5[OBR]

Next, the actual linkages are built in the ObjectVersionAssociation table as shown below.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	116	1	822	1	999	E009-R-E022
523	1	217	1	822	1	999	E486-R-E022
524	1	116	1	823	1	999	E009-R-E022
525	1	218	1	823	1	999	E486-R-E022
526	1	116	1	824	1	999	E009-R-E022
527	1	219	1	824	1	999	E486-R-E022
528	1	822	1	425	1	999	E022-R-E010
529	1	**320**	1	425	1	999	E221-R-E010
530	1	823	1	426	1	999	E022-R-E010
531	1	**320**	1	426	1	999	E221-R-E010
532	1	823	1	427	1	999	E022-R-E010
533	1	**321**	1	427	1	999	E221-R-E010
534	1	824	1	428	1	999	E022-R-E010
535	1	**321**	1	428	1	999	E221-R-E010

The relationTypeCode values used have the following meanings:

E009-R-E022 = includes
E486-R-E022 = is included

E022-R-E010 = defines
E221-R-E010 = is associated with

Lastly, in order to express the 'role' that each of the *flows* has with respect to the *activity* where it starts and ends, one needs to instantiate the respective ActivityModelInformationElementRole [AMIER]. For simplicity, only the attribute corresponding to the *typeCode* is shown. The tables below show the entries in ArchitectureElement and ActivityModelInformationElementRole, but the corresponding entries in Object and ObjectVersion are left out.

ArchitectureElement

*Identifier	*Index	categoryCode
425	1	E010 [AMIER]
426	1	E010 [AMIER]
427	1	E010 [AMIER]
428	1	E010 [AMIER]

ActivityModelInformationElementRole

*Identifier	*Index	AMIER categoryCode
425	1	2[output]
426	1	1[input]
427	1	2[output]
428	1	1[input]

Retrieving the information for this segment of the OV-5, shown in Figure 4-14 of DoDAF Volume II could be accomplished by, for example, querying the database to find out all the instances of InformationElement. Once this is accomplished, the ObjectVersionAssociation table can be traversed to retrieve the related instances of ActivityModelInformationElementRole. Through this, one can retrieve the associated activities, since each instance of ActivityModelInformationElementRole points to the instance of ObjectByReference that corresponds to the *ActivityModelProcessActivity* [AMPA]. In the ObjectVersionAssociation table, the AMPA entries point to the corresponding instances of ActivityModel and ProcessActivity. Filtering for just the ProcessActivity permits to extract the name of the *activities* and, from the ActivityModelInformationElementRole, one already has the 'role' for the *flow*.

The table below shows the final result where for each one of the *flows* there are two 'roles' showing whether it is an "output" (Role = 2) or an "input" (Role = 1). As can be seen, the resulting query matches the content of the OV-5 shown in Figure 4-14 in DoDAF Volume II for the two flows, *Flow 2* and *Flow 3*.

Flow 2 is depicted as being an *output* of activity A1 and an *input* for activity A2. Similarly, *Flow 3* is shown as being an *output* of activity A2 and an *input* for activity A3.

Table 4-1 Example of a CADM v1.5 Query Showing Activities, Flows, and Their Roles for a Notional OV-5 Using IDEF0

Flow ID	Flow Index	Flow Name	Role	Activity ID	Activity Index	Activity Name
320	1	IDEF0 Flow 2	2	217	1	IDEF0 Activity A1
320	1	IDEF0 Flow 2	1	218	1	IDEF0 Activity A2
321	1	IDEF0 Flow 3	2	218	1	IDEF0 Activity A2
321	1	IDEF0 Flow 3	1	219	1	IDEF0 Activity A3

The OV-5 template given in Figure 4-15 in DoDAF v1.5 Volume II also mentions the possibility of annotating the product with information concerning the operational nodes that conduct them, the materiel that supports them, the cost of conducting the activity, and so forth. (The types of additional architecture data are notional.)

The cost of *activities* is supported in CADM v1.5 via the attribution of *ActivityModelProcessActivity* [AMPA] (instantiated as ObjectByReference). For each AMPA, one can provide its corresponding ObjectByReferenceCharacterization. The categoryCode = E022.A05 corresponds to the CADM v1.03 attribute *estimatedCostAmount* and it is the means to provide cost information related to the *activities* in OV-5.

To link OV-5 *activities* to the nodes that conduct them in CADM v1.5, one can create the appropriate instances of ObjectByReference corresponding to the associative entity *NodeProcessActivity* of CADM v1.03. In the characterization of each instance, one can set the categoryCode = E394.A01 (*roleCode*), and then choose the value 2 = SUPPORTS CONDUCT OF to state how the node and the OV-5 *activity* are related.

Linkage of OV-5 *activities* to materiel (i.e., classes of it) is normally represented in IDEF0 as a *mechanism* related to the respective OV-5 *activity*. This means that if there is a type of materiel that supports the performance of an OV-5 *activity*, one can create in CADM v1.5 the corresponding ActivityModelInformationElementRole for that *flow*, set the categoryCode = 4 (MECHANISM) and the subcategoryCode = 42 (REFERENCE) and then relate the MaterielType to it in the ObjectVersionAssociation table with the relationTypeCode = E292-R-E019 (*may be a*).

4.6.7.4 Net-Centric Requirements

The specification of SOA services at the operational level can be expressed in CADM v1.5 through OperationalRole, which is linkable to an OV-5 *activity* as a *mechanism* through the ActivityModelInformationElementRole for the *flows* identified for each *activity*. To do that, one needs to set in the respective instance of ActivityModelInformationElementRole the value for the categoryCode = 4 (MECHANISM) and the subcategoryCode = 42 (REFERENCE) and link it to the OperationalRole in the ObjectVersionAssociation table with the relationTypeCode = E424-R-E019 (*is cited as*).

4.6.8 CADM v1.5 Support for Operational Rules Model (OV-6a), Operational State Transition Description (OV-6b), and the Operational Event-Trace Description (OV-6c)

4.6.8.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-6a/b/c provides the timing and sequencing of events that capture operational behavior of a business process or mission thread, for example. Thus, this behavior is related to the activities of OV-5. Behavior modeling and documentation is essential to a successful architecture description, because it is how the architecture behaves that is crucial in many situations. Knowledge of the operational nodes, activities, and information exchanges is crucial; but knowing when, for example, a response should be expected after sending message X to node Y can also be crucial to achieving successful operations.

4.6.8.2 CADM v1.5 Support for Operational Rules Model (OV-6a)

4.6.8.2.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-6a specifies that operational or business rules are constraints on an enterprise, a mission, operation, business, or an architecture. While other OV products (e.g., OV-1, OV-2, and OV-5) describe the structure of a business—what the business can do—for the most part, they do not describe what the business *must* do, or what it *cannot* do.

4.6.8.2.2 High-Level Description

Figure 4-22 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-6a.

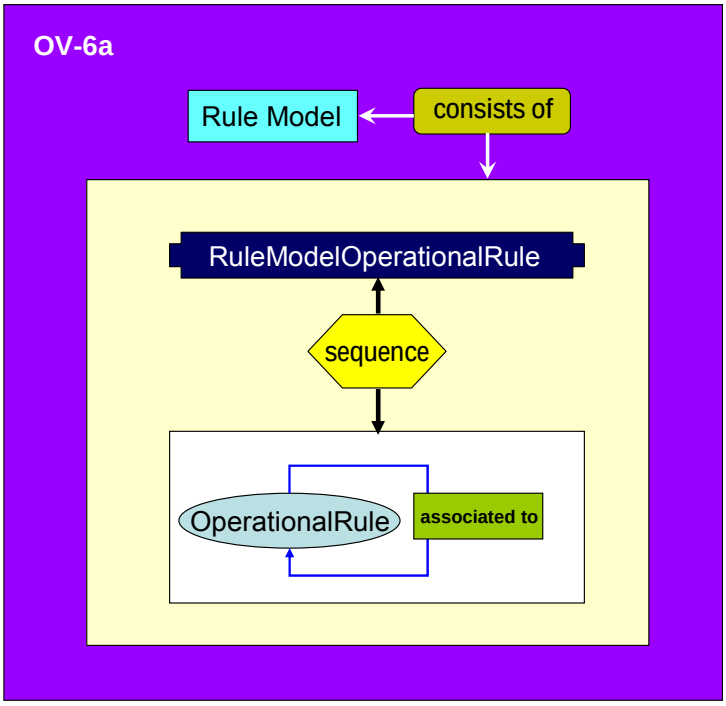


Figure 4-22: High-Level Depiction of CADM v1.5 Data Structures for OV-6a Representation

In CADM v1.5, the DoDAF architecture product OV-6a as an architecture product is expressed as an instance of Document. The OV-6a can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference). The actual data content of the OV-6a is built linking it to instances of the associative entity *RuleModelOperationalRule* from CADM v1.03 (instantiated through ObjectByReference), which collects the instances of OperationalRule (a subtype of Guidance) that make up the rule model itself.

4.6.8.2.3 CADM v1.5 Instantiation

Figure 4-18 in DoDAF v1.5 Volume II shows an example of what a rule model may contain, with the rules written in natural language.

The instantiation of OV-6a as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
275	E038[Architecture]
276	E148[Document]
277	E679[OBR]
111	E678[OVA]
112	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
275	1	Program Architecture C02	4[ArchElem]
276	1	Notional Rule Model	4[ArchElem]
277	1	ArchitectureDocument (OV-6a in Program Architecture C02)	5[OBR]
111	1	Architecture is documented by OV-6a	3[OVA]
112	1	OV-6a documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
277	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
111	1	275	1	277	1	999	E038-R-E045
112	1	276	1	277	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The instance of Document representing the OV-6a can be linked to each of the required instances of *RuleModelOperationalRule* (expressed through ObjectByReference), which will collect the instances of OperationalRule that make the content of the rule model. Note that *RuleModelOperationalRule* relates an instance of Document corresponding to an OV-6a to multiple instances of OperationalRule.

Object

objectIdentifier	pointerCode
371	E679[OBR]
372	E679[OBR]
373	E679[OBR]
374	E679[OBR]
381	E426[Op Rule]
382	E426[Op Rule]
383	E426[Op Rule]
384	E426[Op Rule]
291	E678 [OVA]
292	E678 [OVA]
293	E678 [OVA]
294	E678 [OVA]
295	E678 [OVA]
296	E678 [OVA]
297	E678 [OVA]
298	E678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
371	1	RuleModelOperationalRule D1-1	5[OBR]
372	1	RuleModelOperationalRule D1-2	5[OBR]
373	1	RuleModelOperationalRule D1-3	5[OBR]
374	1	RuleModelOperationalRule D1-4	5[OBR]
381	1	Op Rule 1	4 [ArchElem]
382	1	Op Rule 2	4 [ArchElem]
384	1	Op Rule 3	4 [ArchElem]
385	1	Op Rule 4	4 [ArchElem]
291	1	OV-7 cites Op Rule 1	3 [OVA]
292	1	Op Rule 1 is cited for OV-7	3 [OVA]
293	1	OV-7 cites Op Rule 2	3 [OVA]
294	1	Op Rule 2 is cited for OV-7	3 [OVA]
295	1	OV-7 cites Op Rule 3	3 [OVA]
296	1	Op Rule 3 is cited for OV-7	3 [OVA]
297	1	OV-7 cites Op Rule 4	3 [OVA]
298	1	Op Rule 4 is cited for OV-7	3 [OVA]

ObjectByReference

*Identifier	*Index	categoryCode
371	1	E521[RuleModelOperationalRule]
372	1	E521[RuleModelOperationalRule]
373	1	E521[RuleModelOperationalRule]
374	1	E521[RuleModelOperationalRule]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
291	1	276	1	371	1	999	E520-R-E521
292	1	381	1	371	1	999	E426-R-E521
293	1	276	1	372	1	999	E520-R-E521
294	1	382	1	372	1	999	E426-R-E521
295	1	276	1	373	1	999	E520-R-E521
296	1	384	1	373	1	999	E426-R-E521
297	1	276	1	374	1	999	E520-R-E521
298	1	385	1	374	1	999	E426-R-E521

The relationTypeCode values used have the following meanings:

E520-R-E521 = cites
E426-R-E521 = is cited for

The actual rule is stored in the appropriate attributes of each of the instances of Guidance and OperationalRule. The textual description of the operational rule is recorded in the text of Guidance. A further characterization of the rule can be stated through the attribution of OperationalRule. The instance tables below show how this is done for the notional example (shown in Figure 4-18 in DoDAF v1.5 Volume II) discussed in this section.

ArchitectureElement

*Identifier	*Index	categoryCode
307	1	27 = GUIDANCE
308	1	27 = GUIDANCE
309	1	27 = GUIDANCE
310	1	27 = GUIDANCE

Guidance

*Identifier	*Index	categoryCode	subject Text	text
307	1	13 = OPERATIONAL RULE	Rule 1 for Rule Model A	**IF** activity is *Battle Damage Assessment* **THEN** it consists of *Conduct Battle Damage Assessment*, *Conduct Munitions Effects Assessment*, and *Recommend Restrike*
308	1	13 = OPERATIONAL RULE	Rule 2 for Rule Model A	**IF** *Battle Damage Assessment Report* completed **THEN** *Recommend Restrike* can be completed
309	1	13 = OPERATIONAL RULE	Rule 3 for Rule Model A	**IF** *Munitions Effects Assessment Report* completed **THEN** *Recommend Restrike* can be completed
310	1	13 = OPERATIONAL RULE	Rule 4 for Rule Model A	**IF** *Recommend Restrike* occurs **THEN** facts of *Recommend Restrike* must be based on facts from (*Battle Damage Assessment Report* **AND** *Munitions Effects Assessment Report)*

OperationalRule

*Identifier	*Index	categoryCode	formalLanguage Name
307	5	4 = CRITERION	First Order Predicate Logic
308	6	4 = CRITERION	First Order Predicate Logic
309	6	4 = CRITERION	First Order Predicate Logic
310	9	4 = CRITERION	First Order Predicate Logic

4.6.8.2.4 Net-Centric Requirements

Since the purpose of the OV-6a is to specify operational or business rules that are constraints on an enterprise, mission, operation, business, or architecture, it would subsequently include any required operational or business rules that support NCO**.** Accordingly, the CADM support for the OV-6a is well suited to support the NCE.

4.6.8.3 CADM v1.5 Support for Operational State Transition Description (OV-6b)

4.6.8.3.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-6b is a graphical method of describing how an operational node or activity responds to various events by changing its state. The diagram represents the sets of events to which the architecture will respond (by taking an action to move to a new state) as a function of its current state. Each transition specifies an event and an action.

4.6.8.3.2 High-Level Description

Figure 4-23 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-6b.

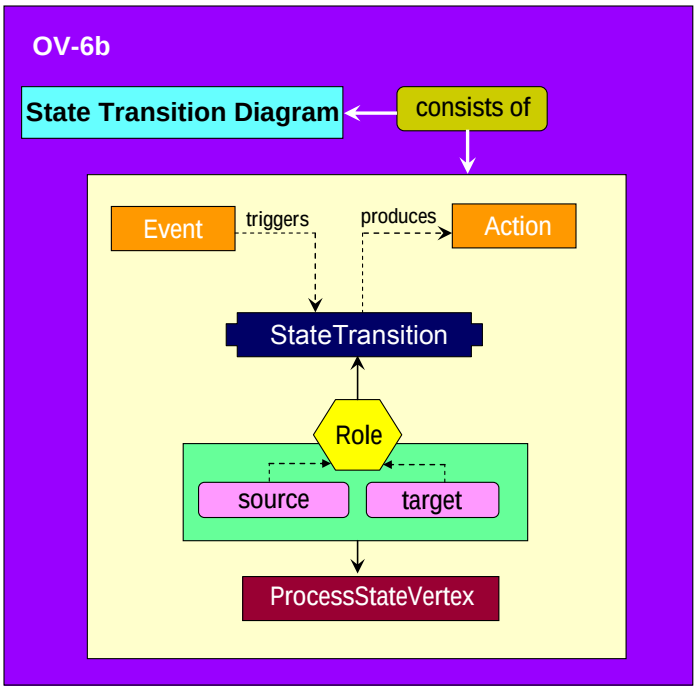


Figure 4-23: High-Level Depiction of CADM v1.5 Data Structures for OV-6b Representation

In CADM v1.5, the DoDAF architecture product OV-6b as an architecture product is expressed as an instance of Document with the architectureProductCategoryCode = 28 [STATE

TRANSITION DESCRIPTION] and the architectureProductSubcategoryCode = 281 [OPERATIONAL STATE TRANSITION DESCRIPTION]**.** The OV-6b can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference).

The actual data content of the OV-6b is built linking it to TransitionProcess, which collects the instances of ProcessStateVertex (the supertype of *ProcessState*, and *ProcessPseudoState* in CADM v1.03. *ProcessState* in CADM v1.03 further subtypes into *NestingProcessState* and *CompositeProcessState*). Instances of Action can be related to a given ProcessStateVertex (subtyped as *ProcessState*) through *ProcessStateAction* to indicate the entry into and exit out of the state transition. Instances of Event (specialized as *ProcessEvent*) can be used to express the *triggers* for the state transitions. The associative entity from CADM v1.03 *TransitionProcessResultingAction* (instantiated as ObjectByReference) links each transition to the outcome actions. The linkage of the resulting actions to OV-5 ProcessActivity instances is also supported through the CADM v1.03 associative entity *ProcessActivityAction.*

4.6.8.3.3 CADM v1.5 Instantiation

Figure 4-24 shows an example of an operational state transition diagram for air traffic operations.

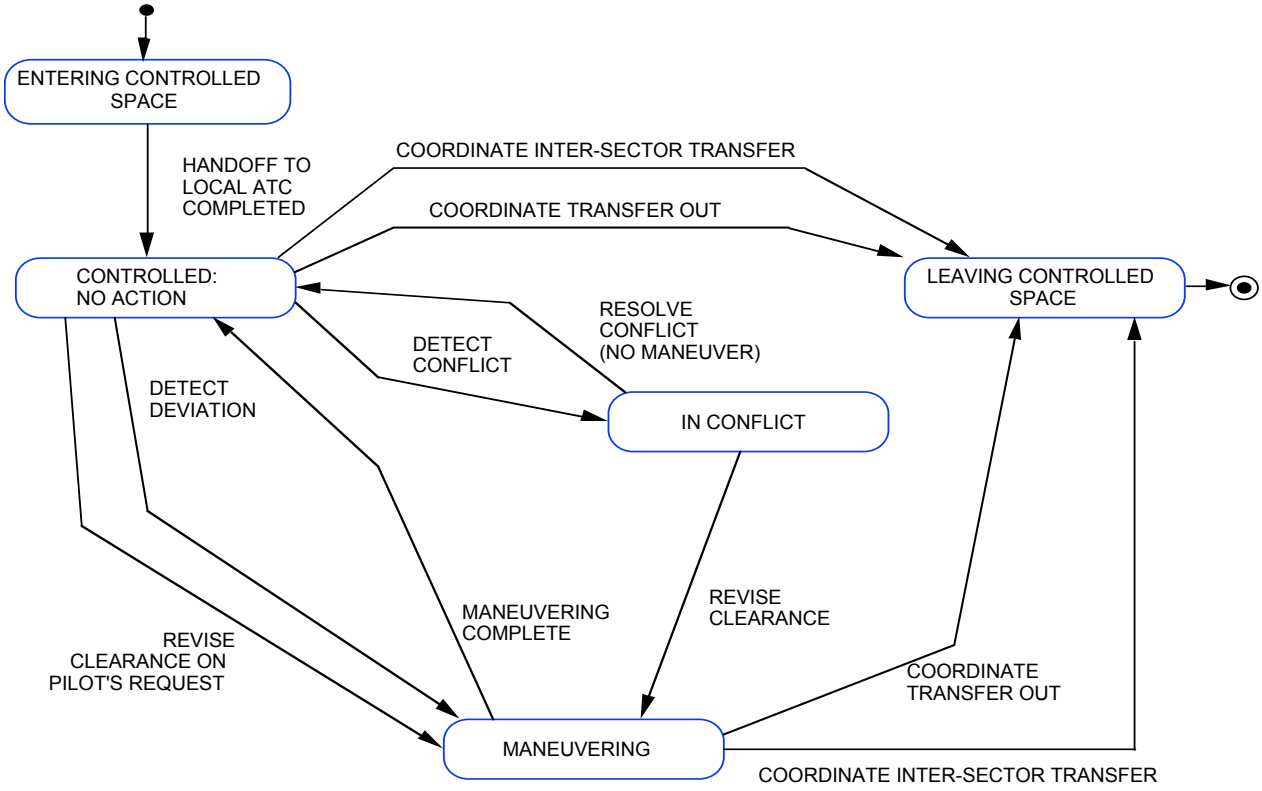


Figure 4-24: Operational OV-6b Air Traffic Operations Example

The instantiation of OV-6b for this example as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E679[OBR]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	OV-6b- Air Traffic Operations	4[ArchElem]
127	1	ArchitectureDocument (OV-6b in Project X Architecture)	5[OBR]
522	1	Architecture is documented by OV-6b	3[OVA]
523	1	OV-6b documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

For an OV-6b, the following convention is used: Instances of ProcessStateVertex are created for oval elements of the OV-6b diagram and the initial and final states (which are treated as pseudo states). This results in the following mapping for the example shown in Figure 4-24:

Initial state (as a *ProcessPseudoState*, a subtype of ProcessStateVertex)

Entering Controlled Space (as a *ProcessState*, a subtype of ProcessStateVertex)

Controlled: No Action (as a *ProcessState*, a subtype of ProcessStateVertex)

In Conflict (as a *ProcessState*, a subtype of ProcessStateVertex)

Maneuvering (as a *ProcessState*, a subtype of ProcessStateVertex)

Leaving Controlled-Space (as a *ProcessState*, a subtype of ProcessStateVertex)

End state (as a ***ProcessPseudoState***, a subtype of ProcessStateVertex)

The instantiation of these states is shown below:

Object

objectIdentifier	pointerCode
307	E502[ProcessStateVertex]
308	E502[ProcessStateVertex]
309	E502[ProcessStateVertex]
310	E502[ProcessStateVertex]
311	E502[ProcessStateVertex]
312	E502[ProcessStateVertex]
313	E502[ProcessStateVertex]

ObjectVersion

*Identifier	*Index	name	categoryCode
307	1	Initial state	4[ArchElem]
308	1	Entering Controlled Space	4[ArchElem]
309	1	Controlled: No Action	4[ArchElem]
310	1	In Conflict	4[ArchElem]
311	1	Maneuvering	4[ArchElem]
312	1	Leaving Controlled-Space	4[ArchElem]
313	1	End state	4[ArchElem]

The actions associated with the pseudo states are those that represent the *entry* and *exit* for the state transition diagram. The tables below show their instantiation and how they are linked to the pseudo states.

Object

objectIdentifier	pointerCode
611	E001[Action]
612	E001[Action]
614	E679[OBR]
615	E679[OBR]
616	E678[OVA]
617	E678[OVA]
618	E678[OVA]
619	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
611	1	Action A001	4[ArchElem]
612	1	Action A002	4[ArchElem]
614	1	ProcessStateAction PSA001	5[OBR]
615	1	ProcessStateAction PSA002	5[OBR]
616	1	Action A001 to Initial state	3[OVA]
617	1	Initial state to Action A001	3[OVA]
618	1	Action A002 to End state	3[OVA]
619	1	End state to Action A002	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
614	1	E501[ProcessStateAction]
615	1	E501[ProcessStateAction]

ObjectByReferenceCharacterization

*Identifier	OBR Identifier	OBR Index	categoryCode	valueText
101	614	1	E501.A01	1
102	614	1	E501.A02	1 (entry)
103	615	1	E501.A01	1
104	615	1	E501.A02	2 (exit)

The catergoryCode values used have the following meanings:

E501.A01 = SequenceIdentifierText

E501.A02 = RoleCode

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
616	1	611	1	614	1	999	E001-R-E501
617	1	307	1	614	1	999	E500-R-E501
618	1	612	1	615	1	999	E001-R-E501
619	1	313	1	615	1	999	E500-R-E501

The relationTypeCode values used have the following meanings:

E001-R-E501 = represents

E500-R-E501 = represents

As indicated above, the content of the OV-6b is expressed through the instantiation of TranstionProcess. For each instance of TranstionProcess, one can indicate its source and target states (represented in CADM as instances of ProcessStateVertex), the event that triggers the transition, as well as the operational rule that may act as the guard condition for the transition. In CADM v1.5, these links are all represented through entries in the ObjectVersionAssociation table. The attribute labelName in TranstionProcess is used to capture the text that is attached to each of the arrows in the state transition diagram. The instance tables below describe how this is done for the example shown in Figure 4-24.

Object

objectIdentifier	pointerCode
701	E663[TransitionProcess]
702	E663[TransitionProcess]
703	E663[TransitionProcess]
704	E663[TransitionProcess]
705	E663[TransitionProcess]
706	E663[TransitionProcess]
707	E663[TransitionProcess]
708	E663[TransitionProcess]
709	E663[TransitionProcess]
710	E663[TransitionProcess]
711	E663[TransitionProcess]
712	E663[TransitionProcess]
713	E663[TransitionProcess]

ObjectVersion

*Identifier	*Index	name	categoryCode
701	1	TRN001	4[ArchElem]
702	1	TRN002	4[ArchElem]
703	1	TRN003	4[ArchElem]
704	1	TRN004	4[ArchElem]
705	1	TRN005	4[ArchElem]
706	1	TRN006	4[ArchElem]
707	1	TRN007	4[ArchElem]
708	1	TRN008	4[ArchElem]
709	1	TRN009	4[ArchElem]
710	1	TRN010	4[ArchElem]
711	1	TRN011	4[ArchElem]
712	1	TRN012	4[ArchElem]
713	1	TRN013	4[ArchElem]

ArchitectureElement

*Identifier	*Index	categoryCode
701	1	72 = TRANSITION-PROCESS
702	1	72 = TRANSITION-PROCESS
703	1	72 = TRANSITION-PROCESS
704	1	72 = TRANSITION-PROCESS
705	1	72 = TRANSITION-PROCESS
706	1	72 = TRANSITION-PROCESS
707	1	72 = TRANSITION-PROCESS
708	1	72 = TRANSITION-PROCESS
709	1	72 = TRANSITION-PROCESS
710	1	72 = TRANSITION-PROCESS
711	1	72 = TRANSITION-PROCESS
712	1	72 = TRANSITION-PROCESS
713	1	72 = TRANSITION-PROCESS

TransitionProcess

*Identifier	*Index	labelName
701	1	—
702	1	Handoff to local ATC completed
703	1	Coordinate Inter-sector transfer
704	1	Coordinate transfer out
705	1	Resolve conflict (no maneuver)
706	1	Detect conflict
707	1	Detect deviation
708	1	Revise clearance on pilot’s request
709	1	Maneuvering complete
710	1	Revise clearance
711	1	Coordinate transfer out
712	1	Coordinate inter-sector transfer
713	1	—

The intances of ObjectVersionAssociation required to specify the source and target state for each transition are shown below.

Object

objectIdentifier	pointerCode
901	E678[OVA]
902	E678[OVA]
903	E678[OVA]
904	E678[OVA]
905	E678[OVA]
906	E678[OVA]
907	E678[OVA]
908	E678[OVA]
909	E678[OVA]
910	E678[OVA]
911	E678[OVA]
912	E678[OVA]
913	E678[OVA]
914	E678[OVA]
915	E678[OVA]
916	E678[OVA]
917	E678[OVA]
918	E678[OVA]
919	E678[OVA]
920	E678[OVA]
921	E678[OVA]
922	E678[OVA]
923	E678[OVA]
924	E678[OVA]
925	E678[OVA]
926	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
901	1	PSV[307] -- PSV[308] {source}	E678[OVA]
902	1	PSV[307] -- PSV[308] {target}	E678[OVA]
903	1	PSV[308] -- PSV[309] {source}	E678[OVA]
904	1	PSV[308] -- PSV[309] {target}	E678[OVA]
905	1	PSV[309] -- PSV[310] {source}	E678[OVA]
906	1	PSV[309] -- PSV[310] {target}	E678[OVA]
907	1	PSV[309] -- PSV[310] {source}	E678[OVA]
908	1	PSV[309] -- PSV[310] {target}	E678[OVA]
909	1	PSV[311] -- PSV[309] {source}	E678[OVA]
910	1	PSV[311] -- PSV[309] {target}	E678[OVA]
911	1	PSV[309] -- PSV[311] {source}	E678[OVA]
912	1	PSV[309] -- PSV[311] {target}	E678[OVA]
913	1	PSV[309] -- PSV[312] {source}	E678[OVA]
914	1	PSV[309] -- PSV[312] {target}	E678[OVA]
915	1	PSV[309] -- PSV[312] {source}	E678[OVA]
916	1	PSV[309] -- PSV[312] {target}	E678[OVA]
917	1	PSV[312] -- PSV[309] {source}	E678[OVA]
918	1	PSV[312] -- PSV[309] {target}	E678[OVA]
919	1	PSV[311] -- PSV[312] {source}	E678[OVA]

*Identifier	*Index	name	categoryCode
920	1	PSV[311] -- PSV[312] {target}	E678[OVA]
921	1	PSV[312] -- PSV[310] {source}	E678[OVA]
922	1	PSV[312] -- PSV[310] {target}	E678[OVA]
923	1	PSV[312] -- PSV[310] {source}	E678[OVA]
924	1	PSV[312] -- PSV[310] {target}	E678[OVA]
925	1	PSV[310] -- PSV[313] {source}	E678[OVA]
926	1	PSV[310] -- PSV[313] {target}	E678[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
901	1	307	1	701	1	999	E502-R2-E663
902	1	308	1	701	1	999	E502-R1-E663
903	1	308	1	702	1	999	E502-R2-E663
904	1	309	1	702	1	999	E502-R1-E663
905	1	309	1	703	1	999	E502-R2-E663
906	1	310	1	703	1	999	E502-R1-E663
907	1	309	1	704	1	999	E502-R2-E663
908	1	310	1	704	1	999	E502-R1-E663
909	1	311	1	705	1	999	E502-R2-E663
910	1	309	1	705	1	999	E502-R1-E663
911	1	309	1	706	1	999	E502-R2-E663
912	1	311	1	706	1	999	E502-R1-E663
913	1	309	1	707	1	999	E502-R2-E663
914	1	312	1	707	1	999	E502-R1-E663
915	1	309	1	708	1	999	E502-R2-E663
916	1	312	1	708	1	999	E502-R1-E663
917	1	312	1	709	1	999	E502-R2-E663
918	1	309	1	709	1	999	E502-R1-E663
919	1	311	1	710	1	999	E502-R2-E663
920	1	312	1	710	1	999	E502-R1-E663
921	1	312	1	711	1	999	E502-R2-E663
922	1	310	1	711	1	999	E502-R1-E663
923	1	312	1	712	1	999	E502-R2-E663
924	1	310	1	712	1	999	E502-R1-E663
925	1	310	1	713	1	999	E502-R2-E663
926	1	313	1	713	1	999	E502-R1-E663

The relationTypeCode values used have the following meanings:

E502-R2-E663 = is source for
E502-R1-E663 = is target for

The same approach would be used to link the corresponding trigger events to each of the instances of TransitionProcess.

The link between the OV-6b Document and each of the instances of TransitionProcess is done in a similar fashion through ObjectVersionAssociation with the owning Document instance linked to each of the component instances of TransitionProcess.

4.6.8.3.4 Net-Centric Requirements

Since the purpose of the OV-6b is to specify a graphical method of describing how an operational node or activity responds to various events by changing its state, it would subsequently include any required operational or business rules that support NCO**.** Accordingly, the CADM support for the OV-6b is well suited to support the NCE.

4.6.8.4 CADM v1.5 Support for Operational Event-Trace Description (OV-6c)

4.6.8.4.1 Product Definition

As stated in DoDAF v1.5 Volume II, the OV-6c provides a time-ordered examination of the information exchanges between participating operational nodes as a result of a particular scenario. Each event-trace diagram should have an accompanying description that defines the particular scenario or situation.

4.6.8.4.2 High-Level Description

Figure 4-25 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-6c.

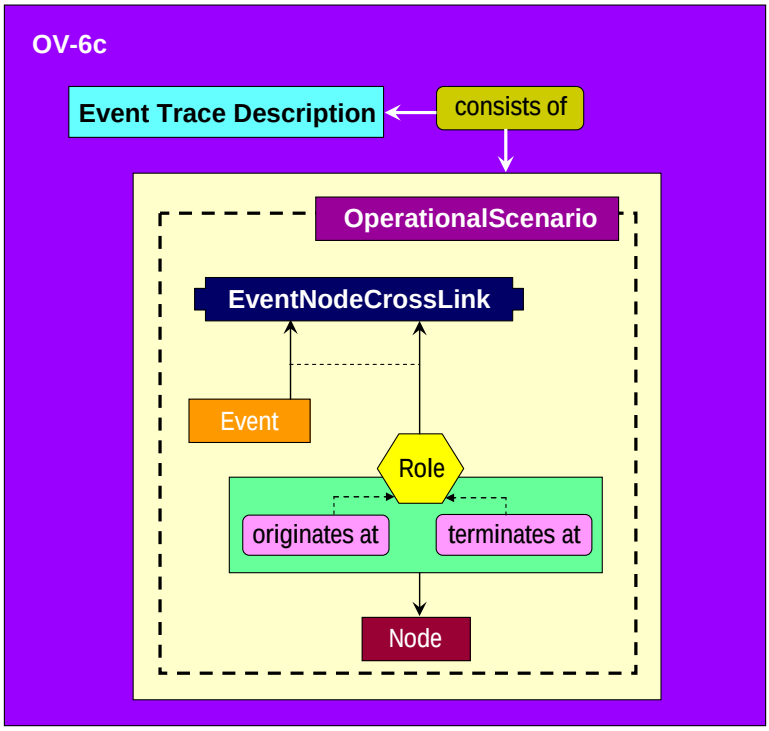


Figure 4-25: High-Level Depiction of CADM v1.5 Data Structures for OV-6c Representation

The DoDAF architecture product OV-6c is expressed in CADM v1.5 as an instance of Document. The OV-6c can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference).

The actual data content of the OV-6c is built linking it to instances of EventNodeCrossLink, which collects the instances of Node with roles “originating” and “terminating” with regard to the event trace. For every pair of nodes, one can also specify the instances of Event that are involved in the temporal sequence. The OV-6c link to OperationalScenario provides the context within which the event trace description takes place.

In addition to the links mentioned above, the EventNodeCrossLink can also be related to instances of OperationalRule and DirectedConstraint, both subtypes of Guidance.

4.6.8.4.3 CADM v1.5 Instantiation

Figure 4-22 in DoDAF v1.5 Volume II shows an example of what an event-trace description may contain.

The instantiation of OV-6c as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E045[ArchitectureDocument]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	OV-6c	4[ArchElem]
127	1	ArchitectureDocument (OV-6b in Project X Architecture)	3[OVA]
522	1	Architecture is documented by OV-6c	3[OVA]
523	1	OV-6c documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

Object

objectIdentifier	pointerCode
306	E427[OperationalScenario]
307	E159[EventNodeCrosslink]
308	E159[EventNodeCrosslink]
309	E159[EventNodeCrosslink]
310	E159[EventNodeCrosslink]
311	E159[EventNodeCrosslink]
312	E159[EventNodeCrosslink]
313	E159[EventNodeCrosslink]
314	E159[EventNodeCrosslink]
315	E359[Node]
316	E359[Node]
317	E359[Node]
318	E156[Event]
319	E156[Event]
320	E156[Event]
321	E156[Event]
322	E156[Event]
323	E156[Event]
324	E156[Event]
325	E156[Event]
564	E678[OVA]
565	E678[OVA]
566	E678[OVA]
567	E678[OVA]
568	E678[OVA]
569	E678[OVA]
570	E678[OVA]
571	E678[OVA]
572	E678[OVA]
573	E678[OVA]
574	E678[OVA]
575	E678[OVA]
576	E678[OVA]
577	E678[OVA]
578	E678[OVA]
579	E678[OVA]
580	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
306	1	OV-6c Scenario	4[ArchElem]
307	1	EventNodeCrosslink1[Event 1 for Node 2(316)]	5[OBR]
308	1	EventNodeCrosslink2[Event 2 for Node 2(316)]	5[OBR]
309	1	EventNodeCrosslink3[Event 3 for Node 2(316)]	5[OBR]
310	1	EventNodeCrosslink4[Event 4 for Node 3(317)]	5[OBR]
311	1	EventNodeCrosslink5[Event 5 for Node 1(315)]	5[OBR]

*Identifier	*Index	name	categoryCode
312	1	EventNodeCrosslink6[Event 6 for Node 2(316)]	5[OBR]
313	1	EventNodeCrosslink7[Event 7 for Node 1(315)]	5[OBR]
314	1	EventNodeCrosslink8[Event 8 for Node 3(317)]	5[OBR]
315	1	Node 1	4[ArchElem]
316	1	Node 2	4[ArchElem]
317	1	Node 3	4[ArchElem]
318	1	Event 1	4[ArchElem]
319	1	Event 2	4[ArchElem]
320	1	Event 3	4[ArchElem]
321	1	Event 4	4[ArchElem]
322	1	Event 5	4[ArchElem]
323	1	Event 6	4[ArchElem]
324	1	Event 7	4[ArchElem]
325	1	Event 8	4[ArchElem]
564	1	OV-6c Scenario describes Document	3[OVA]
565	1	Event 1[318] is part of EventNodeCrosslink 1[307]	3[OVA]
566	1	EventNodeCrosslink 1 to Node2[316]	3[OVA]
567	1	Event 2[319] is part of EventNodeCrosslink 2[308]	3[OVA]
568	1	EventNodeCrosslink 2 to Node2[316]	3[OVA]
569	1	Event 3[320] is part of EventNodeCrosslink 3[309]	3[OVA]
570	1	EventNodeCrosslink 3 to Node2[316]	3[OVA]
571	1	Event 4[321] is part of EventNodeCrosslink 4[310]	3[OVA]
572	1	EventNodeCrosslink 4 to Node3[317]	3[OVA]
573	1	Event 5[322] is part of EventNodeCrosslink 5[311]	3[OVA]
574	1	EventNodeCrosslink 5 to Node1[315]	3[OVA]
575	1	Event 6[323] is part of EventNodeCrosslink 6[312]	3[OVA]
576	1	EventNodeCrosslink 6 to Node2[316]	3[OVA]
577	1	Event 7[324] is part of EventNodeCrosslink 7[313]	3[OVA]
578	1	EventNodeCrosslink 7 to Node1[315]	3[OVA]
579	1	Event 8[325] is part of EventNodeCrosslink 8[314]	3[OVA]
580	1	EventNodeCrosslink 8 to Node3[317]	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
564	1	306	1	126	1	999	E148-R-E427
565	1	307	1	318	1	999	E156-R-E159
566	1	307	1	316	1	999	E359-R-E159
567	1	308	1	319	1	999	E156-R-E159
568	1	308	1	316	1	999	E359-R-E159
569	1	309	1	320	1	999	E156-R-E159
570	1	309	1	316	1	999	E359-R-E159
571	1	310	1	321	1	999	E156-R-E159

572	1	310	1	317	1	999	E359-R-E159
573	1	311	1	322	1	999	E156-R-E159
574	1	311	1	315	1	999	E359-R-E159
575	1	312	1	323	1	999	E156-R-E159
576	1	312	1	316	1	999	E359-R-E159
577	1	313	1	324	1	999	E156-R-E159
578	1	313	1	315	1	999	E359-R-E159
579	1	314	1	325	1	999	E156-R-E159
580	1	314	1	317	1	999	E359-R-E159

The relationTypeCode values used have the following meanings:

E148-R-E427 = describes
E156-R-E159 = is crosslink for
E359-R-E159 = is the terminator for

4.6.8.4.4 Net-Centric Requirements

The specification of service functionality provider, service consumer, and unanticipated user at the operational level can be expressed in CADM v1.5 through OperationalRole, which is linkable to an OV-2 *node* through instances of ObjectByReference corresponding to the CADM v1.03 entity *NodeOperationalRole.* This allows the specification of the *role* identified for each *node*. The applicable codes in the ObjectVersionAssociation table are:

E359-R-E389 = [Node] represents [NodeOperationalRole]

E424-R-E389 = [OperationalRole] is represented by [NodeOperationalRole]

4.6.9 CADM v1.5 Support for Logical Data Model (OV-7)

4.6.9.1 Product Definition

As stated in DoDAF v1.5 Volume II, OV-7 describes the structure of an architecture domain's system data types and the structural business process rules (defined in the architecture's OV) that govern the system data. It provides a definition of architecture domain data types, their attributes or characteristics, and their interrelationships.

4.6.9.2 High-Level Description

Figure 4-26 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an OV-7 that is represented as an IDEF1X diagram.

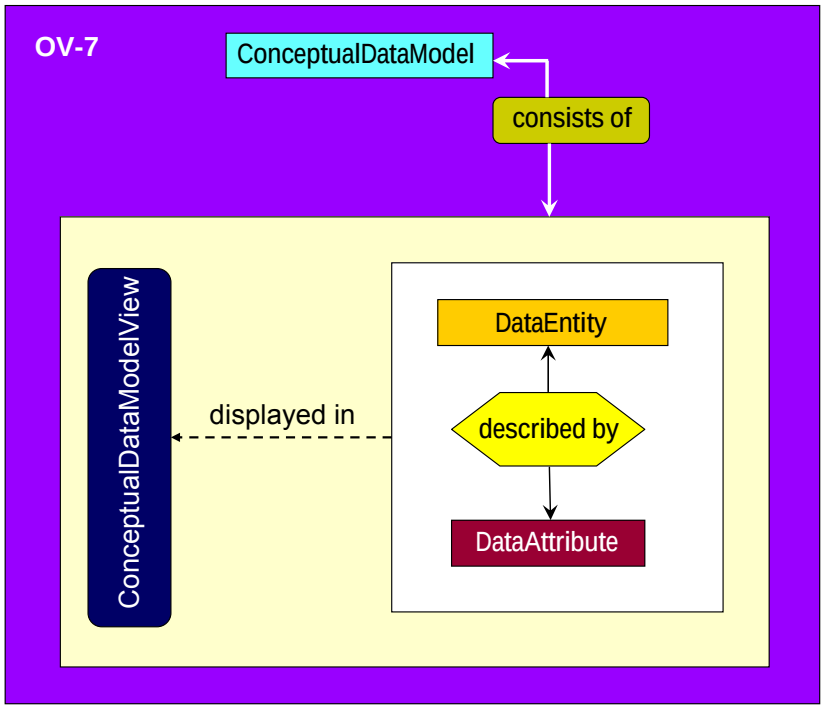


Figure 4-26: High-Level Depiction of CADM v1.5 Data Structures for OV-7 Representation (IDEF1X Style)

The DoDAF architecture product OV-7 is expressed in CADM v1.5 as an instance of Document. The OV-7 can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference). The OV-7 can be linked to its corresponding instance of InformationAsset, which represents the actual content of the model.

The data content of a DoDAF architecture product OV-7 in IDEF1X notation is expressed in CADM v1.5 as an instance of ConceptualModel (a subtype of InformationAsset) that is composed of *entities*, represented as instances of DataEntity. For each of the instances of DataEntity, there may be one or more instances of DataAttribute (i.e., the logical names of the columns in the physical tables). In addition, the attributes can be further characterized to reflect the data types used in a given architecture domain's system using ObjectByReference corresponding to the CADM v1.03 structure *DataDomain* (an implicit subtype of InformationAsset with typeCode = 3 [DATA DOMAIN]). Structural business rules are captured by building entity associations with instances of ObjectByReference corresponding to the CADM v1.03 structure *DataEntityRelationship*. The subviews of a ConceptualModel are generated through the relation of the pertinent instances of DataEntity to ConceptualDataModelView.

4.6.9.3 CADM v1.5 Instantiation

The instantiation of OV-7 as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E679 [OBR]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	OV-7	4[ArchElem]
127	1	ArchitectureDocument (OV-7 in Project X Architecture)	3[OVA]
522	1	Architecture is documented by OV-7	3[OVA]
523	1	OV-7 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The next step is to relate the OV-7 to the instances of InformationAsset that corresponds to the actual content of the model. The pertinent subtypes are DataEntity, DataAttribute, and ConceptualDataModel. To represent the example shown in **Figure 4-27,** we need to instantiate the following entries:

1. Six instances of DataEntity
2. 15 instances of DataAttribute
3. One instance of ConceptualDataModel

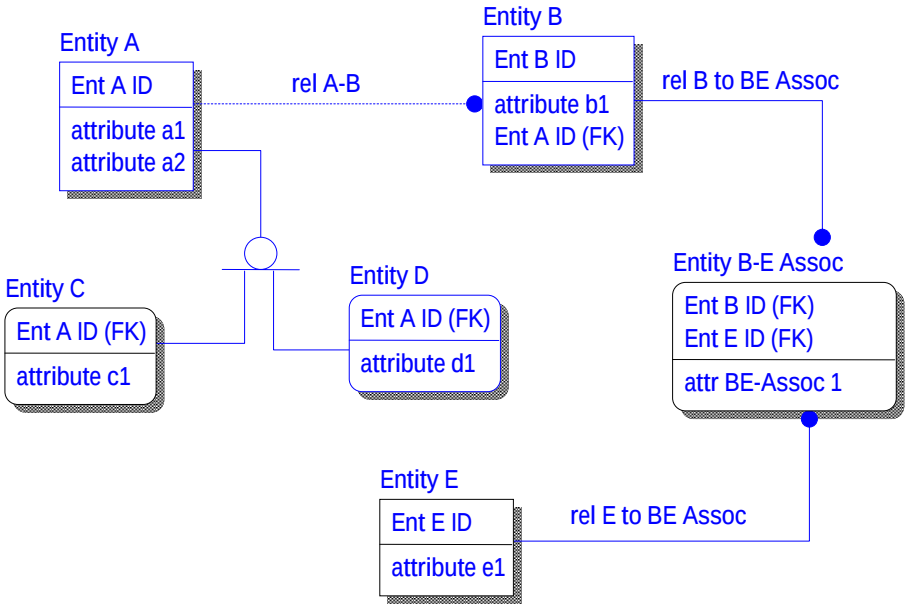


Figure 4-27: OV-7 – Template

The instance tables below show how this is expressed in CADM v1.5.

Object

objectIdentifier	pointerCode
214	E112[ConceptualModel]
215	E215[ConceptualDataModelView]
216	E679 [OBR for ConceptualDataModelViewDataEntity]
217	E679 [OBR for ConceptualDataModelViewDataEntity]
218	E679 [OBR for ConceptualDataModelViewDataEntity]
219	E679 [OBR for ConceptualDataModelViewDataEntity]
220	E679 [OBR for ConceptualDataModelViewDataEntity]
221	E679 [OBR for ConceptualDataModelViewDataEntity]
315	E133[DataEntity]
316	E133[DataEntity]
317	E133[DataEntity]
318	E133[DataEntity]
319	E133[DataEntity]
320	E133[DataEntity]
321	E118[DataAttribute]
322	E118[DataAttribute]
323	E118[DataAttribute]
324	E118[DataAttribute]
325	E118[DataAttribute]
326	E118[DataAttribute]
327	E118[DataAttribute]
328	E118[DataAttribute]
329	E118[DataAttribute]

objectIdentifier	pointerCode
330	E118[DataAttribute]
331	E118[DataAttribute]
332	E118[DataAttribute]
333	E118[DataAttribute]
334	E118[DataAttribute]
335	E118[DataAttribute]
622	E678[OVA]
623	E678[OVA]
624	E678[OVA]
625	E678[OVA]
626	E678[OVA]
627	E678[OVA]
628	E678[OVA]
629	E678[OVA]
630	E678[OVA]
631	E678[OVA]
632	E678[OVA]
633	E678[OVA]
634	E678[OVA]
635	E678[OVA]
636	E678[OVA]
637	E678[OVA]
638	E678[OVA]
639	E678[OVA]
640	E678[OVA]
641	E678[OVA]
642	E678[OVA]
643	E678[OVA]
644	E678[OVA]
645	E678[OVA]
646	E678[OVA]
647	E678[OVA]
648	E678[OVA]
649	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
214	1	Logical Data Model OV-7 Template	4[ArchElem]
215	1	Logical Data Model OV-7 View 1	4[ArchElem]
216	1	CDMVDE01[Entity A in OV-7(215)]	5[OBR]
217	1	CDMVDE02[Entity B in OV-7(215)]	5[OBR]
218	1	CDMVDE01[Entity C in OV-7(215)]	5[OBR]
219	1	CDMVDE01[Entity D in OV-7(215)]	5[OBR]
220	1	CDMVDE01[Entity B-E in OV-7(215)]	5[OBR]
221	1	CDMVDE01[Entity E in OV-7(215)]	5[OBR]
315	1	Entity A	4[ArchElem]
316	1	Entity B	4[ArchElem]
317	1	Entity C	4[ArchElem]
318	1	Entity D	4[ArchElem]
319	1	Entity B-E Assoc	4[ArchElem]
320	1	Entity E	4[ArchElem]
321	1	Ent A ID	4[ArchElem]
322	1	Attribute a1	4[ArchElem]

*Identifier	*Index	name	categoryCode
323	1	Attribute a2	4[ArchElem]
324	1	Ent B ID	4[ArchElem]
325	1	Attribute b1	4[ArchElem]
326	1	Ent A ID (FK)	4[ArchElem]
327	1	Ent A ID (FK)	4[ArchElem]
328	1	Attribute c1	4[ArchElem]
329	1	Ent A ID (FK)	4[ArchElem]
330	1	Attribute d1	4[ArchElem]
331	1	Ent B ID (FK)	4[ArchElem]
332	1	Ent E ID (FK)	4[ArchElem]
333	1	Attr BE-Assoc 1	4[ArchElem]
334	1	Ent E ID	4[ArchElem]
335	1	Attribute e1	4[ArchElem]
622	1	OV-7 View 1 is in OV-7 Template	3[OVA]
623	1	CDMVDE01 is in OV-7 View1[215]	3[OVA]
624	1	Entity A is part of CDMVDE01	3[OVA]
625	1	Ent A ID is part of Entity A	3[OVA]
626	1	Attribute a1 is part of Entity A	3[OVA]
627	1	Attribute a2 is part of Entity A	3[OVA]
628	1	CDMVDE02 is in OV-7[215]	3[OVA]
629	1	Entity B is part of CDMVDE02	3[OVA]
630	1	Ent B ID ia part of Entity B	3[OVA]
631	1	Attribute b1 is part of Entity B	3[OVA]
632	1	Ent A ID (FK) is part of Entity B	3[OVA]
633	1	CDMVDE03 is in OV-7[215]	3[OVA]
634	1	Entity C is part of CDMVDE03	3[OVA]
635	1	Ent A ID (FK) is part of Entity C	3[OVA]
636	1	Attribute c1 is part of Entity C	3[OVA]
637	1	CDMVDE04 is in OV-7[215]	3[OVA]
638	1	Entity D is part of CDMVDE04	3[OVA]
639	1	Ent A ID (FK) is part of Entity D	3[OVA]
640		Attribute d1 is part of Entity D	3[OVA]
641	1	CDMVDE05 is in OV-7[215]	3[OVA]
642	1	Entity B-E Assoc is part of CDMVDE05	3[OVA]
643	1	Ent B ID (FK) is part of Entity B-E Assoc	3[OVA]
644	1	Ent E ID (FK) is part of Entity B-E Assoc	3[OVA]
645	1	Attr BE-Assoc 1 is part of Entity B-E Assoc	3[OVA]
646	1	CDMVDE06 is in OV-7[214]	3[OVA]
647	1	Entity E is part of CDMVDE06	3[OVA]
648	1	Ent E ID is part of Entity E	3[OVA]
649	1	Attribute e1is part of Entity E	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
622	1	215	1	214	1	999	E112-R-E113
623	1	216	1	215	1	999	E113-R-E114
624	1	216	1	315	1	999	E133-R-E114
625	1	321	1	315	1	999	E133-R-E118
626	1	322	1	315	1	999	E133-R-E118
627	1	323	1	315	1	999	E133-R-E118

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
628	1	217	1	215	1	999	E113-R-E114
629	1	217	1	316	1	999	E133-R-E114
630	1	324	1	316	1	999	E133-R-E118
631	1	325	1	316	1	999	E133-R-E118
632	1	326	1	316	1	999	E133-R-E118
633	1	218	1	215	1	999	E113-R-E114
634	1	218	1	317	1	999	E133-R-E114
635	1	327	1	317	1	999	E133-R-E118
636	1	328	1	317	1	999	E133-R-E118
637	1	219	1	215	1	999	E113-R-E114
638	1	219	1	318	1	999	E133-R-E114
639	1	329	1	318	1	999	E133-R-E118
640	1	330	1	318	1	999	E133-R-E118
641	1	220	1	215	1	999	E113-R-E114
642	1	220	1	319	1	999	E133-R-E114
643	1	331	1	319	1	999	E133-R-E118
644	1	332	1	319	1	999	E133-R-E118
645	1	333	1	319	1	999	E133-R-E118
646	1	221	1	215	1	999	E113-R-E114
647	1	221	1	320	1	999	E133-R-E114
648	1	334	1	320	1	999	E133-R-E118
649	1	335	1	320	1	999	E133-R-E118

The relationTypeCode values used have the following meanings:

E112-R-E113 = is represented in
E113-R-E114 = displays
E133-R-E114 = is displayed in
E133-R-E118 = is described by

4.6.9.4 Net-Centric Requirements

The specification of discovery metadata at the operational level can be expressed in CADM v1.5 through DiscoveryMetadata, which is linkable to Document through instances of ObjectByReference corresponding to the CADM v1.03 entity *DocumentDiscoveryMetadata*. The latter is linkable to InformationAsset through instances of ObjectByReference corresponding to the CADM v1.03 entity *InformationAssetDocument*. To do that, one needs to create the respective instance of *DocumentDiscoveryMetadata* and create a link from DiscoveryMetadata in the ObjectVersionAssociation table with the relationTypeCode = E147-R-E152 (is used to discover). Next, a link can be created from *DocumentDiscoveryMetadata* to *InformationAssetDocument* in the ObjectVersionAssociation table with the relationTypeCode = E152-R-E217 (may apply to). The latter can be then related to the respective instance of InformationAsset in the ObjectVersionAssociation table with the relationTypeCode = E215-R-E217 (is cited in).

4.6.10 CADM v1.5 Support for Systems and Services Interface Description (SV-1)

4.6.10.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-1 depicts systems nodes and the systems resident at these nodes to support organizations/human roles represented by operational nodes of the OV-2. SV-1 also identifies the interfaces between systems and systems at nodes.

4.6.10.2 High-Level Description

Figure 4-28 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-1.

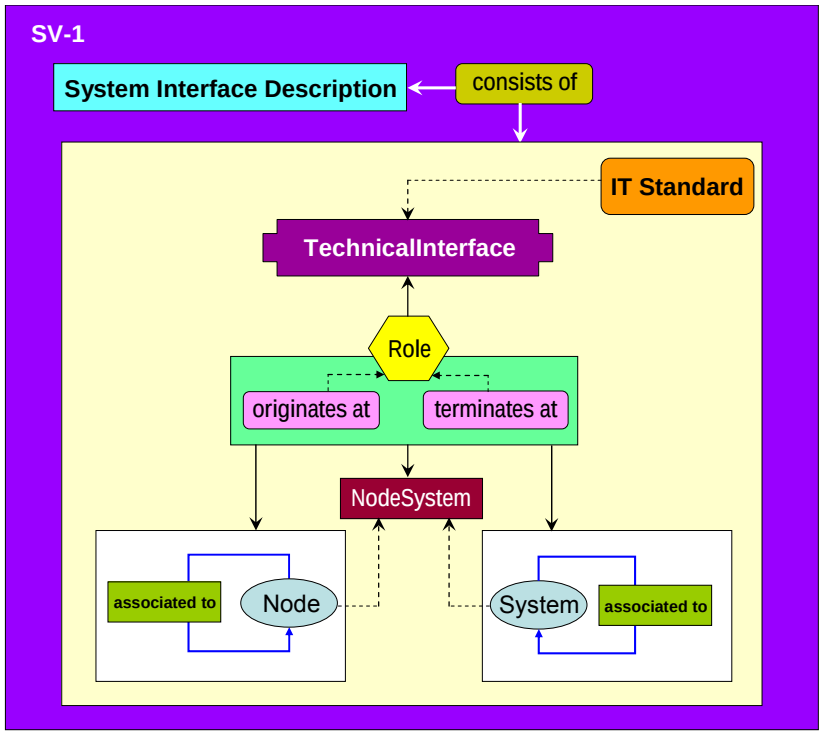


Figure 4-28: High-Level Depiction of CADM v1.5 Data Structures for SV-1 Representation

The DoDAF architecture product SV-1 as an architecture product is expressed in CADM v1.5 as an instance of Document. This instance can be connected to the appropriate instance of Architecture that it is part of. The instance of Document links to the actual data content of the SV-1 through one or more instances of the CADM v1.03 entity *SystemInterfaceDescriptionElement* (modeled via ObjectByReference).

The function of these instances of *SystemInterfaceDescriptionElement* is to collect all the pertinent instances of TechnicalInterface that make the SV-1. The TechnicalInterface is the CADM v1.5 entity that serves as the focus for the specifications of the logical *interfaces*. System associations can be linked to TechnicalInterface as well as node associations and the instances of *system* at a *node* that either send or receive information **Figure 4-29**. Each TechnicalInterface can also be associated to InformationTechnologyStandard (a subtype of Agreement), which has a number of subtypes that allow the specification of applicable standards for the TechnicalInterface. Where the means for transfer are known, they can be expressed as instances of CommunicationMedium, which, in turn, can be linked to TechnicalInterface.

4.6.10.3 CADM v1.5 Instantiation

The figure below shows an example of what systems interface description may contain.

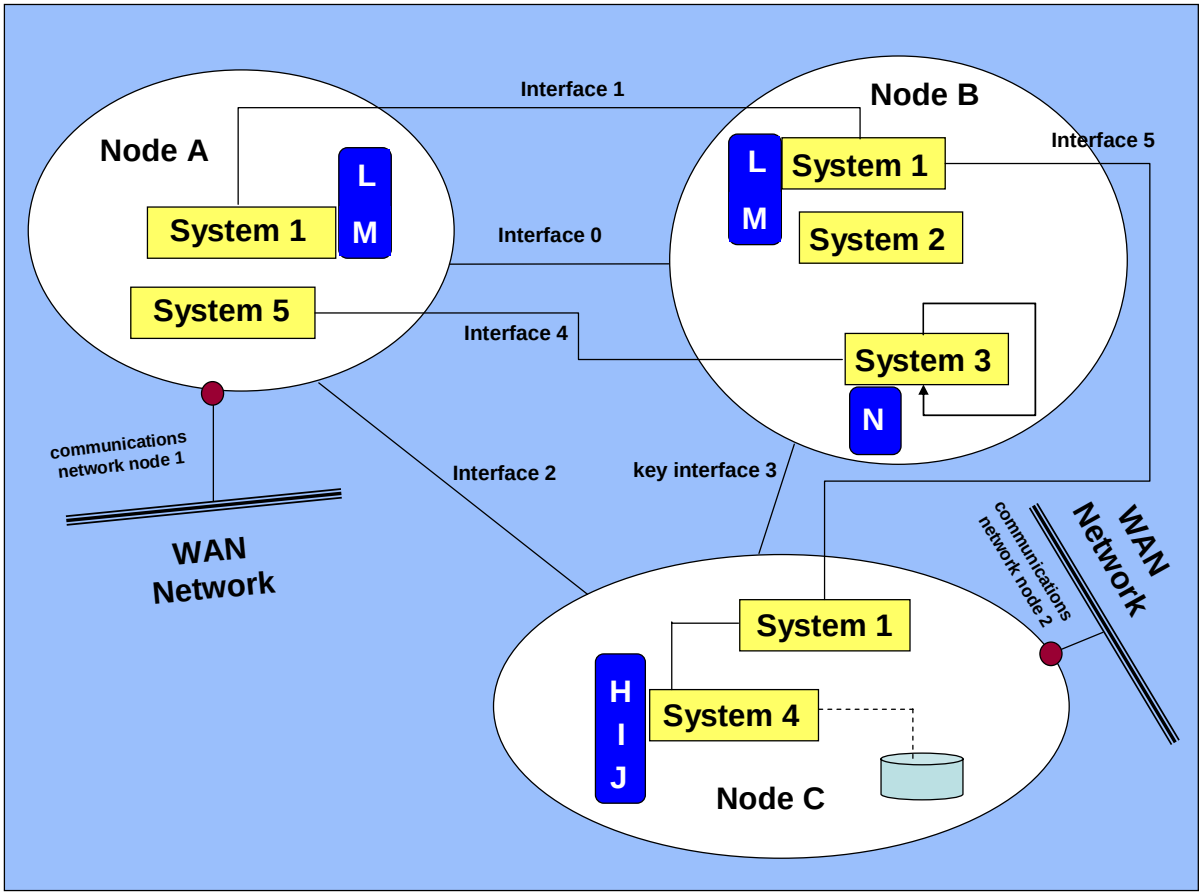


Figure 4-299: SV-1 Showing Node Edge to Node Edge and Systems-Systems Interface Example

The instantiation of SV-1 as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
11141	E148[Document]
127	E679[OBR]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
11141	1	SV-1	4[ArchElem]
127	1	ArchitectureDocument (SV-1 in Project X Architecture)	5[OBR]
522	1	Architecture is documented by SV-1	3[OVA]
523	1	SV-1 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	11141	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The SV-1 is linked to the actual content of the product through the CADM v1.03 entity *SystemInterfaceDescriptionElement*.

The representation of the data for an SV-1 in CADM v1.5 requires as the first step the specification of the systems involved. The entries corresponding to the notional example are given below.

Object

objectIdentifier	pointerCode
7501	E563 [System]
7502	E563 [System]
7503	E563 [System]
7504	E563 [System]
7505	E563 [System]
7506	E563 [System]

ObjectVersion

*Identifier	*Index	name	categoryCode
7501	1	System 1	4 [AE]
7502	1	System 2	4 [AE]
7503	1	System 3	4 [AE]
7504	1	System 4	4 [AE]
7505	1	System 5	4 [AE]
7506	1	Shared Database G	4 [AE]

In the example, there are two system associations. The entries in the ObjectVersionAssociation table are shown below (the instances for the corresponding Object and ObjectVersion are not shown).

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
125001	1	7501	1	7504	1	465	NULL
125002	1	7504	1	7506	1	465	NULL

The value categoryCode = 465 [*SystemAssociation*]

According to the figure for the notional example, there is one Network specified for this SV-1. The tables below show the instantiation for this portion of the product.

Object

objectIdentifier	pointerCode
107	E333[Network]

ObjectVersion

*Identifier	*Index	name	categoryCode
107	1	Fire Direction Net	4 [AE]

There are five intances of Node.

Object

objectIdentifier	pointerCode
15101	E359[Node]
15102	E359[Node]
15103	E359[Node]
15104	E359[Node]
15105	E359[Node]

ObjectVersion

*Identifier	*Index	Name	categoryCode
15101	1	Node A	4 [AE]
15102	1	Node B	4 [AE]
15103	1	Node C	4 [AE]
15104	1	Comm Network Node 1	4 [AE]
15105	1	Comm Network Node 2	4 [AE]

There are six instances of *NodeAssociation* among these five nodes. The entries in the ObjectVersionAssociation table are shown below the Object and ObjectVersion tables.

Object

objectIdentifier	pointerCode
115101	E678[OVA]
115102	E678[OVA]
115103	E678[OVA]
115104	E678[OVA]
115105	E678[OVA]
115106	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
115101	1	Node A – Node B	3[OVA]
115102	1	Node A – Node C	3[OVA]
115103	1	Node B – Node C	3[OVA]
115104	1	Node C – Node C	3[OVA]
115105	1	Node A – Comm Network Node 1	3[OVA]
115106	1	Node C – Comm Network Node 2	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
115101	1	15101	1	15102	1	362	NULL
115102	1	15101	1	15103	1	362	NULL
115103	1	15102	1	15103	1	362	NULL
115104	1	15103	1	15103	1	362	NULL
115105	1	15101	1	15104	1	362	NULL
115106	1	15103	1	15105	1	362	NULL

The value categoryCode = E362 [*NodeAssociation*]

There are two instances of *NetworkNodeAssociation*. The entries in the tables below show the corresponding instantiation.

Object

objectIdentifier	pointerCode
35101	E678[OVA]
35102	E678[OVA]
35103	E678[OVA]
35104	E678[OVA]
35121	E679[OBR]
35122	E679[OBR]

ObjectVersion

*Identifier	*Index	Name	categoryCode
35101	1	Network – Connection 1	3[OVA]
35102	1	Network – Connection 1	3[OVA]
35103	1	Network – Connection 2	3[OVA]
35104	1	Network – Connection 2	3[OVA]
35121	1	Network – Connection 1	5[OBR]
35122	1	Network – Connection 2	5[OBR]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
35101	1	107	1	35121	1	999	E333-R-E350
35102	1	15104	1	35121	1	999	E359-R-E350
35103	1	107	1	35122	1	999	E333-R-E350
35104	1	15105	1	35122	1	999	E359-R-E350

The relationTypeCode values used have the following meanings:

E333-R-E350 = [Network] has as a participant [*NetworkNode*]
E359-R-E350 = [Node] participates in [*NetworkNode*]

The nodes are associated with the systems as follows.

Object

objectIdentifier	pointerCode
25101	E678[OVA]
25102	E678[OVA]
25103	E678[OVA]
25104	E678[OVA]
25105	E678[OVA]
25106	E678[OVA]
25107	E678[OVA]
25108	E678[OVA]
25109	E678[OVA]
25110	E678[OVA]
25111	E678[OVA]
25112	E678[OVA]
25113	E678[OVA]
25114	E678[OVA]
25121	E679[OBR]
25122	E679[OBR]
25123	E679[OBR]
25124	E679[OBR]
25125	E679[OBR]
25126	E679[OBR]
25127	E679[OBR]

ObjectVersion

*Identifier	*Index	Name	categoryCode
25101	1	Node A System 1	3[OVA]
25102	1	Node A System 1	3[OVA]
25103	1	Node A System 5	3[OVA]
25104	1	Node A System 5	3[OVA]
25105	1	Node B System 1	3[OVA]
25106	1	Node B System 1	3[OVA]
25107	1	Node B System 2	3[OVA]
25108	1	Node B System 2	3[OVA]
25109	1	Node B System 3	3[OVA]
25110	1	Node B System 3	3[OVA]
25111	1	Node C System 1	3[OVA]
25112	1	Node C System 1	3[OVA]
25113	1	Node C System 4	3[OVA]
25114	1	Node C System 4	3[OVA]
25121	1	Node A System 1	5[OBR]
25122	1	Node A System 5	5[OBR]
25123	1	Node B System 1	5[OBR]
25124	1	Node B System 2	5[OBR]
25125	1	Node B System 3	5[OBR]
25126	1	Node C System 1	5[OBR]
25127	1	Node C System 4	5[OBR]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
25101	1	15101	1	25121	1	999	E359-R-E396
25102	1	7501	1	25121	1	999	E563-R-E396
25103	1	15101	1	25122	1	999	E359-R-E396
25104	1	7505	1	25122	1	999	E563-R-E396
25105	1	15102	1	25123	1	999	E359-R-E396
25106	1	7501	1	25123	1	999	E563-R-E396
25107	1	15102	1	25124	1	999	E359-R-E396
25108	1	7502	1	25124	1	999	E563-R-E396
25109	1	15102	1	25125	1	999	E359-R-E396
25110	1	7503	1	25125	1	999	E563-R-E396
25111	1	15103	1	25126	1	999	E359-R-E396
25112	1	7501	1	25126	1	999	E563-R-E396
25113	1	15104	1	25127	1	999	E359-R-E396
25114	1	7504	1	25127	1	999	E563-R-E396

The relationTypeCode values used have the following meanings:

E359-R-E396 = [Node] is supported by [*NodeSystem*]
E563-R-E396 = [System] supports the functions of [*NodeSystem*]

There are six instances of *SystemFunction*, captured in CADM v1.5 as instances of ProcessActivity.

Object

objectIdentifier	pointerCode
1201	E678[OVA]
1202	E678[OVA]
1203	E678[OVA]
1204	E678[OVA]
1205	E678[OVA]
1206	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
1201	1	System Function H	3[OVA]
1202	1	System Function I	3[OVA]
1203	1	System Function J	3[OVA]
1204	1	System Function L	3[OVA]
1205	1	System Function M	3[OVA]
1206	1	System Function N	3[OVA]

There are six associations among the Systems and ProcessActivities.

Object

objectIdentifier	pointerCode
45101	E678[OVA]
45102	E678[OVA]
45103	E678[OVA]
45104	E678[OVA]
45105	E678[OVA]
45106	E678[OVA]
45107	E678[OVA]
45108	E678[OVA]
45109	E678[OVA]
45110	E678[OVA]
45111	E678[OVA]
45112	E678[OVA]
45121	E679[OBR]
45122	E679[OBR]
45123	E679[OBR]
45124	E679[OBR]
45125	E679[OBR]
45126	E679[OBR]

ObjectVersion

*Identifier	*Index	Name	categoryCode
45101	1	Node A System 1	3[OVA]
45102	1	Node A System 1	3[OVA]
45103	1	Node A System 5	3[OVA]
45104	1	Node A System 5	3[OVA]
45105	1	Node B System 1	3[OVA]
45106	1	Node B System 1	3[OVA]
45107	1	Node B System 2	3[OVA]
45108	1	Node B System 2	3[OVA]
45109	1	Node B System 3	3[OVA]
45110	1	Node B System 3	3[OVA]
45111	1	Node C System 1	3[OVA]
45112	1	Node C System 1	3[OVA]
45121	1	System 1 Function L	5[OBR]
45122	1	System 1 Function M	5[OBR]
45123	1	System 3 Function N	5[OBR]
45124	1	System 4 Function H	5[OBR]
45125	1	System 4 Function I	5[OBR]
45126	1	System 4 Function J	5[OBR]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
45101	1	7501	1	45121	1	999	E563-R-E597
45102	1	1204	1	45121	1	999	E486-R-E597
45103	1	7501	1	45122	1	999	E563-R-E597
45104	1	1205	1	45122	1	999	E486-R-E597
45105	1	7503	1	45123	1	999	E563-R-E597
45106	1	1206	1	45123	1	999	E486-R-E597
45107	1	7504	1	45124	1	999	E563-R-E597
45108	1	1201	1	45124	1	999	E486-R-E597
45109	1	7504	1	45125	1	999	E563-R-E597
45110	1	1202	1	45125	1	999	E486-R-E597
45111	1	7504	1	45126	1	999	E563-R-E597
45112	1	1203	1	45126	1	999	E486-R-E597

The relationTypeCode values used have the following meanings:

E486-R-E597 = [ProcessActivity] is supported by [*SystemProcessActivity*]
E563-R-E597 = [System] is designed to provide [*SystemProcessActivity*]

There are eight instances of TechnicalInterface. Their instantiation is shown in the following tables.

Object

objectIdentifier	pointerCode
10701	E636[TechnicalInterface]
10702	E636[TechnicalInterface]
10703	E636[TechnicalInterface]
10704	E636[TechnicalInterface]
10705	E636[TechnicalInterface]
10706	E636[TechnicalInterface]
10707	E636[TechnicalInterface]
10708	E636[TechnicalInterface]

ObjectVersion

*Identifier	*Index	name	categoryCode
10701	1	I/F 0 (A-B)	4 [AE]
10702	1	I/F 2 (A-C)	4 [AE]
10703	1	Key I/F 3 (B-C)	4 [AE]
10704	1	I/F 1 (A1-B1)	4 [AE]
10705	1	I/F 4 (A5-B3)	4 [AE]
10706	1	Key I/F 3 (B1-C1)	4 [AE]
10707	1	I/F 5 (B3-C4)	4 [AE]
10708	1	Key I/F 6 (C1-C4)	4 [AE]

There are four instances of *TechnicalInterfaceAssociation*. The entries in the ObjectVersionAssociation table are shown below the Object and ObjectVersion tables.

Object

objectIdentifier	pointerCode
515101	E678[OVA]
515102	E678[OVA]
515103	E678[OVA]
515104	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
515101	1	I/F 0 (A-B) to I/F 1 (A1-B1)	3[OVA]
515102	1	I/F 0 (A-B) to I/F 4 (A5-B3)	3[OVA]
515103	1	Key I/F 3 (B-C) to Key I/F 3 (B1-C1)	3[OVA]
515104	1	Key I/F 3 (B-C) to I/F 5 (B3-C4)	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
515101	1	10701	1	10704	1	E637	NULL
515102	1	10701	1	10705	1	E637	NULL
515103	1	10703	1	10706	1	E637	NULL
515104	1	10703	1	10707	1	E637	NULL

The value categoryCode = E637 [*TechnicalInterfaceAssociation*]

The SV-1 consists of several *SystemInterfaceDescriptionElements* that are linked to *TechnicalInterfaces*.

After everything has been defined and associated above, one can instantiate the rows corresponding to the CADM entity *SystemInterfaceDescriptionElement*. The tables below show how this is done for the instances of *SystemInterfaceDescriptionElement* using the instances of TechnicalInterface already created.

Object

objectIdentifier	pointerCode
201	E678[OVA]
202	E678[OVA]
203	E678[OVA]
204	E678[OVA]
205	E678[OVA]
206	E678[OVA]
207	E678[OVA]
208	E678[OVA]
301	E679[OBR]
302	E679[OBR]
303	E679[OBR]
304	E679[OBR]
305	E679[OBR]
306	E679[OBR]
307	E679[OBR]
308	E679[OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
301	1	SIDE 1	5[OBR]
302	1	SIDE 2	5[OBR]
303	1	SIDE 3	5[OBR]
304	1	SIDE 4	5[OBR]
305	1	SIDE 5	5[OBR]
306	1	SIDE 6	5[OBR]
307	1	SIDE 7	5[OBR]
308	1	SIDE 8	

ObjectByReference

*Identifier	*Index	categoryCode
301	1	E587[SystemInterfaceDescriptionElement]
302	1	E587[SystemInterfaceDescriptionElement]
303	1	E587[SystemInterfaceDescriptionElement]
304	1	E587[SystemInterfaceDescriptionElement]
305	1	E587[SystemInterfaceDescriptionElement]
306	1	E587[SystemInterfaceDescriptionElement]
307	1	E587[SystemInterfaceDescriptionElement]
308	1	E587[SystemInterfaceDescriptionElement]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
201	1	10701	1	301	1	999	E636-R-E587
202	1	10702	1	302	1	999	E636-R-E587
203	1	10703	1	303	1	999	E636-R-E587
204	1	10704	1	304	1	999	E636-R-E587
205	1	10705	1	305	1	999	E636-R-E587
206	1	10706	1	306	1	999	E636-R-E587
207	1	10707	1	307	1	999	E636-R-E587
208	1	10708	1	308	1	999	E636-R-E587

The relationTypeCode values used have the following meanings:

E636-R-E587 = [TechnicalInterface] is cited for [*SystemInterfaceDescriptionElement*]

4.6.10.4 Net-Centric Requirements for SV-1

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to SoftwareType through instances of ObjectByReference corresponding to *SoaServiceSoftwareType*. The applicable codes in the ObjectVersionAssociation table are:

E682-R-E686 = [SoaService] uses [*SoaServiceSoftwareType*]

E544-R-E686 = **[**SoftwareType**]** is used in [*SoaServiceSoftwareType*].

4.6.11 CADM v1.5 Support for Systems and Services Communications Description (SV-2)

4.6.11.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-2 depicts pertinent information about communications systems, communications links, and communications networks. SV-2 documents the kinds of communications media that support the systems and implement their interfaces as described in SV-1. Thus, SV-2 shows the communications details of SV-1 interfaces that automate aspects of the needlines represented in OV-2.

4.6.11.2 High-Level Description

Figure 4-30 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-2.

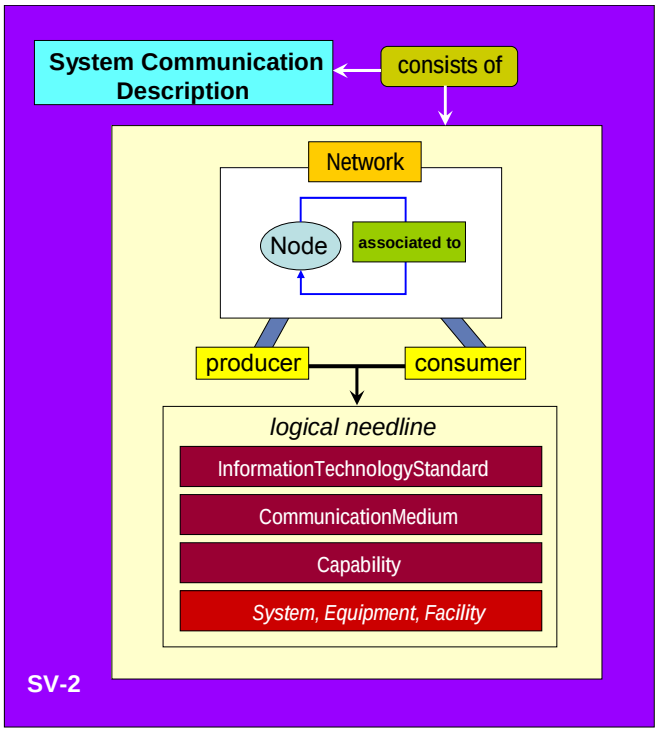


Figure 4-30: High-Level Depiction of CADM v1.5 Data Structures for SV-2 Representation

The DoDAF architecture product SV-2 as an architecture product is expressed in CADM v1.5 as an instance of Document. This instance can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference). The instance of Document is linked to the actual **data content** of the SV-2 through one or more instances of Network. Applicable systems can be related to Network via the CADM v1.03 entity *NetworkSystem* (modeled via ObjectByReference). Similarly, the communication media and the communication systems employed can be related to Network. The composition of the Network is done through pairwise node associations. The CADM v1.03 entity *NodeLink* (a subtype of *NodeAssociation*) is used for that purpose. For each *NodeLink*, one can specify the applicable CommunicationMedium and InformationTechnologyStandard, as well as the specific Capability.

4.6.11.3 CADM v1.5 Instantiation

The **Figure 31** below shows a notional example of an SV-2 product, where nodes are linked to other nodes through specific communication interfaces.

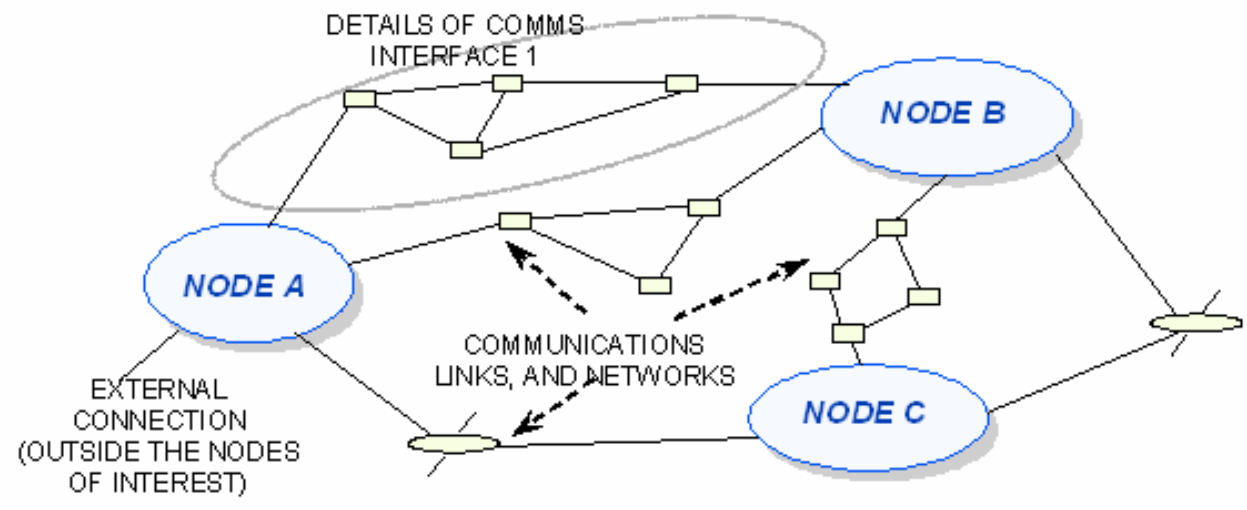


Figure 4-31: Systems Communications Description Example

The SV-2 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
8115	E038[Architecture]
8116	E148[Document]
8117	E679[OBR]
8522	E678[OVA]
8523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
8115	1	Program Architecture C08	4[ArchElem]
8116	1	Notional Systems Communications Description	4[ArchElem]
8117	1	ArchitectureDocument (SV-2 in Program Architecture C08)	5[OBR]
8522	1	Architecture is documented by SV-2	3[OVA]
8523	1	SV-2 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
8117	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
8522	1	8115	1	8117	1	999	E038-R-E045
8523	1	8116	1	8117	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

As with the OV-2 product discussed in the preceding sections for the content of the SV-2, one instantiates Network and the nodes that it comprises. The instance tables below show how this is done in CADM v1.5.

Object

objectIdentifier	pointerCode
207	E333[Network]
217	E359[Node]
218	E359[Node]
219	E359[Node]
247	E679[OBR]
248	E679[OBR]
249	E679[OBR]
524	E678[OVA]
525	E678[OVA]
526	E678[OVA]
527	E678[OVA]
528	E678[OVA]
529	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
207	1	SV-2 Level 1 Decomposition	4[ArchElem]
217	1	Node A	4[ArchElem]
218	1	Node B	4[ArchElem]
219	1	Node C	4[ArchElem]
247	1	NetworkNodeA (Node A in SV-2[116])	5[OBR]
248	1	NetworkNodeB (Node B in SV-2[116])	5[OBR]
249	1	NetworkNodeC (Node C in SV-2[116])	5[OBR]
524	1	NetworkNodeA[247] is part of Network	3[OVA]
525	1	NodeA[217] is part of NetworkNodeA	3[OVA]
526	1	NetworkNodeB[248] is part of Network	3[OVA]
527	1	NodeB[218] is part of NetworkNodeB	3[OVA]
528	1	NetworkNodeC[249] is part of Network	3[OVA]
529	1	NodeB[219] is part of NetworkNodeC	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
247	1	E350[NetworkNode]
248	1	E350[NetworkNode]
249	1	E350[NetworkNode]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
524	1	207	1	247	1	999	E333-R-E350
525	1	217	1	247	1	999	E359-R-E350
526	1	207	1	248	1	999	E333-R-E350
527	1	218	1	248	1	999	E359-R-E350
528	1	207	1	249	1	999	E333-R-E350
529	1	219	1	249	1	999	E359-R-E350

The relationTypeCode values used have the following meanings:

E359-R-E350 = participates in
E333-R-E350 = has as a participant

The relationships among nodes are done through instances of ObjectVersionAssociation corresponding to the CADM v1.03 entity *NodeAssociation*. For the node associations:

Object

objectIdentifier	pointerCode
534	E678[OVA]
535	E678[OVA]
536	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
534	1	Node A to Node B	3[OVA]
535	1	Node A to Node C	3[OVA]
536	1	Node B to Node C	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
534	1	217	1	218	1	E362	NULL
535	1	217	1	219	1	E362	NULL
536	1	218	1	219	1	E362	NULL

To relate the instance of Network to the instance of Document:

Object

objectIdentifier	pointerCode
563	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
563	1	Network to SV-2 Document	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
563	1	207	1	116	1	999	E333-R-E369

The relationTypeCode values used have the following meanings:

E333-R-E369 = is used to specify

Associate *NodeAssociations* and *NodeLink*. For this example, the tables below show an example of the association between *NodeAssociation* (Node A to Node B) and *NodeLink* and the association of *NodeLink* and CommunicationMedium through the characterization of *NodeLinkCommunicationMedium.*

Object

objectIdentifier	pointerCode
605	E679[OBR]
606	E679[OBR]
710	E102[CommunicationMedium]
815	E678[OVA]
816	E678[OVA]
817	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
605	1	Node Link 1	5[OBR]
606	1	NodeLinkCommunicationMedium1 (Communication Medium 1 is part of Node Link 1)	5[OBR]
710	1	Communication Medium 1	4[ArchElem]
815	1	Node A to Node B[534] to Node Link 1	3[OVA]
816	1	NodeLinkCommunication1 is part of Communication Medium 1	3[OVA]
817	1	NodeLinkCommunication1 is part of Node Link 1	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
815	1	605	1	534	1	999	E362-S-E378
816	1	606	1	710	1	999	E102-R-E381
817	1	606	1	605	1	999	E378-R-E381

The relationTypeCode values used have the following meanings:

E362-S-E378 = is a
E102-R-E381 = is a mode for
E378-R-E381 = has

4.6.11.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to EquipmentType through instances of ObjectByReference corresponding to *SoaServiceEquipmentType*. The applicable codes in the ObjectVersionAssociation table are:

E682-R-E687 = [SoaService] uses [*SoaServiceEquipmentType*]

E153-R-E687 = [EquipmentType] is used in [*SoaServiceEquipmentType*]

4.6.12 CADM v1.5 Support for Systems-Systems Matrix, Services-Systems Matrix, and Services-Services Matrix (SV-3)

4.6.12.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-3 provides detail on the interface characteristics described in SV-1 for the architecture, arranged in matrix form.

4.6.12.2 High-Level Description

Figure 4-32 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-3.

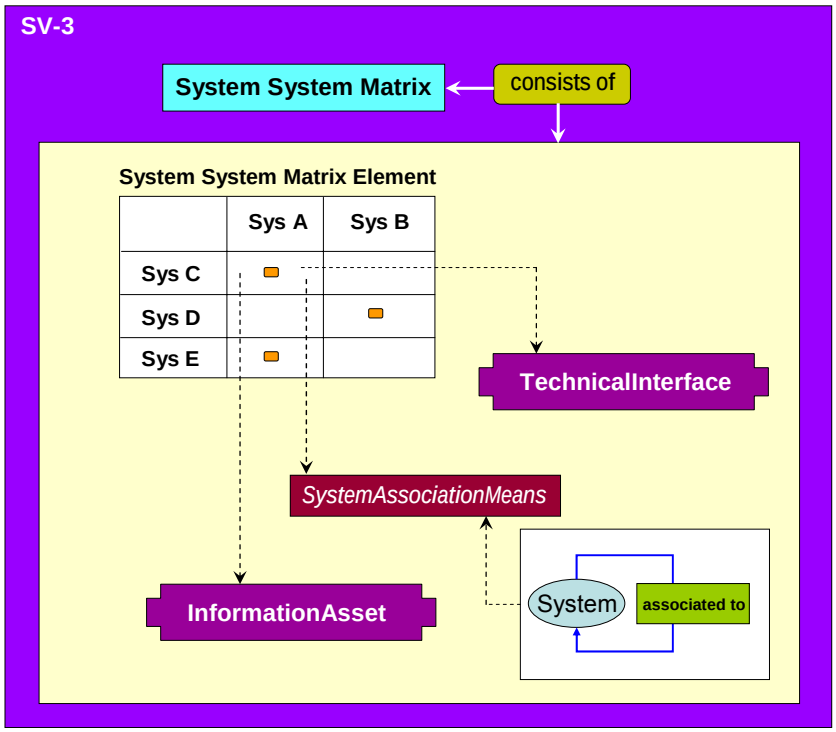


Figure 4-32: High-Level Depiction of CADM v1.5 Data Structures for SV-3 Representation

The DoDAF architecture product SV-3 is expressed in CADM v1.5 as an instance of Document. The SV-3 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference). As shown in the template provided in DoDAF v1.5 Volume II, the SV-3 is a presentation format for system-to-system associations that highlights the types of interface

characteristics for each pair of systems. All of this information is captured in CADM v1.5 through TechnicalInterface (see SV-1 above). The content of the SV-3 is expressed though instances of the CADM v1.03 entity *SystemSystemMatrixElement* that links the system pairs (modeled through *SystemAssociation* and *SystemAssociationMeans*) as well as the applicable instance of TechnicalInterface. The semantics of the "dots" in the matrix cells (see template) can also be specified in CADM v1.5 by creating a set of instances of InformationAsset (typeCode = 3 [DATA-DOMAIN], and subtypeCode = 31 [DATA-DOMAIN-LIST]) and linking the pertinent value to each *SystemSystemMatrixElement*, where applicable.

4.6.12.3 CADM v1.5 Instantiation

The following instance tables show how the SV-3 product is represented in CADM v1.5. The example follows **Figure 4-33**. For simplicity, only the data for the cells corresponding to the systems GCCS, MCS/P and ASAS are shown.

	GCCS	MCS/P	FBCB2	M1A2 SEP	M2A3	ASAS	CGS	GBCS	IMETS	IREMBAS
GCCS		●								
MCS/P	●		●	●	●	●				
FBCB2		●		●	●	●				
M1A2 SEP		●	●		●	●				
M2A3		●	●	●		●				
ASAS	●	●	●	●	●		●	●	●	●
CGS						●				
GBCS						●				
IMETS						●				
IREMBAS						●				
AFATDS	●	●	●	●	●	●				
BFIST			●	●	●					
Paladin										
FAAVS										
MLRS										
FAADC3I	●	●	●	●	●	●				
Avenger										
BSFV-E										
GBS										
CSSCS		●	●			●				
SAMS										
SAAS										
SPDS-R										
DAMMSR										
ULLS										
….										
….										

Figure 4-33: Notional Example of an SV-3 Product

The instantiation of SV-3 as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
11144	E148[Document]
129	E679[OBR]
524	E678[OVA]
525	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
125	1	Project X Architecture	4[ArchElem]
11144	1	Example SV-3	4[ArchElem]
129	1	ArchitectureDocument (SV-3 in Project X Architecture)	5[OBR]
524	1	Architecture is documented by Example SV-3	3[OVA]
525	1	Example SV-3 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
129	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
524	1	125	1	129	1	999	E038-R-E045
525	1	11144	1	129	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The following instance tables describe the GCCS, MCS/P and ASAS systems and their associations:

Object

objectIdentifier	pointerCode
3741	E563[System]
3742	E563[System]
3746	E563[System]
828	E565[SystemAssociation]
829	E565[SystemAssociation]
724	E678[OVA]
725	E678[OVA]
726	E678[OVA]
727	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
3741	1	Global Command and Control System (GCCS)	4[ArchElem]
3742	1	Maneuver Control System (MCS/P)	4[ArchElem]
3746	1	All Sources Analysis System (ASAS)	4[ArchElem]
828	1	GCCS associates with MCS/P	5[OBR]
829	1	GCCS associates with ASAS	5[OBR]
724	1	GCCS and MCS/P	3[OVA]
725	1	MCS/P and GCCS	3[OVA]
726	1	GCCS and ASAS	3[OVA]
727	1	ASAS and GCCS	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
828	1	E565[SystemAssociation]
829	1	E565[SystemAssociation]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
724	1	3741	1	828	1	E565 [SysAssn]	NULL
725	1	3742	1	828	1	E565 [SysAssn]	NULL
726	1	3741	1	829	1	E565 [SysAssn]	NULL
727	1	3746	1	829	1	E565 [SysAssn]	NULL

Once the systems have been defined and their associations established, one can proceed to specify the instances of applicable TechnicalInterface and link them to the corresponding records of *SystemSystemMatrixElement.*

Object

objectIdentifier	pointerCode
10801	E636[TechnicalInterface]
10802	E636[TechnicalInterface]
21101	E612[SystemSystemMatrixElement]
21102	E612[SystemSystemMatrixElement]
741	E678[OVA]
742	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
10801	1	GCCS-MCS/P	4[ArchElem]
10802	1	GCCS-ASAS	4[ArchElem]
21101	1	GCCS relates to MCS/P	4[ArchElem]
21102	1	GCCS relates to ASAS	4[ArchElem]
741	1	GCCS-MCS/P describes SSME	3[OVA]
742	1	GCCS-ASAS describes SSME	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
741	1	10801	1	21101	1	999	E636-R-E612
742	1	10802	1	21102	1	999	E636-R-E612

The relationTypeCode values used have the following meanings:

E636-R-E612 = describes

In CADM v1.5, it is possible to specify the semantics of the “dots” in the SV-3 matrix. This is done via the CADM v1.03 entity *DataDomainListValue*, a many-child of *DataDomainList* (a subtype of *DataDomain*, which in turn is a subtype of InformationAsset). The InformationAsset

has typeCode=3 (DataDomain). There is only one *DataDomainListValue*, since the meaning of "dot present" is simply "Yes."

Object

objectIdentifier	pointerCode
901	E215[InformationAsset]
756	E127[DataDomain]
757	E128[DataDomainList]
758	E129[DataDomainListValue]
760	E678[OVA]
761	E678[OVA]
762	E678[OVA]
763	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
901	1	GCCS-MCS/P	4[ArchElem]
756	1	Data domain values	5[OBR]
757	1	List of matrix cell values	5[OBR]
758	1	Solid Dot is in matrix	5[OBR]
760	1	InformationAsset is a DataDomain	3[OVA]
761	1	Data Domain is a DataDomainList	3[OVA]
762	1	DataDomainList contains Solid Dot	3[OVA]
763	1	SSME contains DDLV	5[OBR]

ObjectByReference

*Identifier	*Index	categoryCode
756	1	E127[DataDomain]
757	1	E128[DataDomainList]
758	1	E129[DataDomainListValue]
763	1	E613[SystemSystemMatrixElementDataDomainListValue]

ObjectByReferenceCharacterization

*OBRCIdentifier	*OBRIdentifier	*OBRIndex	categoryCode	valueText
799	758	1	E129.A01	YES (Solid dot in matrix)

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	Object OV Identifier	object OV Index	category Code	relationType Code
760	1	901	1	756	1	997	E215-S-E127
761	1	756	1	757	1	997	E127-S-E128
762	1	757	1	758	1	999	E128-R-E129
763	1	758	1	21101	1	999	E129-R-E613

The relationTypeCode values used have the following meanings:

E215-S-E127 and E127-S-E128 are both "is a"
E128-R-E129 = contains
E129-R-E613 = is used to characterize

4.6.12.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to *TechnicalInterfaceType* through instances of ObjectByReference corresponding to *SoaServiceTechnicalInterfaceType*. The applicable codes in the ObjectVersionAssociation table are:

E682-R-E688 = [SoaService] has a [*SoaServiceTechnicalInterfaceType*]

E636-R-E688 = [*TechnicalInterfaceType*] applies to [*SoaServiceTechnicalInterfaceType*]

TechnicalInterfaceType, in turn, is linkable to TechnicalInterface through the ObjectVersionAssociation table, code E643-R-E636 = [*TechnicalInterfaceType*] is the type for [TechnicalInterface].

4.6.13 CADM v1.5 Support for Systems and Services Functionality Description (SV-4a/b)

4.6.13.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-4 documents system functional hierarchies and system functions, and the system data flows between them. Although there is a correlation between OV-5 or business-process hierarchies and the system functional hierarchy of SV-4, it need not be a one-to-one mapping, hence, the need for the Operational Activity to Systems Function Traceability Matrix (SV-5), which provides that mapping.

4.6.13.2 High-Level Description

Figure 4-34 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-4

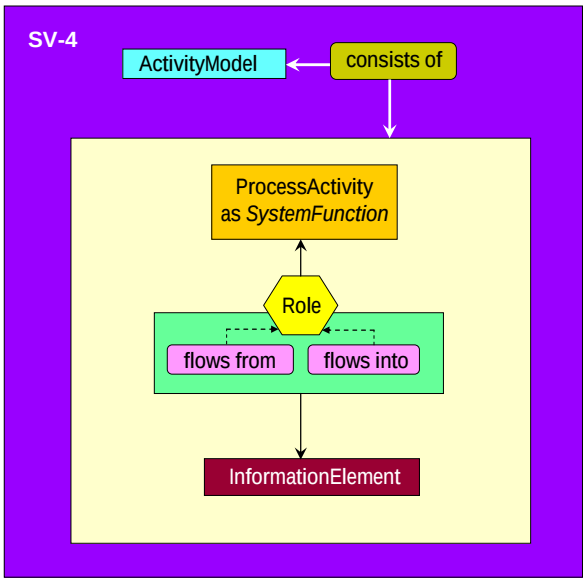


Figure 4-34: High-Level Depiction of CADM v1.5 Data Structures for SV-4 Representation

The DoDAF architecture product SV-4 is expressed in CADM v1.5 as an instance of Document. This allows the linkage to the pertinent Architecture in the manner that has been described in preceding sections. The data content of a DoDAF architecture product SV-4 as a Data Flow Diagram is expressed in CADM v1.5 as an instance of ActivityModel (a subtype of InformationAsset) that is composed of *system functions*, represented as instances of ProcessActivity. For each instance of ProcessActivity, there may be one or more instances of InformationElement. The information flows are represented as instances of InformationElement, and their roles as "inputs" or "outputs" are specified through ActivityModelInformationElementRole.

Figure 5-19 in Volume II of DoDAF v1.5 depicts the template for SV-4 products as a Data Flow Diagram. As shown Figure 5-19, there may be any number of *system functions* in an SV-4 built using this template. Each *system function* has a name, and there are flows starting at some *system function* and ending at another. As mentioned above, the representation of this data content in CADM v1.5 utilizes an instance of ActivityModel, as many instances of ProcessActivity as there are *system functions*, and as many instances of InformationElement as there are *flows* in the SV-4.

4.6.13.3 CADM v1.5 Instantiation

The SV-4 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
305	E038[Architecture]
306	E148[Document]
307	E679[OBR]
611	E678[OVA]
612	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
305	1	Program Architecture X	4[ArchElem]
306	1	Notional Data Flow Diagram	4[ArchElem]
307	1	ArchitectureDocument (SV-4 in Program Architecture[306]	5[OBR]
611	1	Architecture is documented by OV-5	3[OVA]
612	1	SV-4 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
307	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
611	1	305	1	307	1	999	E038-R-E045
612	1	306	1	307	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in

E148-R-E045 = records

The instance tables below show how CADM v1.5 captures the notional system function *System Function 1* from Figure 5-15 in DoDAF Volume II, and its associated *flows* (Flow 1 and Flow 2).

Object

objectIdentifier	pointerCode
116	E009[ActivityModel]
217	E486[ProcessActvity]
218	E486[ProcessActvity]
219	E486[ProcessActvity]
320	E221[InformationElement]
321	E221[InformationElement]
425	E010[ActivityModelInformationElementRole]
426	E010[ActivityModelInformationElementRole]
427	E010[ActivityModelInformationElementRole]
428	E010[ActivityModelInformationElementRole]
522	E678[OVA]
523	E678[OVA]
524	E678[OVA]
525	E678[OVA]
526	E678[OVA]
527	E678[OVA]
528	E678[OVA]
529	E678[OVA]
530	E678[OVA]
531	E678[OVA]
532	E678[OVA]
533	E678[OVA]
822	E022[ActivityModelProcessActivity]
823	E022[ActivityModelProcessActivity]
824	E022[ActivityModelProcessActivity]

ObjectVersion

*Identifier	*Index	name	categoryCode
116	1	SV-4 Data Flow Diagram (Template)	4[ArchElem]
217	1	System Function 1	4[ArchElem]
218	1	System Function 2	4[ArchElem]
219	1	External Source 1	4[ArchElem]
320	1	Data Flow 1	4[ArchElem]
321	1	Data Flow 2	4[ArchElem]
425	1	AMIER01 for Data Flow 1	4[ArchElem]
426	1	AMIER02 for Data Flow 1	4[ArchElem]
427	1	AMIER03 for Data Flow 2	4[ArchElem]
428	1	AMIER04 for Data Flow 2	4[ArchElem]
522	1	System Function 1 connected through AMPA01	3[OVA]
523	1	SV-4 connected through AMPA01	3[OVA]
524	1	System Function 2 connected through AMPA02	3[OVA]
525	1	SV-4 connected through AMPA02	3[OVA]
526	1	External Source 1 connected through AMPA03	3[OVA]
527	1	SV-4 connected through AMPA03	3[OVA]
528	1	AMPA[522] is part of AMIER01	3[OVA]
529	1	**Data Flow 1** in AMIER01 **starts at External Source 1**	3[OVA]
530	1	AMPA[523] is part of AMIER02	3[OVA]
531	1	**Data Flow 1** in AMIER02 **ends at System Function 1**	3[OVA]
532	1	AMPA[523] is part of AMIER03	3[OVA]
533	1	**Data Flow 2** in AMIER03 **starts at System Function 1**	3[OVA]
534	1	AMPA[524] is part of AMIER04	3[OVA]
535	1	**Data Flow 2** in AMIER04 **ends at System Function 2**	3[OVA]
822	1	AMPA01 (System Function 1 in SV-4[116])	5[OBR]
823	1	AMPA02 (System Function 2 in SV-4[116])	5[OBR]
824	1	AMPA03 (External Source 1 in SV-4[116])	5[OBR]

Next, the actual linkages are built in the **ObjectVersionAssociation** table as shown below.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	116	1	822	1	999	E009-R-E022
523	1	217	1	822	1	999	E486-R-E022
524	1	116	1	823	1	999	E009-R-E022
525	1	218	1	823	1	999	E486-R-E022
526	1	116	1	824	1	999	E009-R-E022
527	1	219	1	824	1	999	E486-R-E022
528	1	824	1	425	1	999	E022-R-E010
529	1	**320**	1	425	1	999	E221-R-E010
530	1	822	1	426	1	999	E022-R-E010
531	1	**320**	1	426	1	999	E221-R-E010
532	1	822	1	427	1	999	E022-R-E010
533	1	**321**	1	427	1	999	E221-R-E010
534	1	823	1	428	1	999	E022-R-E010
535	1	**321**	1	428	1	999	E221-R-E010

The relationTypeCode values used have the following meanings:

E009-R-E022 = includes
E486-R-E022 = is included
E022-R-E010 = defines
E221-R-E010 = is associated with

Lastly, in order to express the 'role' that each of the *flows* has with respect to the *activity* where it starts and ends, one needs to instantiate the respective ActivityModelInformationElementRole [AMIER]. For simplicity, only the attribute corresponding to the *typeCode* is shown. The tables below show the entries in ArchitectureElement and ActivityModelInformationElementRole.

ArchitectureElement

*Identifier	*Index	categoryCode
425	1	E010 [AMIER]
426	1	E010 [AMIER]
427	1	E010 [AMIER]
428	1	E010 [AMIER]

ActivityModelInformationElementRole

*Identifier	*Index	AMIER categoryCode
425	1	2[output]
426	1	1[input]
427	1	2[output]
428	1	1[input]

Retrieving the information for this segment of the SV-4 shown in Figure 5-19 of DoDAF Volume II could be accomplished by, for example, querying the database to find out all the instances of InformationElement. Once this is accomplished, the ObjectVersionAssociation table can be traversed to retrieve the related instances of ActivityModelInformationElementRole. Through this process one can retrieve the associated system functions, since each instance of ActivityModelInformationElementRole points to the instance of ObjectByReference that

corresponds to the *ActivityModelProcessActivity* [AMPA]. In the ObjectVersionAssociation table, the AMPA entries point to the corresponding instances of ActivityModel and ProcessActivity. Since filtering for just the ProcessActivity permits extraction of the name of the *system functions* and from the ActivityModelInformationElementRole, one already has the 'role' for the *data flow*.

The table below shows the final result, for each one of the *data flows*, where there are two 'roles' showing whether it is an "output" (Role = 2) or an "input" (Role =2). As can be seen, the resulting query matches the content of the SV-4 shown in Figure 5-19 in DoDAF Volume II for the two flows, *Data Flow 1* and *Data Flow 2*.

Data Flow 1 is depicted as being an *output* of system function *External Source 1* and an *input* for system function *System Function 1*. Similarly, *Flow 2* is shown as being an *output* of system function *System Function 1* and an *input* for system function *System Function 2*.

Table 4-2: Example of a CADM v1.5 Query Showing System Functions, Flows, and Their Roles for a Notional SV-4 as a Data Flow Diagram

Flow ID	Flow Index	Flow Name	Role	Function ID	System Function Index	System Function Name
320	1	Data Flow 1	2	219	1	External Source 1
320	1	Data Flow 1	1	217	1	System Function 1
321	1	Data Flow 2	2	217	1	System Function 1
321	1	Data Flow 2	1	218	1	System Funciton 2

4.6.13.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to SoaService via *SoaServiceAssociation*. The attribute E684.A01 = typeCode in the ObjectVersionAssociationCharacterization for the instances of *SoaServiceAssociation* can be set to "*sends to and receives data from.*" Next, one can link the instances of the service associations to ActivityModelInformationElementRole for the flows that have categoryCode = 1 (input) or categoryCode = 2 (output).

4.6.14 CADM v1.5 Support for Operational Activity to Systems Function Traceability Matrix, Operational Activity to Systems Traceability Matrix, Operational Activity to Services Traceability Matrix (SV-5a/b/c)

4.6.14.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-5 is a specification of the relationships between the set of operational activities applicable to an architecture and the set of system functions applicable to that architecture.

4.6.14.2 High-Level Description

Figure 4-35 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-5.

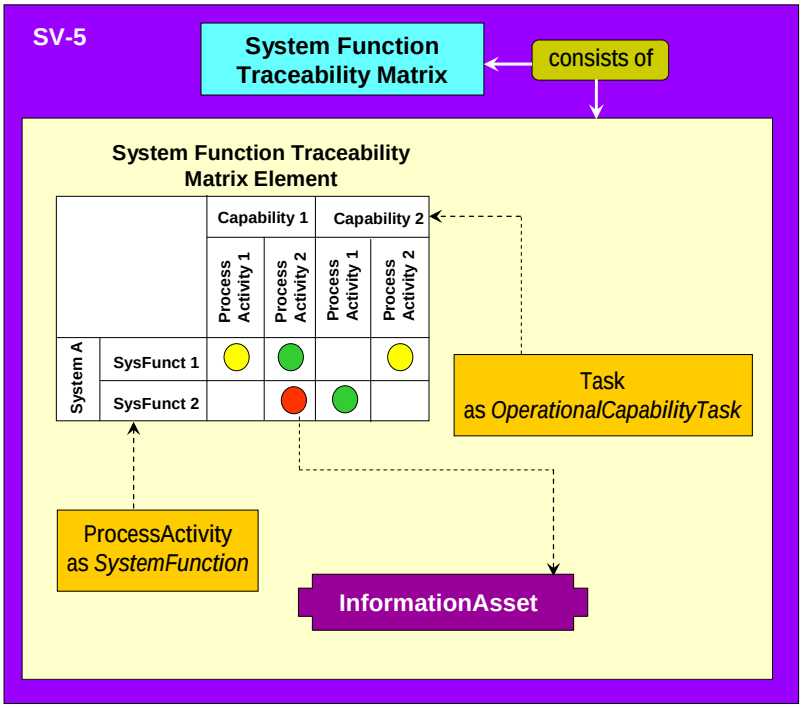


Figure 4-35: High-Level Depiction of CADM v1.5 Data Structures for SV-5 Representation

The DoDAF architecture product SV-5 is expressed in CADM v1.5 as an instance of Document. The SV-5 document can be connected to the appropriate instance of Architecture that it is part of the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference). As shown in Figure 4-35, the SV-5 is a presentation format that highlights the relationship between *system functions* and *operational activities* (both modeled in CADM v1.5 through ProcessActivity). The data content of the SV-5 is built by instantiating the CADM v1.03 entity *SystemFunctionTraceabilityMatrixElement* (through ObjecByReference) and linking it to *SystemProcessActivity* (the association between systems and system functions) and the operational activities modeled as instances of ProcessActivity.

The capabilities in the SV-5 product are understood as subtypes of Task, specifically, as instances of *OperationalCapabilityTask*, which is defined as a Task that represents the potential ability to carry out military functions that contribute to a MissionArea. Operational activities modeled as instances of ProcessActivity are related to capabilities (as defined above) through the CADM v1.03 entity *ProcessActivityTask* (modeled as ObjecByReference).

4.6.14.3 CADM v1.5 Instantiation

Figure 4-36 shows a notional template for the SV-5 product. For simplicity, only one system with two system functions, and two *operational capabilities* with two *operational activities* each, are shown.

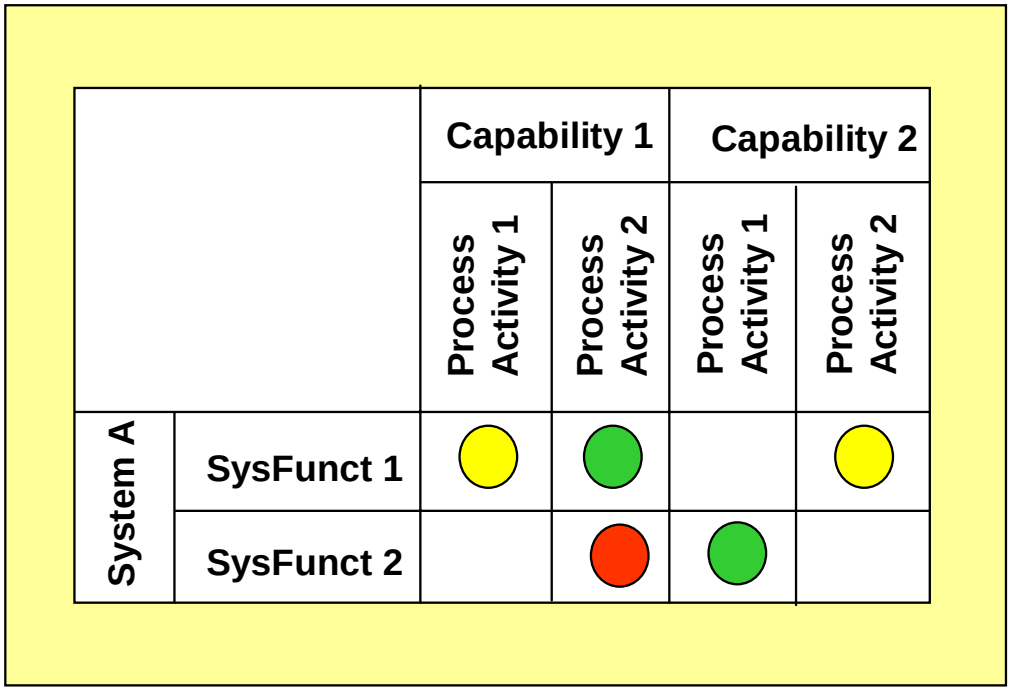


Figure 4-36: Notional SV-5 Template (Partial View)

The instantiation of SV-5 as **Document** and its relation to an appropriate instance of **Architecture** is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
11147	E148[Document]
229	E679[OBR]
601	E678[OVA]
602	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
125	1	Project X Architecture	4[ArchElem]
11147	1	Example SV-5	4[ArchElem]
229	1	ArchitectureDocument (SV-5 in Project X Architecture)	5[OBR]
601	1	Architecture is documented by SV-5	3[OVA]
602	1	SV-5 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
229	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
601	1	125	1	229	1	999	E038-R-E045
602	1	11147	1	229	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = [Architecture] is recorded in [ArchitectureDocument]
E148-R-E045 = [Document] records [ArchitectureDocument]

The instance tables for System A in the example are shown below.

Object

objectIdentifier	pointerCode
7501	E563[System]

ObjectVersion

*Identifier	*Index	Name	categoryCode
7501	1	System A	4[ArchElem]

Similarly, the following instance tables are used to express the *operational capabilities* shown in the notional SV-5 template.

Object

objectIdentifier	pointerCode
8111	E622[Task]
8112	E622[Task]

ObjectVersion

*Identifier	*Index	Name	categoryCode
8111	1	Capability 1	4[ArchElem]
8112	1	Capability 2	4[ArchElem]

The system functions are captured in CADM v1.5 as instances of ProcessActivity. The following instance table shows how this is done for this SV-5 example.

Object

objectIdentifier	pointerCode
5001	E486[ProcessActivity]
5002	E486[ProcessActivity]
5003	E486[ProcessActivity]
5004	E486[ProcessActivity]
5101	E486[ProcessActivity]
5102	E486[ProcessActivity]

ObjectVersion

*Identifier	*Index	Name	categoryCode
5001	1	PA 1 for C1	4[ArchElem]
5002	1	PA 2 for C1	4[ArchElem]
5003	1	PA 1 for C2	4[ArchElem]
5004	1	PA 2 for C2	4[ArchElem]
5101	1	System A-SF1	4[ArchElem]
5102	1	System A-SF2	4[ArchElem]

The next steps are essentially an exercise in the use of ObjectVersionAssociation to connect each of the subcomponents to each other (i.e., System A to its functions, the operational capabilities to their corresponding instances of ProcessActivity).

Since, in each case, the procedure is the same, only one such instance will be shown. First, we need to link System A to System Function 1. The table below shows how this is done through the use of the CADM v1.03 entity *SystemProcessActivity*.

Object

objectIdentifier	pointerCode
5701	E678[OVA]
5702	E678[OVA]
5721	E679[OBR]

ObjectVersion

*Identifier	*Index	Name	categoryCode
5701	1	Syst-to-SysProcAct	3[OVA]
5702	1	ProcAct-to-SysProcAct	3[OVA]
5721	1	SystProcAct01	5[OBR]

ObjectByReference

*Identifier	*Index	categoryCode
5721	1	E597[SystemProcessActivity]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
5701	1	7501	1	5721	1	999	E486-R-E597
5702	1	5101	1	5721	1	999	E563-R-E597

The relationTypeCode values used have the following meanings:

E563-R-E597 = [System] is designed to provide [*SystemProcessActivity*]
E486-R-E597 = [ProcessActivity (as SystemFunction)] is assigned to [*SystemProcessActivity*]

The linkage of instances of Task (subtyped as *OperationalCapabilityTask*) to the ProcessActivity instances follows the same pattern. The tables below show how this is done for Capability 1 and Process Activity 1 in the notional example.

As before one creates the instances of Object and ObjectVersion required. The following instance table shows how this is done for this SV-5 example.

Object

objectIdentifier	pointerCode
5001	E486[ProcessActivity]
5002	E486[ProcessActivity]
5003	E486[ProcessActivity]
5004	E486[ProcessActivity]
8111	E622[Task]
8112	E622[Task]
8113	E622[Task]

ObjectVersion

*Identifier	*Index	Name	categoryCode
5001	1	C1 PA1	4[ArchElem]
5002	1	C1 PA2	4[ArchElem]
5003	1	C2 PA1	4[ArchElem]
5004	1	C2 PA2	4[ArchElem]
8111	1	Operational Capability 1	4[ArchElem]
8112	1	Operational Capability 2	4[ArchElem]

The association between the instances of Task and their corresponding instances of ProcessActivity through the CADM v1.03 entity *ProcessActivityTask* is shown below for the first *operational capability* and its two *operational activities* from the example.

Object

objectIdentifier	pointerCode
5801	E678[OVA]
5802	E678[OVA]
5803	E678[OVA]
5804	E678[OVA]
5821	E679[OBR]
5822	E679[OBR]

ObjectVersion

*Identifier	*Index	Name	categoryCode
5801	1	ProcAct to ProcActTask 01	3[OVA]
5802	1	Task to ProcActTask 01	3[OVA]
5803	1	ProcAct to ProcActTask 02	3[OVA]
5804	1	Task to ProcActTask 02	3[OVA]
5821	1	ProcessActivityTask 01	5[OBR]
5822	1	ProcessActivityTask 02	5[OBR]

ObjectByReference

*Identifier	*Index	categoryCode
5821	1	E497[ProcessActivityTask]
5822	1	E497[ProcessActivityTask]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
5801	1	5001	1	5821	1	999	E486-R-E497
5802	1	8111	1	5821	1	999	E622-R-E497
5803	1	5002	1	5822	1	999	E486-R-E497
5804	1	8111	1	5822	1	999	E622-R-E497

The relationTypeCode values used have the following meanings:

E486-R-E497 = [ProcessActivity] corresponds to [*ProcessActivityTask*]

E622-R-E497 = [Task] corresponds to [*ProcessActivityTask*]

Once the systems and systems functions are defined and associated, as well as the operational capabilities and their related operational activities, one can instantiate the rows corresponding to the CADM entity *SystemFunctionTraceabilityMatrixElement.* The tables below show how this is done using the instances of ObjectVersionAssociation already built.

Object

objectIdentifier	pointerCode
101	E679[OBR]
2101	E678[OVA]
2102	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
101	1	SFTME 01	5[OBR]
2101	1	SystProcAct 5721 to SFTME 01	4[OVA]
2102	1	ProcActTask 5821 to SFTME 01	4[OVA]
2103	1	ProcActTask 5822 to SFTME 01	4[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
101	1	E582[SystemFunctionTraceabilityMatrixElement]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
2101	1	5721	1	101	1	999	E486-R-E497
2102	1	5821	1	101	1	999	E497-R-E582
2103	1	5822	1	101	1	999	E497-R-E582

The relationTypeCode values used have the following meanings:

E597-R-E582 = [*SystemProcessActivity*] may be cited for [SFTME]

E497-R-E582 = [*ProcessActivityTask*] is cited for [SFTME]

As noted in the section above, the SV-5 document is made up of all the instances of the *SystemFunctionTraceabilityMatrixElement.* In CADM v1.5, the linkage is also done through the ObjectVersionAssociation. The table below shows the entry for the example discussed (the instantiation of the ObjectVersionAssociation is not shown).

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
4101	1	11147	1	101	1	999	E581-R-E582

The relationTypeCode values used have the following meanings:

E581-R-E582 = [SV-5] is defined by [SFTME]

In CADM v1.02/v1.03, an extension to link *DataDomainListValue* to the elements of the SV-5 in a fashion similar to what is available for SV-3 was proposed. Because the change is not part of the approved specification, it is not discussed here. However, its instantiation can be readily accommodated through the use of ObjectVersionAssociation.

4.6.14.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to ProcessActivity through the characterization of *SoaServiceTraceabilityMatrixElement*. To do that, one needs to create the respective instance of *SoaServiceTraceabilityMatrixElement* through ObjectByReference and link it to SoaService in the ObjectVersionAssociation table with the relationTypeCode = E682-R-E689 (may be cited for). The corresponding instance of ProcessActivity can then be related to *SoaServiceTraceabilityMatrixElement* in the ObjectVersionAssociation table with the relationTypeCode = E486-R-E689 (is cited for).

4.6.15 CADM v1.5 Support for Systems and Services Data Exchange Matrix (SV-6)

4.6.15.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-6 specifies the characteristics of the system data exchanged between systems. This product focuses on automated information exchanges (from OV-3) that are implemented in systems. Non-automated information exchanges, such as verbal orders, are captured in the OV products only.

4.6.15.2 High-Level Description

Figure 4-37 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-6.

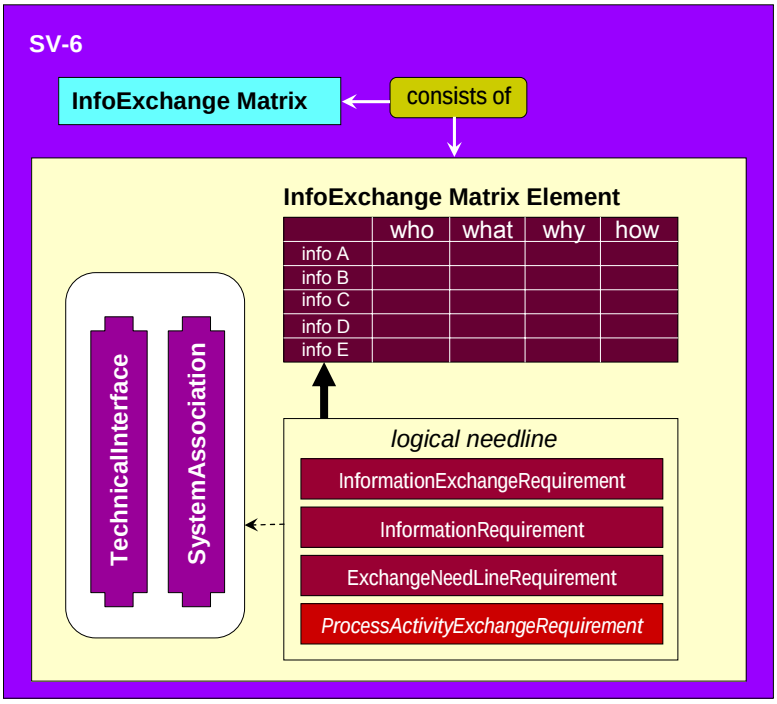


Figure 4-37: High-Level Depiction of CADM v1.5 Data Structures for SV-6 Representation

The DoDAF architecture product SV-6 is expressed in CADM v1.5 as an instance of Document. The SV-6 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference). As shown in Figure 4-37, the SV-6 is a presentation format that describes data exchanges between systems. Portions of this information are captured already in the characterization of the *system interfaces* depicted in the SV-1.

The actual data content of the SV-6 is built by linking it to instances of *InformationExchangeMatrixElement*, which themselves can be linked to instances of InformationElement, ExchangeNeedlineRequirement, and InformationExchangeRequirement. The *InformationExchangeMatrixElement* collects the corresponding **IER** for the sending system and the receiving system. The identification of system functions supported for each pair of systems involved in the exchanged is accomplished through SystemProcessActivity. The specific characteristics (e.g. accuracy, size, timeliness) of the exchange are specified through the attribution of the **IER** (see SV-1 example above).

4.6.15.3 CADM v1.5 Instantiation

The SV-6 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
1127	E679[OBR]
601	E678[OVA]
602	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	SV-6	4[ArchElem]
1127	1	ArchitectureDocument (SV-6 in Program Architecture[126]	5[OBR]
601	1	Architecture is documented by SV-6	3[OVA]
602	1	SV-6 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
1127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
601	1	125	1	1127	1	999	E038-R-E045
602	1	126	1	1127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The instance of Document representing the SV-6 can now be linked to each of the required instances of *InformationExchangeMatrixElement* (expressed through ObjectByReference).

Object

objectIdentifier	pointerCode
701	679 [OBR]
702	679 [OBR]
703	679 [OBR]
704	679 [OBR]
671	678 [OVA]
672	678 [OVA]
673	678 [OVA]
674	678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
701	1	IER Matrix Element 1	5[OBR]
702	1	IER Matrix Element 2	5[OBR]
703	1	IER Matrix Element 3	5[OBR]
704	1	IER Matrix Element 4	5[OBR]
671	1	OV-3[125] contains IER ME 1	3[OVA]
672	1	OV-3[125] contains IER ME 2	3[OVA]
673	1	OV-3[125] contains IER ME 3	3[OVA]
674	1	OV-3[125] contains IER ME 4	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
701	1	E226[IER Matrix Element]
702	1	E226[IER Matrix Element]
703	1	E226[IER Matrix Element]
704	1	E226[IER Matrix Element]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
671	1	126	1	701	1	999	E225-R-E226
672	1	126	1	702	1	999	E225-R-E226
673	1	126	1	703	1	999	E225-R-E226
674	1	126	1	704	1	999	E225-R-E226

E225-R-E226 = contains

Finally, each *InformationExchangeMatrixElement* can be linked to the pertinent instance of InformationExchangeRequirement. For the purpose of illustration, one can take the instance already created for the SV-1 example discussed in the previous section. The only addition required is the new instance of OVA. The *Logical Needline* can be associated to instance of TechnicalInterface and instances of SystemAssociation, as discussed in the SV-1 section above.

Object

objectIdentifier	pointerCode
309	E234[InformationExchangeRequirement]
901	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
309	1	IER for Interface 1	4[ArchElem]
901	1	IER[309]] to IER Matrix Element 1	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
901	1	307	1	701	1	999	E234-R-E226

The relationTypeCode values used have the following meanings:

E234-R-E226 = is referenced in

4.6.15.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to SoaService via *SoaServiceAssociation*. The attribute E684.A01 = typeCode in the ObjectVersionAssociationCharacterization for the instances of *SoaServiceAssociation* can be set to "*sends to and receives data from.*" Next, one can link the instances of the service associations to ActivityModelInformationElementRole for the flows that have categoryCode = 1 (input) or categoryCode = 2 (output).

4.6.16 CADM v1.5 Support for Systems and Services Performance Parameters Matrix (SV-7)

4.6.16.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-7 specifies the quantitative characteristics of systems and system hardware/software items, its interfaces (system data carried by the interface as well as communications link details that implement the interface), and its functions. It specifies the current performance parameters of each system, interface, or system function, and the expected or required performance parameters at specified times in the future.

4.6.16.2 High-Level Description

Figure 4-38 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-7.

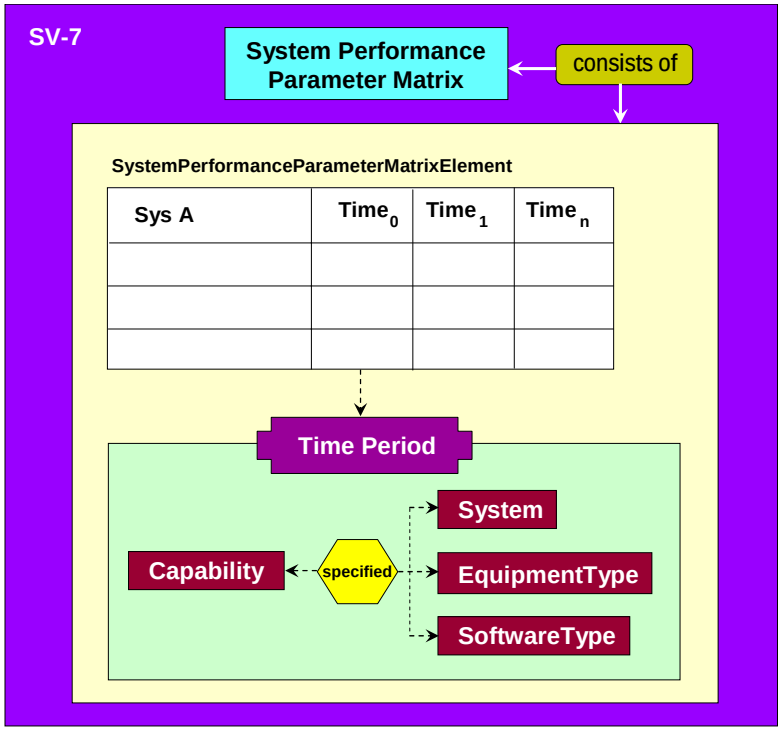


Figure 4-38: High-Level Depiction of CADM v1.5 Data Structures for SV-7 Representation

The DoDAF architecture product SV-7 is expressed in CADM v1.5 as an instance of Document. The SV-7 document can be connected to the appropriate instance of Architecture that

it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference).

The actual data content of the SV-7 is built by linking it to instances of the CADM v1.03 entity *SystemPerformanceParameterMatrixElement* (instantiated through ObjectByReference). Each of the elements can be related to the capability for a system (specified through *SystemCapability*), the applicable time period, as well as any constraints that may apply. Additional data related to the implementation time frame for the systems can be expressed through *SystemImplementationTimeFrame*.

The relationship between System and specific instance of EquipmentType is done through the CADM v1.03 *SystemEquipmentType***,** instantiated through ObjectByReference. Similarly, the linkage between System and Software is accomplished through the CADM v1.03 *SystemSoftwareType*, instantiated through ObjectByReference. The capabilities (e.g. maintainability, response time) for each *hardware element* is specified through the CADM v1.03 entity *MaterielTypeCapabilityNorm*, instantiated through ObjectByReference.

4.6.16.3 CADM v1.5 Instantiation

Figure 4-39 shows a notional SV-7 for an intelligence system. Three periods are specified indicating the current, mid-term, and long-term time frames.

SV-7 for Intelligence System AX-900 (Notional)				
Performance Measure	**Current Value**	**2010 Value**	**2020 Value**	**Units**
Mean Time Between Failure	168	1000	3000	hours
Best Display Resolution	1024 by 1024	1600 by 1200	4096 by 8192	pixels
Diagonal Roam Rate	512	1024	4096	pixels/second

Figure 4-39: Notional SV-7 Template (Partial View)

The SV-7 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
1125	E038[Architecture]
1126	E148[Document]
1127	E679[OBR]
1601	E678[OVA]
1602	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
1125	1	Project Charlie-Bravo Architecture	4[ArchElem]
1126	1	SV-7	4[ArchElem]
1127	1	ArchitectureDocument (SV-7 in Program Architecture[1126]	5[OBR]
1601	1	Architecture is documented by SV-7	3[OVA]
1602	1	SV-7 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
1127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
1601	1	1125	1	1127	1	999	E038-R-E045
1602	1	1126	1	1127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The content of the SV-7 captured through instances of ObjectByReferece corresponding to the CADM v1.03 entity *SystemPerformanceParameterMatrixElement*. The tables below show how this is done in CADM v1.5.

Object

objectIdentifier	pointerCode
9301	679[OBR]
9302	679[OBR]
9303	679[OBR]
9304	679[OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
9301	1	SPPME 01	5[OBR]
9302	1	SPPME 01	5[OBR]
9303	1	SPPME 01	5[OBR]
9304	1	SPPME 01	5[OBR]

ObjectByReference

*Identifier	*Index	categoryCode
9301	1	E595 [SystemPerformanceParameterMatrixElement]
9302	1	E595 [SystemPerformanceParameterMatrixElement]
9303	1	E595 [SystemPerformanceParameterMatrixElement]
9304	1	E595 [SystemPerformanceParameterMatrixElement]

The relationship between the SV-7 and the instances of *SystemPerformanceParameterMatrixElement* is done through ObjectByReference in the usual manner. The table below shows the entries corrensponding to the records declared above (the Object and ObjectVersion entries for the ObjectVersionAssociation are not shown).

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
7701	1	1126	1	9301	1	999	E594-R-E595
7702	1	1126	1	9302	1	999	E594-R-E595
7703	1	1126	1	9303	1	999	E594-R-E595
7704	1	1126	1	9304	1	999	E594-R-E595

The relationTypeCode values used have the following meanings:

E594-R-E595 = [SV-7] cites [*SystemPerformanceParameterMatrixElement*]

Each of the elements that makes up the SV-7 can be linked to instances of *SystemCapability*, Period, as well as instances of *SystemDirectedConstraint*.

The first step is to relate systems to capabilities. As the notional example above shows, there are four capabilities specified for the system in question. Each capability has three values according to the period it is expected to be implemented. This means that one must create a total of 12 entries for *SystemCapability*, so that each capability can be related to its corresponding time frame. Since there are two entries for each *SystemCapability* (one for the capability and one for the system), one needs a total of 24 associations.

The tables below show the entries for instances of Capability, System and their association through *SystemCapability*. (Note: the instances of Object and ObjectVersion for the ObjectVersionAssociation are not shown.)

Object

objectIdentifier	pointerCode
3058	E082 [Capability]
3052	E082 [Capability]
3053	E082 [Capability]
3056	E082 [Capability]
4051	E563 [System]
5051	679 [OBR]
5052	679 [OBR]
5053	679 [OBR]
5054	679 [OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
3058	1	Mean time between failure	4 [AE]
3052	1	Best horizontal display resolution	4 [AE]
3053	1	Best vertical display resolution	4 [AE]
3056	1	Diagonal roam rate	4 [AE]
4051	1	Intelligence System AX-900	4 [AE]
5051	1	SPPME A-01	5 [OBR]
5052	1	SPPME A-02	5 [OBR]
5053	1	SPPME A-03	5 [OBR]
5054	1	SPPME B-01	5 [OBR]
5055	1	SPPME B-02	5 [OBR]
5056	1	SPPME B-03	5 [OBR]
5057	1	SPPME C-03	5 [OBR]
5058	1	SPPME C-01	5 [OBR]
5059	1	SPPME C-02	5 [OBR]
5060	1	SPPME D-03	5 [OBR]
5061	1	SPPME D-01	5 [OBR]
5061	1	SPPME D-02	5 [OBR]

ObjectByReference

*Identifier	*Index	categoryCode
5051	1	E570 [SystemCapabililty]
5052	1	E570 [SystemCapabililty]
5053	1	E570 [SystemCapabililty]
5054	1	E570 [SystemCapabililty]
5055	1	E570 [SystemCapabililty]
5056	1	E570 [SystemCapabililty]
5057	1	E570 [SystemCapabililty]
5058	1	E570 [SystemCapabililty]
5059	1	E570 [SystemCapabililty]
5060	1	E570 [SystemCapabililty]
5061	1	E570 [SystemCapabililty]
5062	1	E570 [SystemCapabililty]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
6701	1	4051	1	5051	1	999	E563-R-E570
6702	1	3058	1	5051	1	999	E082-R-E570
6703	1	4051	1	5052	1	999	E563-R-E570
6704	1	3058	1	5052	1	999	E082-R-E570
6705	1	4051	1	5053	1	999	E563-R-E570
6706	1	3058	1	5053	1	999	E082-R-E570
6707	1	4051	1	5054	1	999	E563-R-E570
6708	1	3052	1	5054	1	999	E082-R-E570
6709	1	4051	1	5055	1	999	E563-R-E570
6710	1	3052	1	5055	1	999	E082-R-E570
6711	1	4051	1	5056	1	999	E563-R-E570
6712	1	3052	1	5056	1	999	E082-R-E570
6713	1	4051	1	5057	1	999	E563-R-E570
6714	1	3053	1	5057	1	999	E082-R-E570
6715	1	4051	1	5058	1	999	E563-R-E570
6716	1	3053	1	5058	1	999	E082-R-E570
6717	1	4051	1	5059	1	999	E563-R-E570
6718	1	3053	1	5059	1	999	E082-R-E570
6719	1	4051	1	5060	1	999	E563-R-E570
6720	1	3056	1	5060	1	999	E082-R-E570
6721	1	4051	1	5061	1	999	E563-R-E570
6722	1	3056	1	5061	1	999	E082-R-E570
6723	1	4051	1	5062	1	999	E563-R-E570
6724	1	3056	1	5062	1	999	E082-R-E570

The relationTypeCode values used have the following meanings:

E563-R-E570 = [System**] performs to [*SystemCapability*]**
E082-R-E570 = [Capability**] is performed by [*SystemCapability*]**

The next step is the creation of the required instances of Period. The tables below show how this is done in CADM v1.5.

Object

objectIdentifier	pointerCode
320021	E467 [Period]
320022	E467 [Period]
320023	E467 [Period]

ObjectVersion

*Identifier	*Index	name	categoryCode
320021	1	Current	4 [AE]
320022	1	2010	4 [AE]
320023	1	2020	4 [AE]

Not shown here but available in CADM v1.5 is the specification for each System of the applicable ImplementationTimeFrame.

The final step is to create the relationship of the *SystemCapability* instances and the Period instances to the pertinent intances of *SystemPerformancePamaterMatrixElement*. This is accomplished through ObjectVersionAssociation, as shown below. (Note: The instances of Object and ObjectVersion for the ObjectVersionAssociation are not shown.) The *SystemCapability* instances are linked to the corresponding instances of Period for the current, mid-term, and long-term time frames. Because there are two relationships for each entry, there are a total of 24 rows in the **ObjectVersionAssociation** table.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
48701	1	4051	1	9301	1	999	E570-R-E595
48702	1	320021	1	9301	1	999	E467-R-E595
48703	1	4051	1	9301	1	999	E570-R-E595
48704	1	320022	1	9301	1	999	E467-R-E595
48705	1	4051	1	9301	1	999	E570-R-E595
48706	1	320023	1	9301	1	999	E467-R-E595
48707	1	4051	1	9302	1	999	E570-R-E595
48708	1	320021	1	9302	1	999	E467-R-E595
48709	1	4051	1	9302	1	999	E570-R-E595
48710	1	320022	1	9302	1	999	E467-R-E595
48711	1	4051	1	9302	1	999	E570-R-E595
48712	1	320023	1	9302	1	999	E467-R-E595
48713	1	4051	1	9303	1	999	E570-R-E595
48714	1	320021	1	9303	1	999	E467-R-E595
48715	1	4051	1	9303	1	999	E570-R-E595
48716	1	320022	1	9303	1	999	E467-R-E595
48717	1	4051	1	9303	1	999	E570-R-E595
48718	1	320023	1	9303	1	999	E467-R-E595
48719	1	4051	1	9304	1	999	E570-R-E595
48720	1	320021	1	9304	1	999	E467-R-E595
48721	1	4051	1	9304	1	999	E570-R-E595
48722	1	320022	1	9304	1	999	E467-R-E595
48723	1	4051	1	9304	1	999	E570-R-E595
48724	1	320023	1	9304	1	999	E467-R-E595

The relationTypeCode values used have the following meanings:

E570-R-E595 = [*SystemCapability*] applies to [SPPME]

E467-R-E595 = [Period] applies to [SPPME]

4.6.16.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to SoaServiceSpecificationTemplate. To do that, one needs to set in the respective instance of SoaService and link it to SoaServiceSpecificationTemplate in the ObjectVersionAssociation table with the relationship E682-R-E681 = *has profile specified by*.

4.6.17 CADM v1.5 Support for Systems and Services Evolution Description (SV-8)

4.6.17.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-8 captures evolution plans that describe how the system, or the architecture in which the system is embedded, will evolve over a lengthy period of time. Generally, the timeline milestones are critical for a successful understanding of the evolution timeline.

4.6.17.2 High-Level Description

Figure 4-40 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-8.

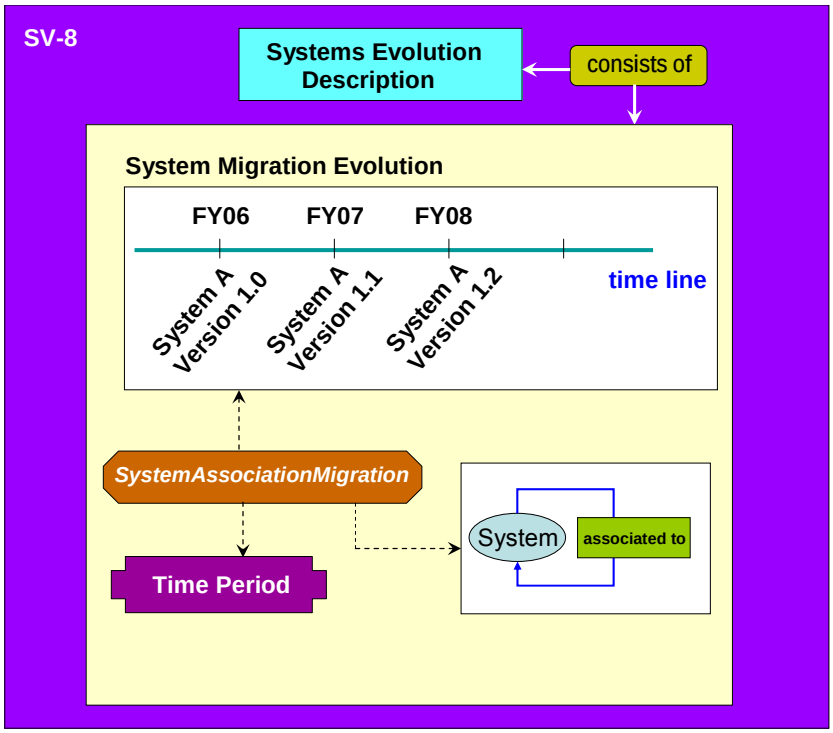


Figure 4-40: High-Level Depiction of CADM v1.5 Data Structures for SV-8 Representation

The DoDAF architecture product SV-8 is expressed in CADM v1.5 as an instance of Document. The SV-8 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference).

The actual data content of the SV-8 is built by linking it to instances of the CADM v1.03 entity *SystemMigrationEvolution* (instantiated through ObjectByReference). Each instance can be

linked to the pertinent pairs of systems that are specified through *SystemAssociationMigration*. The latter can cite a specific Guidance (e.g., InformationTechnologyRequirement, *TechnicalCriterion*), a specific Agreement (e.g., a standard or standard profile), and a time frame Period. *SystemAssociationMigration* is linked to *SystemAssociation*. Through it, one can also specify additional information, e.g., via *SystemAssociationMeans*.

4.6.17.3 CADM v1.5 Instantiation

Figure 41 below shows a notional SV-8 product for the target MIDB system. The migration from the current baseline to the target system is shown as a series of version releases that incorporate new capabilities for each component.

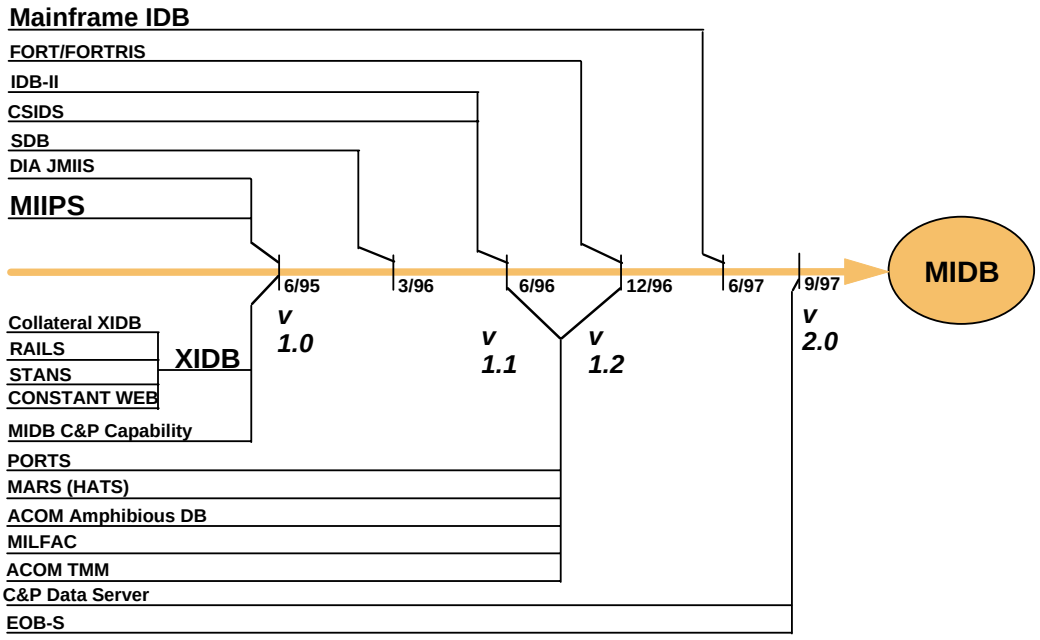


Figure 4-41: Systems Evolution Description Example

The SV-8 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
1136	E038[Architecture]
1137	E148[Document]
1138	E679[OBR]
1605	E678[OVA]
1606	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
1136	1	Project Charlie-Bravo Architecture	4[ArchElem]
1137	1	SV-8	4[ArchElem]
1138	1	ArchitectureDocument (SV-8 in Program Architecture[1136]	5[OBR]
1605	1	Architecture is documented by SV-8	3[OVA]
1606	1	SV-8 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
1138	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
1605	1	1136	1	1138	1	999	E038-R-E045
1606	1	1137	1	1138	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The SV-8 is linked to the actual content of the product through the CADM v1.03 entity *SystemMigrationEvolution*, which collects the entries from *SystemAssociationMigration*, a child of *SystemAssociation* (instantiated as ObjectVersionAssociation). For each of the instances of *SystemAssociationMigration* one can specify the applicable instance of Period in a manner similar to the SV-8 product instantiation discussed in the preceding section.

From the table above, it follows that the representation of the data for an SV-8 in CADM v1.5 requires, as the first step, the specification of the systems involved. The entries corresponding to the notional example shown in the figure above are given below.

Object

objectIdentifier	pointerCode
93051	E563 [System]
93052	E563 [System]
93053	E563 [System]
93054	E563 [System]
94055	E563 [System]
95056	E563 [System]
95057	E563 [System]
95058	E563 [System]
95059	E563 [System]
95060	E563 [System]
95061	E563 [System]
95062	E563 [System]

ObjectVersion

*Identifier	*Index	name	categoryCode
93051	1	MF-IDB	4 [AE]
93052	1	FORTIS	4 [AE]
93053	1	IDB-II	4 [AE]
93054	1	CSIDS	4 [AE]
94055	1	SDB	4 [AE]
95056	1	DIA JMIIS	4 [AE]
95057	1	MIIPS	4 [AE]
95058	1	MIDB 1.0	4 [AE]
95059	1	MIDB 1.1	4 [AE]
95060	1	MIDB 1.2	4 [AE]
95061	1	MIDB 2.0	4 [AE]
95062	1	MIDB	4 [AE]

In the example, there are 10 system associations. The entries in the ObjectVersionAssociation table are shown below (the instances for the corresponding Object and ObjectVersion are not shown).

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
125001	1	95062	1	93051	1	465	NULL
125002	1	95060	1	93052	1	465	NULL
125003	1	95059	1	93053	1	465	NULL
125004	1	95059	1	93054	1	465	NULL
125005	1	95058	1	94055	1	465	NULL
125006	1	95058	1	95056	1	465	NULL
125007	1	95058	1	95057	1	465	NULL
125008	1	95059	1	95058	1	465	NULL
125009	1	95060	1	95059	1	465	NULL
125010	1	95061	1	95060	1	465	NULL

The value categoryCode = 465 [*SystemAssociation*]

The CADM v1.03 entity *SystemAssociationMigration* is a child of *SystemAssociation*. In this example, however, there is only one entry per instance of *SystemAssociation*. The ObjectVersionAssociation entries below show how the instances are related. The corresponding

Object and ObjectVersion for *SystemAssociationMigration* are not shown but their instantation follows the same pattern that has been shown before. The only thing to note is that, in this case, the identifier for the subject instances in the ObjectVersionAssociation correspond to the ones for *SystemAssociation* shown in the previous table, since, unlike the other cases shown before where the CADM v1.03 entity mapped to ObjectByReference, the CADM v1.03 entity *SystemAssociation* is represented directly by ObjectVersionAssociation in CADM v1.5.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
225001	1	125001	1	493051	1	999	NULL
225002	1	125002	1	493052	1	999	NULL
225003	1	125003	1	493053	1	999	NULL
225004	1	125004	1	493054	1	999	NULL
225005	1	125005	1	494055	1	999	NULL
225006	1	125006	1	495056	1	999	NULL
225007	1	125007	1	495057	1	999	NULL
225008	1	125008	1	495058	1	999	NULL
225009	1	125009	1	495059	1	999	NULL
225010	1	125010	1	495060	1	999	NULL

E565-R-E569 = [*SystemAssociation*] is cited for [*SystemAssociationMigration*]

The applicable time frame is specified as an association from Period to *SystemAssociation Migration*. The tables below show that instantiation.

Object

objectIdentifier	pointerCode
171	E467 [Period]
172	E467 [Period]
173	E467 [Period]
174	E467 [Period]
175	E467 [Period]
176	E467 [Period]
177	E467 [Period]

ObjectVersion

*Identifier	*Index	name	categoryCode
171	1	Phase 1	4 [AE]
172	1	Phase 1A	4 [AE]
173	1	Phase 1.1	4 [AE]
174	1	Phase 1.2	4 [AE]
175	1	Phase 1.2A	4 [AE]
176	1	Phase 2.0	4 [AE]
177	1	Beyond Phase 2.0	4 [AE]

The link between these Period entries and the *SystemAssociationMigration* is shown below. The instances of Object and ObjectVersion for ObjectVersionAssociation are not shown.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
725001	1	175	1	493051	1	999	
725002	1	174	1	493052	1	999	
725003	1	173	1	493053	1	999	
725004	1	173	1	493054	1	999	
725005	1	172	1	494055	1	999	
725006	1	172	1	495056	1	999	
725007	1	171	1	495057	1	999	
725008	1	171	1	495058	1	999	
725009	1	173	1	495059	1	999	
725010	1	174	1	495060	1	999	

E467-R-E569 = [Period**] applies to [*SystemAssociationMigration*]**

The connection between the SV-8 document and the records corresponding to *SystemAssociationMigration* is done by creating instances corresponding to *SystemMigrationEvolution* and relating the two parent entities, namely, the SV-8 document and the instances of *SystemAssociationMigration* defined previously. Since the pattern is exactly the same as what has been shown before, the only thing to note is that the codes to use in the ObjectVersionAssociation table are:

E578-R-E588 = [SV-8**]** cites **[***SystemMigrationEvolution***]**
E569-R-E588 = [*SystemAssociationMigration***]** is cited in **[***SystemMigrationEvolution***]**

4.6.17.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to *SoaServiceAssociationMigration* through the characterization of *SoaServiceAssociation.* To do that, one needs to set in the respective instance of *SoaServiceAssociation* and link it to *SoaServiceAssociationMigration* in the ObjectVersionAssociation table with the relationTypeCode = E684-R-E690 (is cited for).

4.6.18 CADM v1.5 Support for Systems Technology Forecase (SV-9)

4.6.18.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-9 defines the underlying current and expected supporting technologies. It is not expected to include predictions of technologies as with a crystal ball. Expected supporting technologies are those that can be reasonably forecast given the current state of technology and expected improvements. New technologies should be tied to specific time periods, which can correlate against the time periods used in SV-8 milestones.

4.6.18.2 High-Level Description

Figure 4-42 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-9.

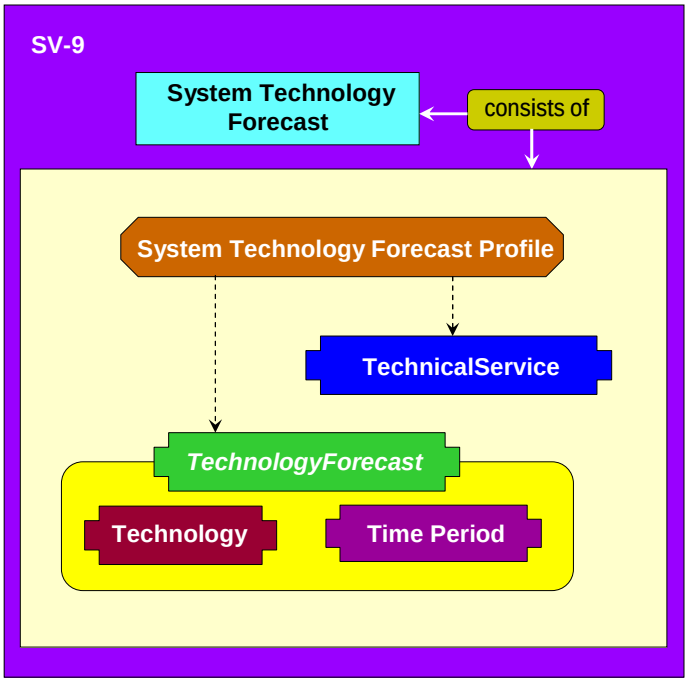


Figure 4-42: High-Level Depiction of CADM v1.5 Data Structures for SV-9 Representation

The DoDAF architecture product SV-9 is expressed in CADM v1.5 as an instance of Document. The SV-9 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference).

The actual data content of the SV-9 is built by linking it to one or more instances of the CADM v1.03 entity *SystemTechnologyForecastProfile*. The latter can be associated with a specific System, a specific TechnicalService, a *TechnologyCountermeasure*, a *TechnologyForecast*, and a Period. The Period is the same entity used to characterize a *TechnicalStandardForecastElement*, *TechnicalStandardForecast*, and *TechnologyForecast*. The specification of *TechnologyForecast* requires the appropriate data in the Technology table. The CADM also provides additional structures related to Technology such as *TechnologyAssociation*, *TechnologyCountermeasure*, *TechnicalCriterion*, *TechnologyForecast*, and *TechnologyIssue*.

4.6.18.3 CADM v1.5 Instantiation

Table 4-3 below shows a notional example for an SV-9 product, where a series of services (as defined by DISR) are shown in terms of current and future availability. The estimated time frames bound the forecast and provide input for implementation decisions.

Table 4-3: Systems Technology Forecast (SV-9)—Notional Example

DISR Service	TECHNOLOGY FORECASTS		
	SHORT TERM (0-6 Months)	MID TERM (6-12 Months)	LONG TERM (12-18 Months)
Application Software			
Support Applications	MS Office 2007 available (for Windows XP)	MS Office 2007 stable enough for full-scale implementation	MS Office 2007 available for Apple platfoms, E-mail on wireless PDAs commonplace
Application Platform			
Data Management	Oracle 10g available MySQL (Open Source DBMS) available	—	—
Operating System	—	Next MS Windows Vista upgrade expected Next Fedora Core 7 Linux major release expected	Next MS Windows server upgrade expected
Physical Environment	—	—	Intel IA-64 becomes standard processor for desktops Low-cost availability of parallel computing technologies for laptops
External Environment			
User Interface	—	Thin screen CRT monitors for PC desktops become price competitive	Thin screen LED monitors become price competitive for desktops Conventional CRT technology monitors for desktops become obsolete
Persistent Storage	5G PCMCIA type 2 card available	—	Disk storage capacity doubles again
Communications networks	—	Wireless internet service available for most telecommuting staff	Broad-band fiber optic connections available for most telecommuting staff

The SV-9 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
91136	E038[Architecture]
91137	E148[Document]
91138	E679[OBR]
91605	E678[OVA]
91606	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
91136	1	Project Charlie-Bravo Architecture	4[ArchElem]
91137	1	SV-9	4[ArchElem]
91138	1	ArchitectureDocument (SV-9 in Program Architecture [91136]	5[OBR]
91605	1	Architecture is documented by SV-9	3[OVA]
91606	1	SV-9 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
91138	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
91605	1	91136	1	91138	1	999	E038-R-E045
91606	1	91137	1	91138	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The content of the SV-9 document is defined through instances of ObjectByReference corresponding to the CADM v1.03 entity *SystemTechnologyForecastProfile*. The latter can be linked to instances of ObjectByReference corresponding to the CADM v1.03 entities *TechnologyForecast*, and TechnologyCountermeasure, as well as instances of Period TechnicalInterface, and System.

To express the data of the notional SV-9 shown in the table above using CADM v1.5, one needs to define first the instances of TechnicalService and *TechnologyForecast*. The instance tables below show how this is done.

Object

objectIdentifier	pointerCode
501	E644 [TechnicalService]
601	E644 [TechnicalService]
602	E644 [TechnicalService]
603	E644 [TechnicalService]
701	E644 [TechnicalService]
702	E644 [TechnicalService]
703	E644 [TechnicalService]

ObjectVersion

*Identifier	*Index	name	categoryCode
501	1	Support Applications	4 [AE]
601	1	Data Management	4 [AE]
602	1	Operating System	4 [AE]
603	1	Physical Environment	4 [AE]
701	1	User Interface	4 [AE]
702	1	Persistent Storage	4 [AE]
703	1	Communications Networks	4 [AE]

Next the instances of Technology and *TechnologyForecast* can be created and their relationships specified through ObjectVersionAssociation.

Object

objectIdentifier	pointerCode
1011	E650 [Technology]
1012	E650 [Technology]
1013	E650 [Technology]
1014	E650 [Technology]
1015	E650 [Technology]
1016	E650 [Technology]
1017	E650 [Technology]
401	E679 OBR]

objectIdentifier	pointerCode
402	E679 OBR]
403	E679 OBR]
404	E679 OBR]
405	E679 OBR]
406	E679 OBR]
407	E679 OBR]
408	E679 OBR]
409	E679 OBR]
410	E679 OBR]
411	E679 OBR]
412	E679 OBR]
413	E679 OBR]
414	E679 OBR]
415	E679 OBR]
416	E679 OBR]
417	E679 OBR]
418	E679 OBR]

ObjectVersion

*Identifier	*Index	name	descriptionText	categoryCode
1011	1	Software Applications Technology		4 [AE]
1012	1	Data Management Technology		4 [AE]
1013	1	Operating System Technology		4 [AE]
1014	1	Physical Environment Technology		4 [AE]
1015	1	User Interface Technology		4 [AE]
1016	1	Data Storage Technology		4 [AE]
1017	1	Communications Technology		4 [AE]
401	1	—	MS Office 2000 available (for Windows 2000)	5 [OBR]
402	1	—	MS Office 2000 stable enough for full-scale implementation	5 [OBR]
403	1	—	MS Office available for Linux	5 [OBR]
404	1	—	E-mail on wireless PDAs commonplace	5 [OBR]
405	1	—	Oracle 9i available	5 [OBR]
406	1	—	MySQL (Open Source DBMS) available	5 [OBR]
407	1	—	Next MS Windows desktop upgrade expected	5 [OBR]
408	1	—	Next Red Hat Linux major release expected	5 [OBR]
409	1	—	Next MS Windows server upgrade expected	5 [OBR]
410	1	—	Intel IA-64 becomes standard processor for desktops	5 [OBR]
411	1	—	Initial use of quantum computing technologies	5 [OBR]
412	1	—	Thin screen CRT monitors for PC desktops become price competitive	5 [OBR]
413	1	—	Thin screen LED monitors become price competitive for desktops	5 [OBR]
414	1	—	Conventional CRT technology monitors for desktops become obsolete	5 [OBR]
415	1	—	5G PCMCIA type 2 card available	5 [OBR]
416	1	—	Disk storage capacity doubles again	5 [OBR]
417	1	—	Cable modem service available for most telecommuting staff	5 [OBR]

*Identifier	*Index	name	descriptionText	categoryCode
418	1	–	Fiber optic connections available for most telecommuting staff	5 [OBR]

ObjectByReference

*Identifier	*Index	categoryCode
401	1	E654 [TechnologyForecast]
402	1	E654 [TechnologyForecast]
403	1	E654 [TechnologyForecast]
404	1	E654 [TechnologyForecast]
405	1	E654 [TechnologyForecast]
406	1	E654 [TechnologyForecast]
407	1	E654 [TechnologyForecast]
408	1	E654 [TechnologyForecast]
409	1	E654 [TechnologyForecast]
410	1	E654 [TechnologyForecast]
411	1	E654 [TechnologyForecast]
412	1	E654 [TechnologyForecast]
413	1	E654 [TechnologyForecast]
414	1	E654 [TechnologyForecast]
415	1	E654 [TechnologyForecast]
416	1	E654 [TechnologyForecast]
417	1	E654 [TechnologyForecast]
418	1	E654 [TechnologyForecast]

The applicable time frame for the *TechnologyForecast* instances is expressed in CADM v1.5 by creating the applicable entries in the Period table and linking them through ObjectVersionAssociation as shown below.

Object

objectIdentifier	pointerCode
271	E467 [Period]
272	E467 [Period]
275	E467 [Period]
276	E467 [Period]

ObjectVersion

*Identifier	*Index	name	categoryCode
271	1	Current	4 [AE]
272	1	Short-Term	4 [AE]
275	1	Mid-Term	4 [AE]
276	1	Long-Term	4 [AE]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
877741	1	272	1	401	1	999	E467-R-E654
877742	1	275	1	402	1	999	E467-R-E654

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
877743	1	276	1	403	1	999	E467-R-E654
877744	1	276	1	404	1	999	E467-R-E654
877745	1	272	1	405	1	999	E467-R-E654
877746	1	272	1	406	1	999	E467-R-E654
877747	1	275	1	407	1	999	E467-R-E654
877748	1	275	1	408	1	999	E467-R-E654
877749	1	276	1	409	1	999	E467-R-E654
877750	1	276	1	410	1	999	E467-R-E654
877751	1	276	1	411	1	999	E467-R-E654
877752	1	275	1	412	1	999	E467-R-E654
877753	1	276	1	413	1	999	E467-R-E654
877754	1	276	1	414	1	999	E467-R-E654
877755	1	272	1	415	1	999	E467-R-E654
877756	1	276	1	416	1	999	E467-R-E654
877757	1	275	1	417	1	999	E467-R-E654
877758	1	276	1	418	1	999	E467-R-E654

E467-R-E654 = [Period] is the time frame for [*TechnologyForecast*]

The association of Technology to *TechnologyForecast* is done through ObjectVersionAssociation. The tables below show the instantiation for the notional example presented in this section. The Object and ObjectVersion tables are not included.

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
71601	1	1011	1	401	1	999	E650-R-E654
71602	1	1011	1	402	1	999	E650-R-E654
71603	1	1011	1	403	1	999	E650-R-E654
71604	1	1011	1	404	1	999	E650-R-E654
71605	1	1012	1	405	1	999	E650-R-E654
71606	1	1012	1	406	1	999	E650-R-E654
71607	1	1013	1	407	1	999	E650-R-E654
71608	1	1013	1	408	1	999	E650-R-E654
71609	1	1013	1	409	1	999	E650-R-E654
71610	1	1014	1	410	1	999	E650-R-E654
71611	1	1014	1	411	1	999	E650-R-E654
71612	1	1015	1	412	1	999	E650-R-E654
71613	1	1015	1	413	1	999	E650-R-E654
71614	1	1015	1	414	1	999	E650-R-E654
71615	1	1016	1	415	1	999	E650-R-E654
71616	1	1016	1	416	1	999	E650-R-E654
71617	1	1017	1	417	1	999	E650-R-E654
71618	1	1017	1	418	1	999	E650-R-E654

E650-R-E654 = [Technology] is expected to have [*TechnologyForecast*]

The last step is to link through *SystemTechnologyForecastProfile* both the SV-9 document and the instances of *TechnologyForecast*. The Object and ObjectVersion for the ObjectVersionAssociation table are not shown.

Object

objectIdentifier	pointerCode
811641	E679 [OBR]
811642	E679 [OBR]
811643	E679 [OBR]
811644	E679 [OBR]
811645	E679 [OBR]
811646	E679 [OBR]
811647	E679 [OBR]
811648	E679 [OBR]
811649	E679 [OBR]
811650	E679 [OBR]
811651	E679 [OBR]
811652	E679 [OBR]
811653	E679 [OBR]
811654	E679 [OBR]
811655	E679 [OBR]
811656	E679 [OBR]
811657	E679 [OBR]
811658	E679 [OBR]

ObjectVersion

*Identifier	*Index	name	categoryCode
811641	1	STFP[811641]	5 [OBR]
811642	1	STFP[811642]	5 [OBR]
811643	1	STFP[811643]	5 [OBR]
811644	1	STFP[811644]	5 [OBR]
811645	1	STFP[811645]	5 [OBR]
811646	1	STFP[811646]	5 [OBR]
811647	1	STFP[811647]	5 [OBR]
811648	1	STFP[811648]	5 [OBR]
811649	1	STFP[811649]	5 [OBR]
811650	1	STFP[811650]	5 [OBR]
811651	1	STFP[811651]	5 [OBR]
811652	1	STFP[811652]	5 [OBR]
811653	1	STFP[811653]	5 [OBR]
811654	1	STFP[811654]	5 [OBR]
811655	1	STFP[811655]	5 [OBR]
811656	1	STFP[811656]	5 [OBR]
811657	1	STFP[811657]	5 [OBR]
811658	1	STFP[811658]	5 [OBR]

ObjectByReference

*Identifier	*Index	categoryCode
811641	1	E617 [SystemTechnologyForecastProfile]
811642	1	E617 [SystemTechnologyForecastProfile]
811643	1	E617 [SystemTechnologyForecastProfile]
811644	1	E617 [SystemTechnologyForecastProfile]
811645	1	E617 [SystemTechnologyForecastProfile]
811646	1	E617 [SystemTechnologyForecastProfile]
811647	1	E617 [SystemTechnologyForecastProfile]
811648	1	E617 [SystemTechnologyForecastProfile]
811649	1	E617 [SystemTechnologyForecastProfile]
811650	1	E617 [SystemTechnologyForecastProfile]
811651	1	E617 [SystemTechnologyForecastProfile]
811652	1	E617 [SystemTechnologyForecastProfile]
811653	1	E617 [SystemTechnologyForecastProfile]
811654	1	E617 [SystemTechnologyForecastProfile]
811655	1	E617 [SystemTechnologyForecastProfile]
811656	1	E617 [SystemTechnologyForecastProfile]
811657	1	E617 [SystemTechnologyForecastProfile]
811658	1	E617 [SystemTechnologyForecastProfile]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
3811641	1	91137	1	811641	1	999	E616-R-E617
3811642	1	91137	1	811642	1	999	E616-R-E617
3811643	1	91137	1	811643	1	999	E616-R-E617
3811644	1	91137	1	811644	1	999	E616-R-E617
3811645	1	91137	1	811645	1	999	E616-R-E617
3811646	1	91137	1	811646	1	999	E616-R-E617
3811647	1	91137	1	811647	1	999	E616-R-E617
3811648	1	91137	1	811648	1	999	E616-R-E617
3811649	1	91137	1	811649	1	999	E616-R-E617
3811650	1	91137	1	811650	1	999	E616-R-E617
3811651	1	91137	1	811651	1	999	E616-R-E617
3811652	1	91137	1	811652	1	999	E616-R-E617
3811653	1	91137	1	811653	1	999	E616-R-E617
3811654	1	91137	1	811654	1	999	E616-R-E617
3811655	1	91137	1	811655	1	999	E616-R-E617
3811656	1	91137	1	811656	1	999	E616-R-E617
3811657	1	91137	1	811657	1	999	E616-R-E617
3811658	1	91137	1	811658	1	999	E616-R-E617
3811659	1	401	1	811641	1	999	E654-R-E617
3811660	1	402	1	811642	1	999	E654-R-E617
3811661	1	403	1	811643	1	999	E654-R-E617
3811662	1	404	1	811644	1	999	E654-R-E617
3811663	1	405	1	811645	1	999	E654-R-E617
3811664	1	406	1	811646	1	999	E654-R-E617
3811665	1	407	1	811647	1	999	E654-R-E617
3811666	1	408	1	811648	1	999	E654-R-E617

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
3811667	1	409	1	811649	1	999	E654-R-E617
3811668	1	410	1	811650	1	999	E654-R-E617
3811669	1	411	1	811651	1	999	E654-R-E617
3811670	1	412	1	811652	1	999	E654-R-E617
3811671	1	413	1	811653	1	999	E654-R-E617
3811672	1	414	1	811654	1	999	E654-R-E617
3811673	1	415	1	811655	1	999	E654-R-E617
3811674	1	416	1	811656	1	999	E654-R-E617
3811675	1	417	1	811657	1	999	E654-R-E617
3811676	1	418	1	811658	1	999	E654-R-E617

E616-R-E617 = [SV-9] is defined by [*SystemTechnologyForecastProfile*]
E654-R-E617 = [*TechnologyForecast*] defines [*SystemTechnologyForecastProfile*]

4.6.18.4 Net-Centric Requirements

Since the purpose of the SV-9 is to provide a summary of emerging technologies that impact the architecture and its existing planned systems, it would subsequently include any required services or other technologies that support NCO. Accordingly, the CADM support for the SV-9 is well suited to support the NCE.

4.6.19 CADM v1.5 Support for Systems and Services Rules Model, Systems and Services State Transition Description, and Systems Event-Trace Description (SV-10a/b/c)

4.6.19.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-10 products describe dynamic behaviors concerning the timing and sequencing of events that capture system performance characteristics of an executing system (i.e., a system performing the system functions described in SV-4). Behavior modeling and documentation is key to a successful architecture description, because it is how the architecture behaves that is crucial in many situations. Although knowledge of the functions and interfaces is also crucial, knowing whether, for example, a response should be expected after sending message X to node Y can be crucial to successful overall operations.

4.6.19.2 CADM v1.5 Support for Systems Rules Model (SV-10a)

4.6.19.2.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-10a describes the constraints on an architecture, on a system(s), or system hardware/software item(s), and/or on a system function(s). While other SV products (e.g., SV-1, SV-2, SV-4, SV-11) describe the static structure of the Systems View (i.e., what the systems can do), they do not describe, for the most part, what the systems *must* do, or what it *cannot* do.

4.6.19.2.2 High-Level Description

Figure 4-43 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-10a.

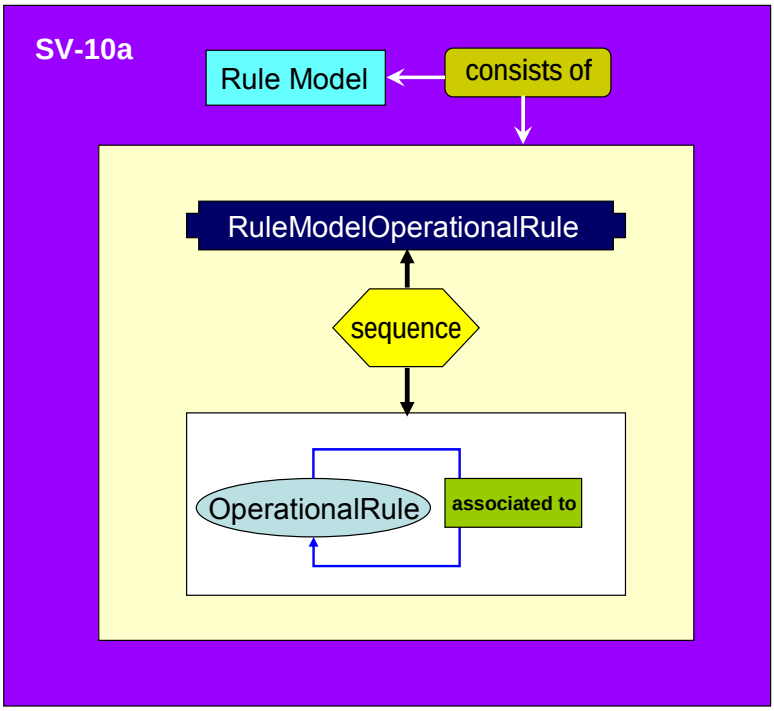


Figure 4-43: High-Level Depiction of CADM v1.5 Data Structures for SV-10a Representation

The DoDAF architecture product SV-10a is expressed in CADM v1.5 as an instance of Document. The SV-10a can be linked to the appropriate instance of Architecture through the associative entity *ArchitectureDocument* (instantiated through ObjectByReference). The actual data content of the SV-10a is built linking it to instances of the associative entity *RuleModelOperationalRule* from CADM v1.03 (instantiated through ObjectByReference), which collects the instances of OperationalRule (a subtype of Guidance) that make up the rule model itself.

4.6.19.2.3 CADM v1.5 Instantiation

The following instance tables show how the SV-10a product is represented in CADM v1.5. The example follows **Figure 4-44**.

If **field A** ***in*** **FORM-X** ***is set to*** **value T,**

Then **field B** ***in*** **FORM-Y** ***must be set to*** **value T**

And **field C** ***in*** **FORM-Z** ***must be set to*** **value T**

End If

Figure 4-44: Notional Example of an SV-10a Product

The instantiation of SV-10a as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
275	E038[Architecture]
276	E148[Document]
277	E679[OBR]
111	E678[OVA]
112	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
275	1	Program Architecture C02	4[ArchElem]
276	1	Notional Systems Rule Model	4[ArchElem]
277	1	ArchitectureDocument (SV-10a in Program Architecture C02)	5[OBR]
111	1	Architecture is documented by SV-10a	3[OVA]
112	1	SV-10a documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
277	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
111	1	275	1	277	1	999	E038-R-E045
112	1	276	1	277	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The instance of Document representing the SV-10a can be linked to each of the required instances of *RuleModelOperationalRule* (expressed through ObjectByReference) which will collect the instances of OperationalRule that make the content of the rule model. Note that *RuleModelOperationalRule* relates an instance of Document corresponding to an SV-10a to multiple instances of OperationalRule.

Object

objectIdentifier	pointerCode
371	E679[OBR]
381	E426[Op Rule]
291	E678 [OVA]
292	E678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
371	1	RuleModelOperationalRule E1-1	5[OBR]
381	1	System Rule 1	4 [ArchElem]
291	1	SV-10a cites Op Rule 1	3 [OVA]
292	1	System Rule 1 is cited for SV-10a	3 [OVA]

ObjectByReference

*Identifier	*Index	categoryCode
371	1	E521[RuleModelOperationalRule]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
291	1	276	1	371	1	999	E520-R-E521
292	1	381	1	371	1	999	E426-R-E521

The relationTypeCode values used have the following meanings:

E520-R-E521 = cites
E426-R-E521 = is cited for

The actual rule is stored in the appropriate attributes of each of the instances of Guidance and OperationalRule. The textual description of the operational rule is recorded in the text of Guidance. A further characterization of the rule can be stated through the attribution of OperationalRule. The instance tables below show how this is done for the notional example discussed at the beginning of this section.

ArchitectureElement

*Identifier	*Index	categoryCode
307	1	27 = GUIDANCE

Guidance

*Identifier	*Index	categoryCode	subject Text	text
307	1	13 = OPERATIONAL RULE	Rule 1 for Rule Model A	**IF** field A in FORM-X is set to value T, **THEN** field B in FORM-Y must be set to value T **AND** field C in FORM-Z must be set to value T End If

OperationalRule

*Identifier	*Index	categoryCode	formalLanguage Name
307	5	6 = CRITERION	Action Assertion Rule

4.6.19.2.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to OperationRule through the instances of ObjectByReference corresponding to *RuleModelOperationalRule RuleModelOperationalRuleSoaService*. To do that, one needs to create the respective instance of *RuleModelOperationalRuleSoaService* and link it to SoaService in the ObjectVersionAssociation table with the relationTypeCode = E682-R-E691 (is governed by) and link it to RuleModelOperationalRule in the ObjectVersionAssociation table with the relationTypeCode = E691-R-E527 (applies to) and set in the respetive instance of RuleModelOperationalRule and link it to OperationalRule in the ObjectVersionAssociation table with the relationTypeCode = E691-R-E521 (is cited for).

4.6.19.3 CADM v1.5 Support for Systems State Transition Description (SV-10b)

4.6.19.3.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-10b is a graphical method of describing a system (or system function) response to various events by changing its state. The diagram basically represents the sets of events to which the systems in the architecture will respond (by taking an action to move to a new state) as a function of its current state. Each transition specifies an event and an action.

4.6.19.3.2 High-Level Description

Figure 4-45 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-10b.

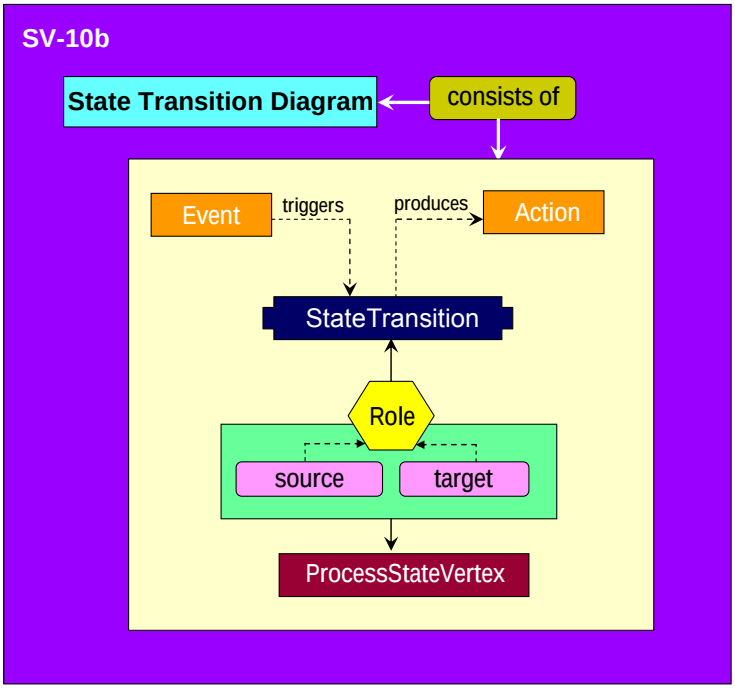


Figure 4-45: High-Level Depiction of CADM v1.5 Data Structures for SV-10b Representation

4.6.19.3.3 CADM v1.5 Instantiation

The following instance tables show how the SV-10b product is represented in CADM v1.5. The example follows **Figure 4-46**.

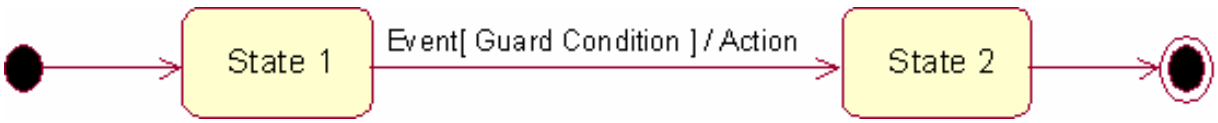


Figure 4-46: Notional Example of an SV-10b Product

The instantiation of SV-10b for this example as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E679[OBR]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	Name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	Notional SV-10b	4[ArchElem]
127	1	ArchitectureDocument (SV-10b in Project X Architecture)	5[OBR]
522	1	Architecture is documented by SV-10b	3[OVA]
523	1	SV-10b documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

For an SV-10b, the following convention is used: Instances of ProcessStateVertex are created for oval elements of the SV-10b diagram and the initial and final states (which are treated as pseudo states).

The instantiation of these states is shown below:

Object

objectIdentifier	pointerCode
307	E502[ProcessStateVertex]
308	E502[ProcessStateVertex]
309	E502[ProcessStateVertex]
310	E502[ProcessStateVertex]

ObjectVersion

*Identifier	*Index	name	categoryCode
307	1	Initial state	4[ArchElem]
308	1	State 1	4[ArchElem]
309	1	State 2	4[ArchElem]
310	1	End State	4[ArchElem]

The actions associated with the pseudo states are those that represent the *entry* and *exit* for the state transition diagram. The tables below show their instantiation and how they are linked to the pseudo states.

Object

objectIdentifier	pointerCode
611	E001[Action]
612	E001[Action]
614	E679[OBR]
615	E679[OBR]
616	E678[OVA]
617	E678[OVA]
618	E678[OVA]
619	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
611	1	Action A001	4[ArchElem]
612	1	Action A002	4[ArchElem]
614	1	ProcessStateAction PSA001	5[OBR]
615	1	ProcessStateAction PSA002	5[OBR]
616	1	Action A001 to Initial state	3[OVA]
617	1	Initial state to Action A001	3[OVA]
618	1	Action A002 to End state	3[OVA]
619	1	End state to Action A002	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
614	1	E501[ProcessStateAction]
615	1	E501[ProcessStateAction]

ObjectByReferenceCharacterization

*Identifier	OBR Identifier	OBR Index	categoryCode	valueText
101	614	1	E501.A01	1
102	614	1	E501.A02	1 (entry)
103	615	1	E501.A01	1
104	615	1	E501.A02	2 (exit)

The categoryCode values used have the following meanings:

E501.A01 = SequenceIdentifierText
E501.A02 = RoleCode

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
616	1	611	1	614	1	999	E001-R-E501
617	1	307	1	614	1	999	E500-R-E501
618	1	612	1	615	1	999	E001-R-E501
619	1	310	1	615	1	999	E500-R-E501

The relationTypeCode values used have the following meanings:

E001-R-E501 = represents
E500-R-E501 = represents

As indicated above, the content of the SV-10b is expressed through the instantiation of TranstionProcess. For each instance of TranstionProcess, one can indicate its source and target states (represented in CADM as instances of ProcessStateVertex), the event that triggers the transition, as well as the system rule that may act as the guard condition for the transition. In CADM v1.5, these links are all represented through entries in the ObjectVersionAssociation table. The attribute labelName in TranstionProcess is used to capture the text that is attached to each of the arrows in the state transition diagram. The instance tables below describe how this is done for the example shown in Figure 4-46 above.

Object

objectIdentifier	pointerCode
701	E663[TransitionProcess]
702	E663[TransitionProcess]
703	E663[TransitionProcess]

ObjectVersion

*Identifier	*Index	name	categoryCode
701	1	TRN001	4[ArchElem]
702	1	TRN002	4[ArchElem]
703	1	TRN003	4[ArchElem]

ArchitectureElement

*Identifier	*Index	categoryCode
701	1	72 = TRANSITION-PROCESS
702	1	72 = TRANSITION-PROCESS
703	1	72 = TRANSITION-PROCESS

TransitionProcess

*Identifier	*Index	labelName
701	1	—
702	1	Event/Action
703	1	—

The intances of ObjectVersionAssociation required to specify the source and target state for each transition are shown below.

Object

objectIdentifier	pointerCode
901	E678[OVA]
902	E678[OVA]
903	E678[OVA]
904	E678[OVA]
905	E678[OVA]
906	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
901	1	PSV[307] -- PSV[308] {source}	E678[OVA]
902	1	PSV[307] -- PSV[308] {target}	E678[OVA]
903	1	PSV[308] -- PSV[309] {source}	E678[OVA]
904	1	PSV[308] -- PSV[309] {target}	E678[OVA]
905	1	PSV[309] -- PSV[310] {source}	E678[OVA]
906	1	PSV[309] -- PSV[310] {target}	E678[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
901	1	307	1	701	1	999	E502-R2-E663
902	1	308	1	701	1	999	E502-R1-E663
903	1	308	1	702	1	999	E502-R2-E663
904	1	309	1	702	1	999	E502-R1-E663
905	1	309	1	703	1	999	E502-R2-E663
906	1	310	1	703	1	999	E502-R1-E663

The relationTypeCode values used have the following meanings:

E502-R2-E663 = is source for
E502-R1-E663 = is target for

The same approach would be used to link the corresponding trigger events to each of the instances of TransitionProcess.

The link between the SV-10b Document and each of the instances of TransitionProcess is done in a similar fashion through ObjectVersionAssociation with the owning Document instance linked to each of the component instances of TransitionProcess.

4.6.19.3.4 Net-Centric Requirements

Since the purpose of the SV-10b is to provide a graphical method of describing a system (or system function) response to various events by changing its state, it would subsequently include any services that support NCO Accordingly, the CADM support for the SV-10b is well suited to support the NCE.

4.6.19.4 CADM v1.5 Support for Systems Event-Trace Description (SV-10c)

4.6.19.4.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-10c provides a time-ordered examination of the system data elements exchanged between participating systems (external and internal), system functions, or human roles as a result of a particular scenario. Each event-trace diagram should have an accompanying description that defines the particular scenario or situation. SV-10c in the Systems View may reflect system-specific aspects or refinements of critical sequences of events described in the OV.

4.6.19.4.2 High-Level Description

Figure 4-47 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-10c.

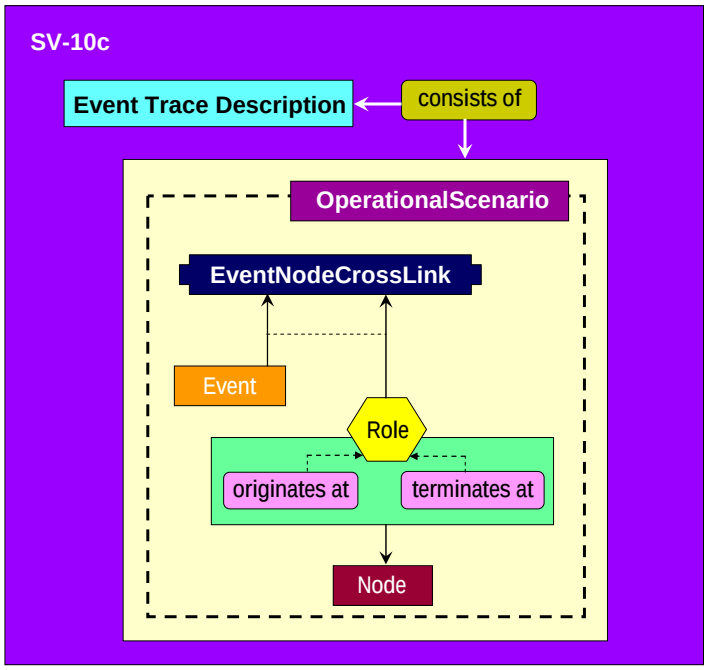


Figure 4-47: High-Level Depiction of CADM v1.5 Data Structures for SV-10c Representation

4.6.19.4.3 CADM v1.5 Instantiation

The following instance tables show how the SV-10c product is represented in CADM v1.5. The example follows **Figure 4-48**.

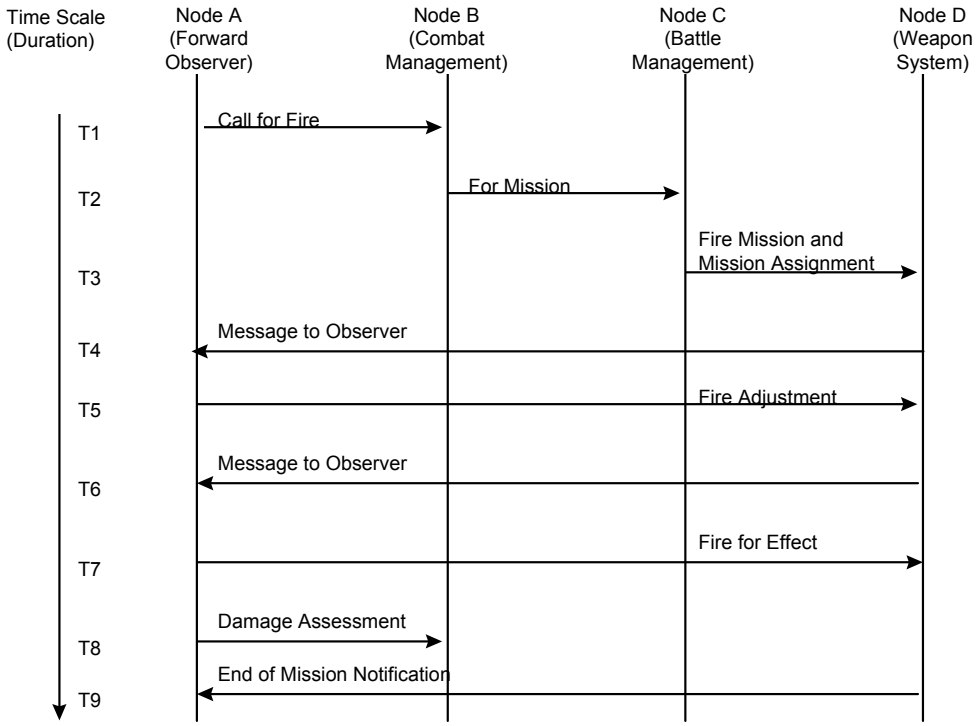


Figure 4-48: Notional Example of an SV-10c Product

The instantiation of SV-10c as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E045[ArchitectureDocument]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	SV-10c	4[ArchElem]
127	1	ArchitectureDocument (SV-10c in Project X Architecture)	3[OVA]
522	1	Architecture is documented by SV-10c	3[OVA]
523	1	SV-10c documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

Object

objectIdentifier	pointerCode
306	E427[OperationalScenario]
307	E159[EventNodeCrosslink]
308	E159[EventNodeCrosslink]
309	E159[EventNodeCrosslink]
310	E159[EventNodeCrosslink]
311	E159[EventNodeCrosslink]
312	E159[EventNodeCrosslink]
313	E159[EventNodeCrosslink]
314	E159[EventNodeCrosslink]
315	E159[EventNodeCrosslink]
316	E359[Node]
317	E359[Node]
318	E359[Node]
319	E359[Node]
320	E156[Event]
321	E156[Event]
322	E156[Event]
323	E156[Event]
324	E156[Event]
325	E156[Event]
326	E156[Event]
327	E156[Event]
328	E156[Event]
564	E678[OVA]
565	E678[OVA]
566	E678[OVA]
567	E678[OVA]
568	E678[OVA]
569	E678[OVA]
570	E678[OVA]
571	E678[OVA]
572	E678[OVA]
573	E678[OVA]
574	E678[OVA]
575	E678[OVA]
576	E678[OVA]
577	E678[OVA]
578	E678[OVA]
579	E678[OVA]
580	E678[OVA]

objectIdentifier	pointerCode
581	E678[OVA]
582	E678[OVA]
583	E678[OVA]
584	E678[OVA]
585	E678[OVA]
586	E678[OVA]
587	E678[OVA]
588	E678[OVA]
589	E678[OVA]
590	E678[OVA]
591	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
306	1	OV-6c Scenario	4[ArchElem]
307	1	ENCL1 [Event 1 from Node A to Node B]	5[OBR]
308	1	ENCL2 [Event 2 from Node B to Node C]	5[OBR]
309	1	ENCL3[Event 3 from Node C to Node D]	5[OBR]
310	1	ENCL4[Event 4 from Node D to Node A]	5[OBR]
311	1	ENCL5[Event 5 from Node A to Node D]	5[OBR]
312	1	ENCL6[Event 6 from Node D to Node A]	5[OBR]
313	1	ENCL7[Event 7 from Node A to Node D]	5[OBR]
314	1	ENCL8[Event 8 from Node A to Node B]	5[OBR]
315	1	ENCL9[Event 9 from Node D to Node A]	5[OBR]
316	1	Node A	4[ArchElem]
317	1	Node B	4[ArchElem]
318	1	Node C	4[ArchElem]
319	1	Node D	4[ArchElem]
320	1	Call for fire	4[ArchElem]
321	1	For mission	4[ArchElem]
322	1	Fire mission and mission assignment	4[ArchElem]
323	1	Message to observer	4[ArchElem]
324	1	Fire adjustment	4[ArchElem]
325	1	Message to observer	4[ArchElem]
326	1	Fire for effect	4[ArchElem]
327	1	Damage Assessment	4[ArchElem]
328	1	End of mission notification	4[ArchElem]
564	1	OV-6c Scenario describes Document	3[OVA]
565	1	Event 1[320] is part of ENCL1[307]	3[OVA]
566	1	ENCL1 from Node A[316]	3[OVA]
567	1	ENCL1 to Node B[317]	3[OVA]
568	1	Event 2[321] is part of ENCL2[308]	3[OVA]
569	1	ENCL2 from Node B[317]	3[OVA]
570	1	ENCL2 to Node C[318]	3[OVA]
571	1	Event 3[320] is part of ENCL3[309]	3[OVA]

*Identifier	*Index	name	categoryCode
572	1	ENCL3 from NodeC[318]	3[OVA]
573	1	ENCL3 to NodeD[319]	3[OVA]
574	1	Event 4[321] is part of ENCL4[310]	3[OVA]
575	1	ENCL4 from NodeD[319]	3[OVA]
576	1	ENCL4 to NodeA[316]	3[OVA]
577	1	Event 5[322] is part of ENCL5[311]	3[OVA]
578	1	ENCL5 from NodeA[316]	3[OVA]
579	1	ENCL5 to NodeD[319]	3[OVA]
580	1	Event 6[323] is part of ENCL 6[312]	3[OVA]
581	1	ENCL6 from Node D[319]	3[OVA]
582	1	ENCL6 to Node A[316]	3[OVA]
583	1	Event 7[324] is part of ENCL7[313]	3[OVA]
584	1	ENCL7 from NodeA[316]	3[OVA]
585	1	ENCL7 to NodeD[319]	3[OVA]
586	1	Event8[325] is part of ENCL8[314]	3[OVA]
587	1	ENCL8 from NodeA[316]	3[OVA]
588	1	ENCL8 to NodeB[317]	3[OVA]
589	1	Event 9[326] is part of ENCL9[315]	3[OVA]
590	1	ENCL9 from NodeD[319]	3[OVA]
591	1	ENCL9 to Nodea[316]	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
564	1	306	1	126	1	999	E148-R-E427
565	1	307	1	320	1	999	E156-R-E159
566	1	307	1	316	1	999	E359-R-E159
567	1	307	1	317	1	999	E359-R-E159
568	1	308	1	321	1	999	E156-R-E159
569	1	308	1	317	1	999	E359-R-E159
570	1	308	1	318	1	999	E359-R-E159
571	1	309	1	322	1	999	E156-R-E159
572	1	309	1	318	1	999	E359-R-E159
573	1	309	1	319	1	999	E359-R-E159
574	1	310	1	323	1	999	E156-R-E159
575	1	310	1	319	1	999	E359-R-E159
576	1	310	1	316	1	999	E359-R-E159
577	1	311	1	324	1	999	E156-R-E159
578	1	311	1	316	1	999	E359-R-E159
579	1	311	1	319	1	999	E359-R-E159
580	1	312	1	325	1	999	E156-R-E159
581	1	312	1	319	1	999	E359-R-E159
582	1	312	1	316	1	999	E359-R-E159
583	1	313	1	326	1	999	E156-R-E159
584	1	313	1	316	1	999	E359-R-E159
585	1	313	1	319	1	999	E359-R-E159
586	1	314	1	327	1	999	E156-R-E159
587	1	314	1	316	1	999	E359-R-E159
588	1	314	1	317	1	999	E359-R-E159
589	1	315	1	328	1	999	E156-R-E159
590	1	315	1	319	1	999	E359-R-E159

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
597	1	315	1	316	1	999	E359-R-E159

The relationTypeCode values used have the following meanings:

E148-R-E427 = describes
E156-R-E159 = is crosslink for
E359-R-E159 = is the originator for
E359-R-E159 = is the terminator for

4.6.19.4.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to Node through the characterization of NodeSoaService. To do that, one needs to set in the respective instance of NodeSoaService and link it to SoaService in the ObjectVersionAssociation table with the relationTypeCode = E682-R-E685 (supports the functions of) and link it to Node in the ObjectVersionAssociation table with the relationTypeCode = E359-R-E685 (is supported by).

4.6.20 CADM v1.5 Support for Physical Schema (SV-11)

4.6.20.1 Product Definition

As stated in DoDAF v1.5 Volume II, the SV-11 is one of the architecture products closest to the actual system design in the Framework. The product defines the structure of the various kinds of system data that are utilized by the systems in the architecture.

4.6.20.2 High-Level Description

Figure 4-49 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of an SV-11.

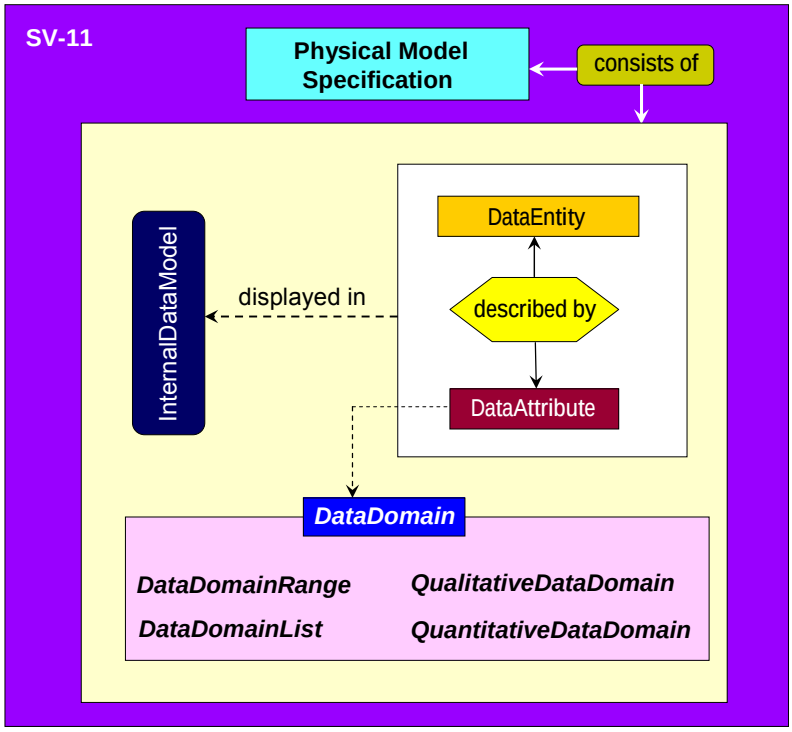


Figure 4-49: High-Level Depiction of CADM v1.5 Data Structures for SV-11 Representation

The DoDAF architecture product SV-11 is expressed in CADM v1.5 as an instance of Document. The SV-11 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference).

The actual data content of the SV-11 is built by linking it to one or more instances of UserPresentationView. This allows the specification of a physical structure, which may differ somewhat from the logical data model (e.g., where denormalization or materialized views are created for implementation). Where the physical and logical models remain tightly coupled (i.e., there is a one-to-one correspondence between the logical entities and attributes and the corresponding tables and columns) the physical model can be built essentially in the same way as the OV-7. The constraints on data types and valid domain ranges are expressed though the CADM v1.03 entity *DataDomain* and its sutypes. The abbreviatedName for each InformationAsset can be used to capture the table and column names respectively.

4.6.20.3 CADM v1.5 Instantiation

The following instance tables show how the SV-11 product is represented in CADM v1.5. The example follows **Figure 4-50**.

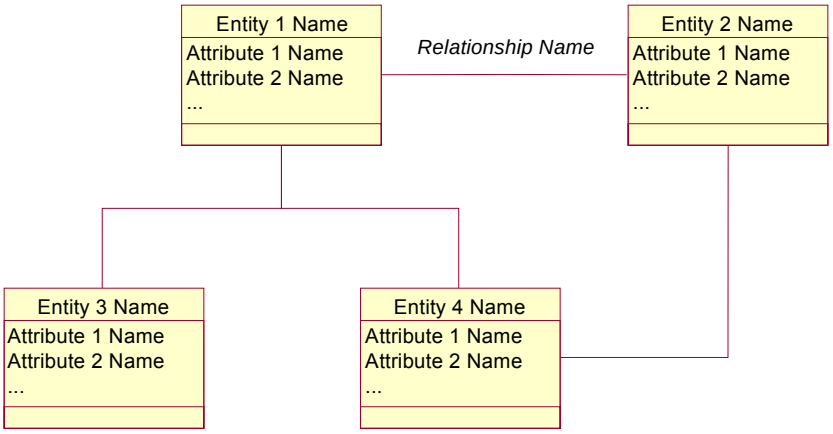


Figure 4-50 Notional Example of an SV-10c Product

The instantiation of SV-11 as Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E045[ArchitectureDocument]
522	E678[OVA]
523	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	SV-11	4[ArchElem]
127	1	ArchitectureDocument (SV-11 in Project X Architecture)	3[OVA]
522	1	Architecture is documented by SV-11	3[OVA]
523	1	SV-11 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
522	1	125	1	127	1	999	E038-R-E045
523	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The next step is to relate the SV-11 to the instances of InformationAsset that corresponds to the actual content of the model. The pertinent subtypes are DataEntity, DataAttribute and ConceptualDataModel.

The instance tables below show how this is expressed in CADM v1.5.

Object

objectIdentifier	pointerCode
214	E112[ConceptualModel]
215	E215[ConceptualDataModelView]
216	E114[IConceptualDataModelViewDataEntity]
217	E114[IConceptualDataModelViewDataEntity]
218	E114[IConceptualDataModelViewDataEntity]
219	E114[IConceptualDataModelViewDataEntity]
220	E136[DataEntityRelationship]
221	E136[DataEntityRelationship]
222	E136[DataEntityRelationship]
223	E136[DataEntityRelationship]
315	E133[DataEntity]
316	E133[DataEntity]
317	E133[DataEntity]
318	E133[DataEntity]
321	E118[DataAttribute]
322	E118[DataAttribute]
323	E118[DataAttribute]
324	E118[DataAttribute]
325	E118[DataAttribute]
326	E118[DataAttribute]
327	E118[DataAttribute]
328	E118[DataAttribute]
622	E678[OVA]
623	E678[OVA]
624	E678[OVA]
625	E678[OVA]
626	E678[OVA]
627	E678[OVA]
628	E678[OVA]
629	E678[OVA]
630	E678[OVA]
631	E678[OVA]
632	E678[OVA]
633	E678[OVA]
634	E678[OVA]
635	E678[OVA]
636	E678[OVA]
637	E678[OVA]
638	E678[OVA]
639	E678[OVA]
640	E678[OVA]
641	E678[OVA]
642	E678[OVA]
643	E678[OVA]
644	E678[OVA]
645	E678[OVA]
646	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
214	1	Physical Data Model SV-11 Template	4[ArchElem]
215	1	Physical Data Model SV-11 View 1	4[ArchElem]
216	1	CDMVDE01[Entity 1 in SV-11(215)]	5[OBR]
217	1	CDMVDE02[Entity 2 in SV-11(215)]	5[OBR]
218	1	CDMVDE03[Entity 3 in SV-11(215)]	5[OBR]
219	1	CDMVDE04[Entity 4 in SV-11(215)]	5[OBR]
220	1	DER1[Entity 1 to Entity 2]	5[OBR]
221	1	DER2[Entity 1 to Entity 3]	5[OBR]
222	1	DER3[Entity 1 to Entity 4]	5[OBR]
223	1	DER4[Entity 2 to Entity 4]	5[OBR]
315	1	Entity 1	4[ArchElem]
316	1	Entity 2	4[ArchElem]
317	1	Entity 3	4[ArchElem]
318	1	Entity 4	4[ArchElem]
321	1	Attribute 1 for Entity 1	4[ArchElem]
322	1	Attribute 2 for Entity 1	4[ArchElem]
323	1	Attribute 1 for Entity 2	4[ArchElem]
324	1	Attribute 2 for Entity 2	4[ArchElem]
325	1	Attribute 1 for Entity 3	4[ArchElem]
326	1	Attribute 2 for Entity 3	4[ArchElem]
327	1	Attribute 1 for Entity 4	4[ArchElem]
328	1	Attribute 2 for Entity 4	4[ArchElem]
622	1	DER1 from Entity 1	3[OVA]
623	1	DER1 to Entity 2	3[OVA]
624	1	DER2 from Entity 1	3[OVA]
625	1	DER2 to Entity 3	3[OVA]
626	1	DER3 from Entity 1	3[OVA]
627	1	DER3 to Entity 4	3[OVA]
628	1	DER4 from Entity 2	3[OVA]
629	1	DER4 to Entity 4	3[OVA]
630	1	SV-11 View 1 is in SV-11 Template	3[OVA]
631	1	CDMVDE01 is in SV-11 View1[215]	3[OVA]
632	1	Entity 1 is part of CDMVDE01	3[OVA]
633	1	Attribute 1 is part of Entity 1	3[OVA]
634	1	Attribute 2 is part of Entity 1	3[OVA]
635	1	CDMVDE02 is in SV-11[215]	3[OVA]
636	1	Entity 2 is part of CDMVDE02	3[OVA]
637	1	Attribute 1 is part of Entity 2	3[OVA]
638	1	Attribute 2 is part of Entity 2	3[OVA]
639	1	CDMVDE03 is in SV-11[215]	3[OVA]
640	1	Entity 3 is part of CDMVDE03	3[OVA]
641	1	Attribute 1 is part of Entity 3	3[OVA]
642	1	Attribute 2 is part of Entity 3	3[OVA]
643	1	CDMVDE04 is in SV-11[215]	3[OVA]
644	1	Entity 4 is part of CDMVDE04	3[OVA]
645	1	Attribute 1 is part of Entity 4	3[OVA]
646	1	Attribute 2 is part of Entity 4	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
622	1	622	1	315	1	999	E133-R-E136
623	1	622	1	316	1	999	E133-R-E136
624	1	623	1	315	1	999	E133-R-E136
625	1	623	1	317	1	999	E133-R-E136
626	1	624	1	315	1	999	E133-R-E136
627	1	624	1	318	1	999	E133-R-E136
628	1	625	1	316	1	999	E133-R-E136
629	1	625	1	318	1	999	E133-R-E136
630	1	215	1	214	1	999	E112-R-E113
631	1	216	1	215	1	999	E113-R-E114
632	1	216	1	315	1	999	E133-R-E114
633	1	321	1	315	1	999	E133-R-E118
634	1	322	1	315	1	999	E133-R-E118
635	1	217	1	215	1	999	E113-R-E114
636	1	217	1	316	1	999	E133-R-E114
637	1	323	1	316	1	999	E133-R-E118
638	1	324	1	316	1	999	E133-R-E118
639	1	218	1	215	1	999	E113-R-E114
640	1	218	1	317	1	999	E133-R-E114
641	1	325	1	317	1	999	E133-R-E118
642	1	326	1	317	1	999	E133-R-E118
643	1	219	1	215	1	999	E113-R-E114
644	1	219	1	318	1	999	E133-R-E114
645	1	327	1	318	1	999	E133-R-E118
646	1	328	1	318	1	999	E133-R-E118

The relationTypeCode values used have the following meanings:

E133-R-E136 = is ordinate of
E133-R-E136 = is subordinate of
E112-R-E113 = is represented in
E113-R-E114 = displays
E133-R-E114 = is displayed in
E133-R-E118 = is described by

4.6.20.4 Net-Centric Requirements

The specification of discovery metadata at the system and service level can be expressed in CADM v1.5 through DiscoveryMetadata, which is linkable to DiscoveryMetadataDocument, which is linkable to InformationAsset through the characterization of InformationAssetDocument. To do that, one needs to set in the respective instance of DiscoveryMetadataDocument and link it to DiscoveryMetadata in the ObjectVersionAssociation table with the relationTypeCode = E147-R-E152 (is used to discover) and InformationAssetDocument in the ObjectVersionAssociation table with the relationTypeCode = E152-R-E217 (may apply to) and set in the respective instance of InformationAsset and link it to InformationAssetDocument in the ObjectVersionAssociation table with the relationTypeCode = E215-R-E217 (is cited in).

4.6.21 CADM v1.5 Support for Technical Standards Profile (TV-1)

4.6.21.1 Product Definition

As stated in DoDAF v1.5 Volume II, the TV-1 collects the various systems standards rules that implement and sometimes constrain the choices that can be made in the design and implementation of an architecture.

4.6.21.2 High-Level Description

Figure 4-51 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of a TV-1.

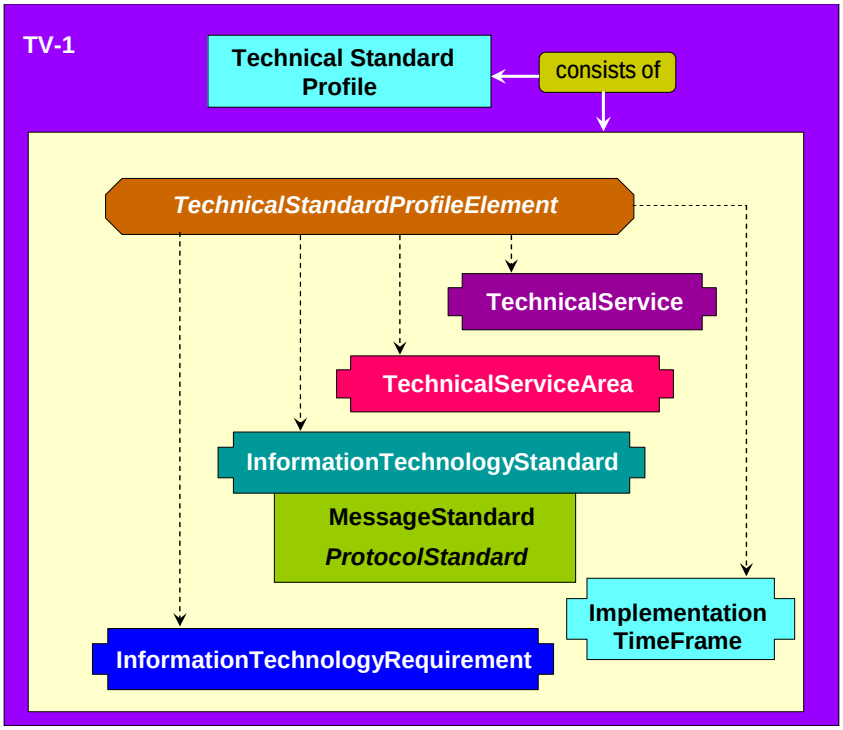


Figure 4-51: High-Level Depiction of CADM v1.5 Data Structures for TV-1 Representation

The DoDAF architecture product TV-1 is expressed in CADM v1.5 as an instance of Document. The TV-1 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference).

The actual data content of the TV-1 is built by linking it to one or more instances of *TechnicalStandardProfileElement*. The latter can be linked to InformationTechnologyStandard, Guidance (e.g., InformationTechnologyRequirement, *TechnicalGuideline*), TechnicalService, and ImplementationTimeFrame to build the specification of the profile.

4.6.21.3 CADM v1.5 Instantiation

The following instance tables show how the TV-1 product is represented in CADM v1.5. The example follows **Figure 4-52**.

Service Area	Service	Standard
Operating System	Kernel	FIPS Pub 151-1 (POSIX.1)
	Shell and Utilities	IEEE P1003.2
Software Engineering Services	Programming Languages	FIPS Pub 119 (ADA)
User Interface	Client Server Operations	FIPS Pub 158 (X-Window System)
	Object Definition and Management	DoD Human Computer Interface Style Guide
	Window Management	FIPS Pub 158 (X-Window System)
	Dialogue Support	Project Standard
Data Management	Data Management	FIPS Pub 127-2 (SQL)
Data Interchange	Data Interchange	FIPS Pub 152 (SGML)
	Electronic Data Interchange	FIPS Pub 161 (EDI)
Graphics	Graphics	FIPS Pub 153 (PHIGS)
• • •		

Figure 4-52: Notional Example of a TV-1 Product

The TV-1 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E679[OBR]
601	E678[OVA]
602	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	TV-1	4[ArchElem]
127	1	ArchitectureDocument (TV-1 in Program Architecture[126]	5[OBR]
601	1	Architecture is documented by TV-1	3[OVA]
602	1	TV-1 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
601	1	125	1	127	1	999	E038-R-E045
602	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

Object

objectIdentifier	pointerCode
315	E645[TechnicalServiceArea]
316	E645[TechnicalServiceArea]
317	E645[TechnicalServiceArea]
318	E645[TechnicalServiceArea]
319	E645[TechnicalServiceArea]
320	E645[TechnicalServiceArea]
415	E644[TechnicalService]
416	E644[TechnicalService]
417	E644[TechnicalService]
418	E644[TechnicalService]
419	E644[TechnicalService]
420	E644[TechnicalService]
421	E644[TechnicalService]
422	E644[TechnicalService]
423	E644[TechnicalService]
424	E644[TechnicalService]
425	E644[TechnicalService]
515	E260[InformationTechnologyStandard]
516	E260[InformationTechnologyStandard]
517	E260[InformationTechnologyStandard]
518	E260[InformationTechnologyStandard]
519	E260[InformationTechnologyStandard]
520	E260[InformationTechnologyStandard]
521	E260[InformationTechnologyStandard]
522	E260[InformationTechnologyStandard]
523	E260[InformationTechnologyStandard]
524	E260[InformationTechnologyStandard]
525	E260[InformationTechnologyStandard]
701	679 [OBR]
702	679 [OBR]
703	679 [OBR]
801	678 [OVA]
802	678 [OVA]
803	678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
315	1	Operating System	4[ArchElem]
316	1	Software Engineering Services	4[ArchElem]
317	1	User Interface	4[ArchElem]
318	1	Data Management	4[ArchElem]
319	1	Data Interchange	4[ArchElem]
320	1	Graphics	4[ArchElem]
415	1	Kernel	4[ArchElem]
416	1	Shell and Utilities	4[ArchElem]
417	1	Programming Languages	4[ArchElem]
418	1	Client Server Operations	4[ArchElem]
419	1	Object Definition and Management	4[ArchElem]
420	1	Window Management	4[ArchElem]
421	1	Dialogue Support	4[ArchElem]
422	1	Data Management	4[ArchElem]

423	1	Data Interchange	4[ArchElem]
424	1	Electonic Data Interchange	4[ArchElem]
425	1	Graphics	4[ArchElem]
515	1	FIPS Pub 151-1	4[ArchElem]
516	1	IEEE P1003.2	4[ArchElem]
517	1	FIPS Pub 119	4[ArchElem]
518	1	FIPS Pub 158	4[ArchElem]
519	1	DoD Human Computer Interface Style Guide	4[ArchElem]
520	1	FIPS Pub 158	4[ArchElem]
521	1	Project Standard	4[ArchElem]
522	1	FIPS Pub 127-2	4[ArchElem]
523	1	FIPS Pub 152	4[ArchElem]
524	1	FIPS Pub 161	4[ArchElem]
525	1	FIPS Pub 153	4[ArchElem]
701	1	Technical Standard Profile Element 1	5[OBR]
702	1	Technical Standard Profile Element 2	5[OBR]
703	1	Technical Standard Profile Element 3	5[OBR]
801	1	TV-1[126[contains TSPE 1[701]	3[OVA]
802	1	TV-1[126[contains TSPE 2[702]	3[OVA]
803	1	TV-1[126[contains TSPE 3[703]	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
701	1	E649[TSPE]
702	1	E649[TSPE]
703	1	E649[ITSPE]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
801	1	701	1	126	1	999	E648-R-E649
802	1	702	1	126	1	999	E648-R-E649
803	1	703	1	126	1	999	E648-R-E649

The relationTypeCode values used have the following meanings:

E648-R-E649 = **comprises**

Finally, each *TechnicalStandardProfileElement* can be linked to the pertinent instances of TechnicalServiceArea, TechnicalService, and InformationTechnologyStandard. For the purpose of illustration, one can take an instance of TechnicalService and create a new instance of OVA.

Object

objectIdentifier	pointerCode
415	E644[TechnicalService]
901	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
415	1	Kernel	4[ArchElem]
901	1	TechnicalService[415]] toTSPE1[701]	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
901	1	415	1	701	1	999	E644-R-E649

The relationTypeCode values used have the following meanings:

E644-R-E649 = is the focus of

4.6.21.4 Net-Centric Requirements

The specification of services at the system and service level can be expressed in CADM v1.5 through SoaService, which is linkable to InformationTechnologyStandard through the characterization of SoaServiceInformationTechnologyStandard. To do that, one needs to set in the respective instance of SoaServiceInformationTechnologyStandard and link it to SoaService in the ObjectVersionAssociation table with the relationTypeCode = E563-R-E585 (is used in) and link it to InformationTechnologyStandard in the ObjectVersionAssociation table with the relationTypeCode = E260-R-E585 (uses).

4.6.22 CADM v1.5 Support for Technical Standards Forecast (TV-2)

4.6.22.1 Product Definition

As stated in DoDAF v1.5 Volume II, the TV-2 contains expected changes in technology-related standards and conventions, which are documented in the TV-1 product. The forecast for evolutionary changes in the standards should be correlated against the time periods as mentioned in the SV-8 and SV-9 products.

4.6.22.2 High-Level Description

Figure 4-53 shows a high-level conceptual depiction of the CADM v1.5 data structures that support the description of a TV-2.

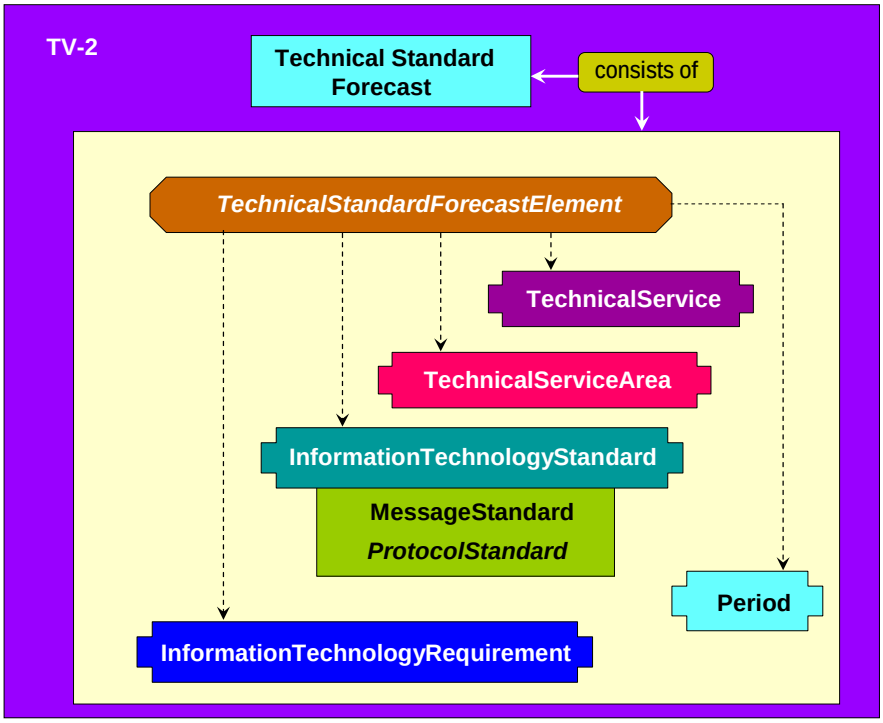


Figure 4-53: High-Level Depiction of CADM v1.5 Data Structures for TV-2 Representation

The DoDAF architecture product TV-2 is expressed in CADM v1.5 as an instance of Document. The TV-2 document can be connected to the appropriate instance of Architecture that it is part of through the CADM v1.03 entity *ArchitectureDocument* (modeled as ObjecByReference).

The actual **data content** of the TV-2 is built by linking it to one or more instances of *TechnicalStandardForecastElement*. The latter can be linked to Period, and TechnicalService, TechnicalServiceArea to create the applicable standard forecast.

4.6.22.3 CADM v1.5 Instantiation

The following instance tables show how the TV-2 product is represented in CADM v1.5. The example follows **Figure 4-54**.

Service Areas	Service	Status	As of 6/93	Expected by 12/93	Expected by 12/94	Expected by 12/94	Comments
Operating System	Kernel	Now	FIPS PUB 151-1	FIPS PUB 151-2			
	Shell & Utilities	Now	IEEE 1003.2	FIPS Addition			
	Real Time Extension	Future	IEEE 1003.4	FIPS Addition			
Program-ming	Program-ming Language	Now	FIPS PUB 119 - Ada		FIPS PUB 119-1 Ada9X		
	CASE Tools & Environ-ment	Now	ECMA Spec 149 - PCTE				
User Interface	• • •						
Data Manage-ment	Data-Diction-ary/Direct-ory	Now	FIPS PUB 156 - IRDS				
	Data Manage-ment	Now	FIPS PUB 127-1-SQL		FIPS PUB 127-2-SQL+	FIPS PUB 127-3-SQL++	
• • •							

Figure 4-54: Notional Example of a TV-2 Product

The TV-2 as an instance of Document and its relation to an appropriate instance of Architecture is shown below.

Object

objectIdentifier	pointerCode
125	E038[Architecture]
126	E148[Document]
127	E679[OBR]
601	E678[OVA]
602	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
125	1	Project X Architecture	4[ArchElem]
126	1	TV-2	4[ArchElem]
127	1	ArchitectureDocument (TV-2 in Program Architecture[126]	5[OBR]
601	1	Architecture is documented by TV-2	3[OVA]
602	1	TV-1 documents Architecture	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
127	1	E045[ArchitectureDocument]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
601	1	125	1	127	1	999	E038-R-E045
602	1	126	1	127	1	999	E148-R-E045

The relationTypeCode values used have the following meanings:

E038-R-E045 = is recorded in
E148-R-E045 = records

The content of the TV-2 is created by linking the appropriate instances of Period, TechnicalServiceArea, TechnicalService and the applicable standards to instances of ObjectByReference corresponding to the CADM v1.03 entity *TechnicalStandardForecastElement*.

Object

objectIdentifier	pointerCode
215	E467[Period]
216	E467[Period]
217	E467[Period]
218	E467[Period]
219	E467[Period]
220	E467[Period]
701	647 [OBR]
702	647 [OBR]
703	647 [OBR]
704	647 [OBR]
705	647 [OBR]
706	647 [OBR]
707	647 [OBR]
708	647 [OBR]
315	E645[TechnicalServiceArea]
316	E645[TechnicalServiceArea]
317	E645[TechnicalServiceArea]
318	E645[TechnicalServiceArea]
319	E645[TechnicalServiceArea]
320	E645[TechnicalServiceArea]
321	E645[TechnicalServiceArea]
415	E644[TechnicalService]
416	E644[TechnicalService]
417	E644[TechnicalService]
418	E644[TechnicalService]
419	E644[TechnicalService]
420	E644[TechnicalService]
421	E644[TechnicalService]
422	E644[TechnicalService]
423	E644[TechnicalService]
424	E644[TechnicalService]
425	E644[TechnicalService]
515	E260[InformationTechnologyStandard]
516	E260[InformationTechnologyStandard]
517	E260[InformationTechnologyStandard]
518	E260[InformationTechnologyStandard]
519	E260[InformationTechnologyStandard]
520	E260[InformationTechnologyStandard]
521	E260[InformationTechnologyStandard]
522	E260[InformationTechnologyStandard]
523	E260[InformationTechnologyStandard]
524	E260[InformationTechnologyStandard]
525	E260[InformationTechnologyStandard]

objectIdentifier	pointerCode
526	E260[InformationTechnologyStandard]
527	E260[InformationTechnologyStandard]
528	E260[InformationTechnologyStandard]
529	E260[InformationTechnologyStandard]
801	678 [OVA]
802	678 [OVA]
803	678 [OVA]
804	678 [OVA]
805	678 [OVA]
806	678 [OVA]
807	678 [OVA]
808	678 [OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
215	1	Base	4[ArchElem]
216	1	Future	4[ArchElem]
217	1	Future	4[ArchElem]
218	1	Future	4[ArchElem]
219	1	Long Range	4[ArchElem]
220	1	Long Range	4[ArchElem]
701	1	Technical Standard Forecast Element 1	5[OBR]
702	1	Technical Standard Forecast Element 2	5[OBR]
703	1	Technical Standard Forecast Element 3	5[OBR]
704	1	Technical Standard Forecast Element 4	5[OBR]
705	1	Technical Standard Forecast Element 5	5[OBR]
706	1	Technical Standard Forecast Element 6	5[OBR]
707	1	Technical Standard Forecast Element 7	5[OBR]
708	1	Technical Standard Forecast Element 8	5[OBR]
315	1	Information Processing	4[ArchElem]
316	1	Information Transfer	4[ArchElem]
317	1	Info Modeling, Metadata, and Info Exchange	4[ArchElem]
318	1	Human-Computer Interface	4[ArchElem]
319	1	Information Security	4[ArchElem]
320	1	Combat Support Information	4[ArchElem]
321	1	Sensor Exploitation	4[ArchElem]
415	1	1 Base-5	4[ArchElem]
416	1	10 BASE -F	4[ArchElem]
417	1	10 Base-2	4[ArchElem]
418	1	10 Base-5	4[ArchElem]
419	1	10 Base-Broad	4[ArchElem]
420	1	10 Base-FL	4[ArchElem]
421	1	10 Base-T	4[ArchElem]
422	1	10 BROAD 36	4[ArchElem]
423	1	10.005MHz-10.100MHz	4[ArchElem]
424	1	100 BASE-F	4[ArchElem]
425	1	100 Base-FX	4[ArchElem]
515	1	FIPS Pub 151-1	4[ArchElem]
516	1	IEEE P1003.2	4[ArchElem]
517	1	FIPS Pub 119	4[ArchElem]
518	1	FIPS Pub 158	4[ArchElem]

519	1	DoD Human Computer Interface Style Guide	4[ArchElem]
520	1	Project Standard	4[ArchElem]
521	1	FIPS Pub 127-2	4[ArchElem]
522	1	FIPS Pub 152	4[ArchElem]
523	1	FIPS Pub 161	4[ArchElem]
524	1	FIPS Pub 153	4[ArchElem]
525	1	FIPS Pub 151-2	4[ArchElem]
526	1	IEEE P1003.2 Add'n	4[ArchElem]
527	1	IEEE P1003.4	4[ArchElem]
528	1	IEEE P1003.4 Add'n	4[ArchElem]
529	1	FIPS 119, Ada9X	4[ArchElem]
801	1	TV-2[126[contains TSFE 1[701]	3[OVA]
802	1	TV-2[126[contains TSFE 2[702]	3[OVA]
803	1	TV-2[126[contains TSFE 3[703]	3[OVA]
804	1	TV-2[126[contains TSFE 1[704]	3[OVA]
805	1	TV-2[126[contains TSFE 2[705]	3[OVA]
806	1	TV-2[126[contains TSFE 3[706]	3[OVA]
807	1	TV-2[126[contains TSFE 2[707]	3[OVA]
808	1	TV-2[126[contains TSFE 3[708]	3[OVA]

ObjectByReference

*Identifier	*Index	categoryCode
701	1	E647[TSPE Matrix Element]
702	1	E647[TSPE Matrix Element]
703	1	E647[ITSPE Matrix Element]
704	1	E647[TSPE Matrix Element]
705	1	E647[TSPE Matrix Element]
706	1	E647[ITSPE Matrix Element]
707	1	E647[TSPE Matrix Element]
708	1	E647[ITSPE Matrix Element]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
801	1	701	1	126	1	999	E646-R-E647
802	1	702	1	126	1	999	E646-R-E647
803	1	703	1	126	1	999	E646-R-E647
804	1	704	1	126	1	999	E646-R-E647
805	1	705	1	126	1	999	E646-R-E647
806	1	706	1	126	1	999	E646-R-E647
807	1	707	1	126	1	999	E646-R-E647
808	1	708	1	126	1	999	E646-R-E647

The relationTypeCode values used have the following meanings:

E646-R-E647= **predicts**

Finally, each *TechnicalStandardForecastElement* can be linked to the pertinent instances of TechnicalService, *Timeframe*, Period, and InformationTechnologyStandard. For the purpose of illustration, one can take an instance of TechnicalService and create a new instance of OVA.

Object

objectIdentifier	pointerCode
415	E644[TechnicalService]
901	E678[OVA]

ObjectVersion

*Identifier	*Index	name	categoryCode
415	1	1-Base 5	4[ArchElem]
901	1	TechnicalService[415]] toTSFE1[701]	3[OVA]

ObjectVersionAssociation

*Identifier	*Index	subject OV Identifier	subject OV Index	object OV Identifier	object OV Index	category Code	relationType Code
901	1	701	1	415	1	999	E644-R-E647

The relationTypeCode values used have the following meanings:

E644-R-E647 = is referenced in

4.6.22.4 Net-Centric Requirements

Since the purpose of the TV-2 is to identify critical technology standards, their fragility, and the impact of these standards on the future development and maintainability of the architecture and its constituent elements, it would subsequently include any required services or other technologies that support NCO. Accordingly, the CADM support for the TV-2 is well suited to support the NCE.

ANNEX A
GLOSSARY

A&I	Architectures and Interoperability
A&ID	Architectures and Interoperability Directorate
ACCB	Architecture Configuration Control Board
AIP	Architecture Interoperability Program
ASD(C3I)	Assistant Secretary of Defense (Command, Control, Communications, and Intelligence)
AT&L	Acquisition, Technology, and Logistics
AV	All View
AWG	Architecture Working Group
BRM	Business Reference Model
C/S/As	Commands, Services, and Agencies
C2	Command and Control
C3	Command, Control, Communications
C3	Command, Control, and Consultation
C3I	Command, Control, Communications, and Intelligence
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CADM	Core Architecture Data Model
CES	Core Enterprise Services
CIO	Chief Information Officer
CJCS	Chairman Joint Chiefs of Staff
CJCSI	Chairman Joint Chiefs of Staff Instruction
COAL	Common Operational Activities List
COCOM	Combatant Command
COI	Community of Interest

COTS	Commercial, Off-the-Shelf
CRD	Capstone Requirements Document
CSFL	Common System Function List
DARS	DoD Architecture Registry System
DBMS	Database Management System
DDDS	DoD Data Dictionary System
DDL	Data Definition Language
DDMS	DoD Discovery Metadata Specification
DISR	DoD IT Standards Registry
DML	Data Manipulation Language
DMR	DoD Metadata Registry
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DoDD	DoD Directive
DoDI	DoD Instruction
DSS	Decision Support System
EA	Enterprise Architecture
EID	Enterprise Identifier
FIPS	Federal Information Processing Standard
FJAWG	Federated Joint Architecture Working Group
GIG	Global Information Grid
GOTS	Government, Off-the-shelf
IDEF0	Integrated Definition for Activity Modeling
IDEF1X	Integrated Definition for Data Modeling
IER	Information Exchange Requirement

IT	Information Technology
ITMRA	Information Technology Management Reform Act - Clinger-Cohen Act of 1996
JCA	Joint Capability Areas
JCS	Joint Chiefs of Staff
JMA	Joint Mission Area
JTF	Joint Task Force
M&S	Modeling and Simulation
MCP	Mission Capability Package
MS	Microsoft
NATO	North Atlantic Treaty Organization
NCDS	Net-Centric Data Strategy
NCE	Net-Centric Environment
NCES	Net-Centric Enterprise Services
NCO	Net-Centric Operations
NCOE	Net-Centric Operating Environment
NCW	Net-Centric Warfare
NII	Networks and Information Integration
NRO	National Reconnaissance Office
OASD	Office of the Assistant Secretary of Defense
OMG	Object Management Group
OSD	Office of the Secretary of Defense
OV	Operational View
PM	Program Manager
RDBMS	Relational Database Management System
SECDEF	Secretary of Defense

SOA	Service Oriented Architecture
SME	Subject Matter Expert
SOCOM	Southern Command
SQL	Structured Query Language
SV	Systems and Services View
TV	Technical Standards View
UJTL	Universal Joint Task List
UML	Unified Modeling Language
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language

ANNEX B
DICTIONARY OF TERMS

The terms included here are terms that are used in some restrictive or special sense in this document. Certain terms are not defined (e.g., event, function) because they have been left as primitives, and the ordinary dictionary usage should be assumed. Where the source for a definition is known, the reference has been provided in parentheses following the definition. Terms that are being used by both the Framework and the C4ISR CADM are marked with an asterisk.

Access Control Level	A restriction on the visibility/accessibility of data and metadata. *Public access* level allows visibility/accessibility of artifact metadata and data to all authenticated users. *Protected access* allows visibility of artifact metadata to all authenticated users, but restricts data access to designated users. *Private access* level restricts visibility of metadata and data access to designated users.
Activity	As used in the Framework, an activity is an action performed in conducting the business of an enterprise. It is a general term that does not imply a placement in a hierarchy (that is, it could be a process or a task as defined in other documents and it could be at any level of the hierarchy of the OV-5). It is used to portray operational actions, not hardware/software system functions. (DoDAF)
Analysis	a : A prescribed approach to developing alternative solutions to a defined problem b : In the JCIDS Overlay, for example, the analyses include, but are certainly not limited to, FAA, FNA, FSA, Gap Analysis, Risk Analysis, and so on.
Architecture Artifact	An aggregation of architecture data, structured or unstructured. Examples include integrated architectures or DoDAF views in any format (e.g., .ppt, .doc, .xls, .jpg, .xml), tables of data records, or individual or groups of records representing architecture object instances. An *Architecture Artifact* is a type of Data Asset.
Attribute*	A quantitative or qualitative characteristic of an element or its actions. (CJCSI 3170.01E, 11 MAY 2005)
Authoritative Source	A designation given to a data source by an appropriate authority indicating that the source data is definitive and preferred or mandated for use.
Certification	Affirmation by an appropriate authority (e.g., DoD CIO) of compliance with specified program control elements.
Classification Taxonomy	A set of upper tier classification categories that are managed at the DoD enterprise level, identified as authoritative, and mandated for use.
Communications Medium*	A means of data transmission.

Community of Interest	Collaborative groups of users who must exchange information in pursuit of their shared goals, interests, missions, or business processes and who therefore must have shared vocabulary for the information they exchange. (DoD NCDS, 9 May 2003)
Configuration Management	Configuration management, applied over the life cycle of a product, provides visibility and control of its performance, functional, and physical attributes. Configuration management verifies that a product performs as intended, and is identified and documented in sufficient detail to support its projected life cycle (e.g., fabrication or production, operation, maintenance, repair, replacement, and disposal). (EIA 649, National Consensus Standard for Configuration Management)
Data	A representation of individual facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means. (IEEE 610.12)
Data Asset	Data asset refers to any entity that is composed of data. For example, a database is a data asset that comprises data records. In this document, “data asset” means system or application output files, databases, documents, or web pages. “Data asset” also includes services that may be provided to access data from an application. For example, a service that returns individual records from a database would be a data asset. Similarly, a website that returns data in response to specific queries (e.g., weather.com) would be a data asset. (DoD NCDS, 9 May 2003)
Data Element	A basic unit of data having a meaning and distinct units and values. (Derived from 8320.1) A uniquely named and defined component of a data definition; a data “cell” into which data items (actual values) can be placed; the lowest level of physical representation of data. (Derived from IEEE 610.5)
Data-Entity*	The representation of a set of people, objects, places, events or ideas that share the same characteristic relationships. (DDDS 4362 (A))
Data Model	A representation of the data elements pertinent to an architecture, often including the relationships among the elements and their attributes or characteristics. (DoDAF)
Decision Process	a : A business process developed and employed by management to guide change in an organization toward specific goals b : in the DoD, the seminal transformation processes include: JCIDS, DAS, PPBE, Portfolio Management and net-centric transformation
Dependency	Types are: “Is equivalent to”, “Is part of”, “Supports”, or “Replaces”.
Discovery Metadata	Metadata elements identified for use in searching and locating specific data asset content.
DoDAF Views	a: The 26 architecture products specifically identified in DoDAF v1.0 b:

	the views fall into four categories: OVs (9), SVs (13), TVs (2) and AVs (2).
Executable	A simulation or other analytical assist which performs tasks defined in a transformation process according to encoded instructions
Federated Architecture	A framework for EA development, maintenance and use that links, locates, and aggregates disparate architectures and architecture information via information exchange standards to deliver a seamless outward appearance to users. A federated architecture approach recognizes the uniqueness and specific purpose of disparate architectures, and allows for their autonomy and local governance, while enabling the enterprise to benefit from their content.
Format	The arrangement, order, or layout of data/information. (Derived from IEEE 610.5)
Global Information Grid	The globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating and managing information on demand to warfighters, policy makers, and support personnel. The GIG includes all owned and leased communications and computing systems and services, software (including applications), data, security services, and other associated services necessary to achieve Information Superiority. It also includes National Security Systems as defined in section 5142 of the Clinger-Cohen Act of 1996 (reference (b)). The GIG supports all Department of Defense, National Security, and related Intelligence Community missions and functions (strategic, operational, tactical, and business), in war and in peace. The GIG provides capabilities from all operating locations (bases, posts, camps, stations, facilities, mobile platforms, and deployed sites). The GIG provides interfaces to coalition, allied, and non-DoD users and systems. (DoDD 8100.1, 19 September 2002)
Information Product	a : A document or report that is specifically defined or called out in a DoD policy or instruction; b : In the case of the JCIDS, key information products include JOC, JIC, JCD, ICD, CDD, CPD, PIA, DCR and other JCIDS documents as well as DoDAF products such as the OV1, OV3, and OV5.
Integrated Architecture	An architecture consisting of multiple views or perspectives (Operational View, Systems View, and Technical Standards View) that facilitates integration and promotes interoperability across family of systems and system of systems and compatibility among related architectures (DoDD 4630.5) An architecture description that has integrated Operational, Systems, and Technical Standards Views with common points of reference linking the Operational View and the Systems View and also linking the Systems View and the Technical Standards View. An 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 the same (i.e., same

	names, definitions, and values) as architecture data elements referenced in another view. (DoDAF)
Interoperable Architectures	a: ability of one architecture to use the parts of another architecture b: The key to “interoperable architectures” (variously called: federations, communities, systems of systems, portfolios, capabilities, and so on) is ensuring that the same metadata constructs are shared by member architectures.
Metadata	*Metadata* is descriptive information about the meaning of other data. Metadata can be provided in many forms, including XML. (DoD NCDS, 9 May 2003)
Metadata Catalog	*Metadata Catalog* is a system that contains the instanc es of metadata associated with individual data assets. Typically, a metadata catalog is a software application that uses a database to store and search records that describe such items as documents, images, and videos. Search portals and applications can use metadata catalogs to locate the data assets that are relevant to their queries. (DoD NCDS, 9 May 2003)
Net-Centric Environment	A framework for full human and technical connectivity and interoperability that allows all DOD users and mission partners to share the information they need, when they need it, in a form they can understand and act on with confidence; and protects information from those who should not have it. (NCE Joint Functional Concept, 7 April 2005)
Net-Centric Operating Environment	The exploitation of the human and technical networking of all elements of an appropriately trained joint force by fully integrating collective capabilities, awareness, knowledge, experience, and superior decisionmaking to achieve a high level of agility and effectiveness in dispersed, decentralized, dynamic and uncertain operational environments. (NCE Joint Functional Concept, 7 April 2005)
Net-Centric Operations	The exploitation of the human and technical networking of all elements of an appropriately trained joint force by fully integrating collective capabilities, awareness, knowledge, experience, and superior decision making to achieve a high level of agility and effectiveness in dispersed, decentralized, dynamic and uncertain operational environments. (NCE Joint Functional Concept, 7 April 2005)
Operational Activity Model	A representation of the actions performed in conducting the business of an enterprise. The model is usually hierarchically decomposed into its component actions, and usually portrays the flow of information (and sometimes physical objects) between the component actions. In the Framework, the activity model portrays operational actions, not hardware/software system functions. (DoDAF)
Policy Map	a: A graphical depiction of the lexicon, logic, activities, and deliverable information products expressed in a set of policies and procedures (such as JCIDS or PPBE) usually in the form of a poster or plot; b: A primary

	tool for disambiguating the lexicon and logic within and among specific sets of policies and procedures and for understanding changes as these policy sets evolve through time
Protocol	a: According to Webster, a protocol is “a set of conventions governing the treatment and especially the formatting of data.” DoDAF v2.0 has three protocol structures: Overlay Protocol, mini-protocol, DoDAF v2.0 Architecture Protocol. b : A protocol is created by modeling the Overlay policy and processes. This model is depicted in the Policy Map.
PSA	Principal Staff Assistants (Office of the Secretary of Defense (OSD) officials holding Presidential appointments, Assistants to the Secretary of Defense (SECDEF), and OSD Directors or equivalents who report directly to the Secretary or Deputy Secretary of Defense. (DoDI 5025.1))
Reference Data	Common sets of terms, taxonomies and taxonomy element definitions that are standardized within a COI. Reference data is used within COIs to provide unambiguous reference definitions for data element instances used within architecture descriptions.
Reference Data Set	Instances of entities or objects that are designated as an authoritative reference and preferred or mandated for use. Reference Data Sets are defined and configuration managed by an Authoritative Source. Examples include the list of UJTLS, Common System Functions List (CSFL), Common Operational Activities List (COAL), Common Information Elements List (CIEL), etc.
Shared Space	*Shared space* is a mechanism that provides storage of and access to data for users within a bounded network space. Enterprise-shared space refers to a store of data that is accessible by all users within or across security domains on the GIG. A shared space provides virtual or physical access to any number of data assets (e.g., catalogs, web sites, registries, document storage, and databases). As described in this Strategy, any user, system, or application that posts data uses shared space. (DoD NCDS, 9 May 2003)
Structural Dependency	An association between data elements.
Taxonomy	a : A predefined classification scheme b: set of allowable values for variables described in an analytic task
Web Services	*Web services* are self-describing, self-contained, modular units of software application logic that provide defined business functionality. Web services are consumable software services that typically include some combination of business logic and data. Web services can be aggregated to establish a larger workflow or business transaction. Inherently, the architectural components of web services support messaging, service descriptions, registries, and loosely coupled interoperability. (DoD NCDS, 9 May 2003)

* Definitions shared between the Framework and CADM documents.

Functional Area*	A major area of related activity (e.g., Ballistic Missile Defense, Logistics, or C2 support). (DDDS 4198(A))
Information	The refinement of data through known conventions and context for purposes of imparting knowledge.
Information Exchange	Information that is passed from one operational node to another. Associated with an information exchange are such performance attributes as size, throughput, timeliness, quality, and quantity values.
Information Exchange Requirement*	A requirement for information that is exchanged between nodes. Performance attributes such as size, throughput, timeliness, quality, and quantity values are associated with an IER.
Link	A representation of the physical realization of connectivity between system nodes.
Metadata	data that defines and describes other data (ISO/IEC 11179)
Mission Area*	The general class to which an operational mission belongs. (DDDS 2305(A)) Note: Within a class, the missions have common objectives.
Mission*	An objective together with the purpose of the intended action. (Extension of DDDS 1(A)) Note: Multiple tasks accomplish a mission. (SPAWAR)
Needline*	A requirement that is the logical expression of the need to transfer information among nodes.
Network*	The joining of two or more nodes for a specific purpose.
Node*	A representation of an element of architecture that produces, consumes, or processes data.
Operational Node	A node that performs a role or mission.
Organization*	An administrative structure with a mission. (DDDS 345 (A))
Platform*	A physical structure that hosts systems or systems components.
Process	A group of logically related activities required to execute a specific task or group of tasks. (Army Systems Architecture Framework) Note: Multiple activities make up a process. (SPAWAR)
Requirement*	A need or demand. (DDDS 12451/1 (D))

Role	A function or position. (Webster's)
Rule	Statement that defines or constrains some aspect of the enterprise.
Service	A distinct part of the functionality that is provided by a system on one side of an interface to a system on the other side of an interface. (Derived from IEEE 1003.0)
System	A collection of components organized to accomplish a specific function or set of functions. (IEEE 610.12)
System Function*	A data transform that supports the automation of activities or exchange requirements.
Systems Node	A node with the identification and allocation of resources (e.g., people, platforms, facilities, or systems) required to implement specific roles and missions.
Task	An action or activity (derived from an analysis of the mission and concept of operations) assigned to an individual or organization to provide a capability. (UJTL, CJCSM 3500.04D, 2005)

ANNEX C
Dictionary of UML Terms

The terms included here are UML terms that are used in Volume III, Appendix F of this document. They convey some restrictive or special sense in this section. The sources for these definitions are [Booch et al., 1999] and [Rumbaugh, et al., 1999].

Abstraction	1. The act of identifying the essential characteristics of a thing that distinguish it from all other kinds of things. Abstraction involves looking for similarities across sets of things by focusing on their essential common characteristics. An abstraction always involves the perspective and purpose of the viewer; different purposes result in different abstractions for the same things. All modeling involves abstraction, often at many levels for various purposes. 2. A kind of dependency that relates two elements that represent the same concept at different abstraction levels.
Adornments	Textual or graphical items that are added to an element's basic notation and are used to visualize details from the element's specification.
Artifact	A piece of information that is used or produced by a software development process, such as an external document, or a work product. An artifact can be a model, description, or software.
Association	The semantic relationship between two or more classifiers that involves connections among their instances.
Attribute	An attribute is a named property of a class that describes a range of values that instances of the property may hold.
Building Blocks	There are three kinds of building blocks in UML: Things, Relationships, and diagrams.
Class	A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.
Component	A physical, replaceable part of a system that packages implementation and conforms to and provides the realization of a set of interfaces.
Constraint	A semantic condition or restriction represented as an expression. Certain constraints are predefined in the UML, others may be defined by modelers.
Constraint	An extension of the semantics of a UML element, allowing you to add new rules or modify existing ones.
Dependency	A relationship between two elements in which a change to one element (the supplier) may affect or supply information needed by the other element (the client).
Deployment Diagram	A network of node symbols connected by paths showing communication associations. UML Deployment Diagrams consist of physical nodes and dependency and association relationships among the nodes.

Derivation	A relationship between an element and another element that can be computed from it. Derivation is modeled as a stereotype of an abstraction dependency with the keyword Derive.
Derived Element	A [sic] element that can be computed from other elements and is included for clarity or for design purposes even though it adds no semantic information.
Diagram	A graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs (relationships) and vertices (other model elements). A diagram is contained within a package.
Element	An atomic constituent of a model.
Generalization	A taxonomic relationship between a more general element and a more specific element.
Instance	An individual entity with its own identity and value.
Model	A semantically complete abstraction of a system.
Node	A node is a run-time physical object that represents a computational resource, which generally has at least a memory and often processing capability. Run-time objects and run-time component instances may reside on nodes.
Notes	Notes may contain any combination of text or graphics. A note that renders a comment has no semantic impact, it does not alter the meaning of the model to which it is attached. Notes are used to specify things like requirements, observations, reviews, and explanations, in addition to rendering constraints.
OCL	Object Constraint Language, a text language for specifying constraints and queries.
Operations	An operation is the implementation of a service that can be requested from any object of the class to affect behavior.
Package	A package is a general-purpose mechanism for organizing elements into groups. Graphically, a package is rendered as a tabbed folder.
Realization	The relationship between a specification and its implementation; an indication of the inheritance of behavior without the inheritance of structure.
Refinement	A relationship that represents a fuller specification of something that has already been specified at a certain level of detail or at a different semantic level.
Relationships	There are four kinds of relationships in the UML: Dependency, Association, Generalization, Realization.
Stereotype	An extension of the vocabulary of the UML, which allows you to create new kinds

	of building blocks that are derived from existing ones but are specific to your problem. A stereotype is not the same as a parent class in a parent/child generalization relationship (e.g., parent class polygon, and child class rectangle). Rather, a stereotype is like a meta-type, because each one creates the equivalent of a new class in the UML's meta-model.
Tagged values	Every thing in the UML has its own set of properties: classes have names, attributes, and operations, and so on. With stereotypes you can add new things to the UML; with tagged values, you can add new properties.
Things	The abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things. There are four kinds of things in the UML: Structural things, behavioral things, grouping things, and annotational things.
Trace	A dependency that indicates a historical development process or other extra-model relationship between two elements that represent the same concept without specific rules for deriving one from the other. This is the least specific kind of dependency, and it has minimal semantics. It is mostly of use as a reminder for human thought during development.

ANNEX D
REFERENCES

Reference	Citation
[All-CADM, 2003a]	*All-DoD Core Architecture Data Model (All-CADM) for DoD Architecture Framework v1.0*, Volume 1, Overview Description, Office of the DoD Chief Information Officer, Draft (In Preparation), February 2003, UNCLASSIFIED.
[All-CADM, 2003b]	*All-DoD Core Architecture Data Model (All-CADM) for DoD Architecture Framework v1.0*, Volume 2, Technical Specification, Office of the DoD Chief Information Officer, Draft (In Preparation), February 2003, UNCLASSIFIED.
[All-CADM, 2003c]	*All-DoD Core Architecture Data Model (All-CADM) for DoD Architecture Framework v1.0*, Volume 3-Annexes, Office of the DoD Chief Information Officer, Draft (In Preparation), February 2003, UNCLASSIFIED.
[ASD (C3I) 1998]	*C4ISR Core Architecture Data Model (CADM) v1.5 (CADM v2.0),* Assistant Secretary of Defense for Command, Control, and Communications, Architecture and Interoperability Directorate, 1 December 1998, UNCLASSIFIED.
[ASD, 1999]	Assistant Secretary of Defense (Command, Control, Computers, and Intelligence), Memorandum, Subject: *Global Information Grid*, 22 September 1999.
[ASN(RDA)CHENG, 2002]	ASN(RDA)CHENG MCP *Strike Architecture Team, PROS Architecture Assessment Report*, October 2002.
[Bienvenu et al., 2000]	Bienvenu, M., I. Shin, and A. Levis, *C4ISR Architectures III: An Object-Oriented Approach for Architecture Design,* Systems Engineering, Vol. 3, No. 4, Fall 2000.
[Booch et al., 1999]	Booch, G., J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley Publishing Company, Reading MA, April 1999.
[C4ISR AWG, 1998]	C4ISR Architecture Working Group, *Levels of Information System Interoperability (LISI)*, 30 March 1998.
[CJCS, 1994]	Chairman, Joint Chiefs of Staff, *DoD Dictionary of Military and Associated Terms*, Joint Publication 1-02, 23 March 1994.
[CJCSM 3500.04D]	*Universal Joint Task List* (UJTL), published by Chairman, Joint Chiefs of Staff, 1 August 2005.
[DEB, 2000]	Defense Electronic Business, *Joint Electronic Commerce Architecture*, 2000.

Reference	Citation
[DDMS, 2003]	DoD Deputy Chief Information Officer, 29 July 2005, "DoD Discovery Metadata Specification (DDMS) v1.3"
[DeMarco 1979]	DeMarco, Tom, *Structured Analysis and Systems Specification*, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.
[Department of the Air Force, 2000]	Department of The Air Force, Air Force Instruction 33-124, *Enterprise Information Technology Architectures*, 1 May 2000.
[Department of the Navy, undated]	Department of the Navy Chief Information Officer, *Architecture Development Process Model*, Online, Available: www.doncio.navy.mil, undated.
[Department of the Treasury, 2000]	Department of the Treasury Chief Information Officers.
[DISA, 2000]	Defense Information Systems Agency, *Joint Technical Architecture*, Version 3.1, 31 March 2000.
[DISC4, 1998]	Director for Information Systems, Command, Control, Communications and Computers (DSC4), *Army Enterprise Architecture Guidance Document*, Version 1.1, 23 December 1998.
[DoD JP-1-02]	Department of Defense Dictionary of Military and Associated Terms, Joint Publication, August 2002.
[DoD, 2001]	Interim Regulation 5000.2-R, 4 January 2001.
[FIPS 183 1993]	*Integration Definition for Function Modeling (IDEF0)*, Federal Information Processing Standards (FIPS) Publication 183, 21 December 1993.
[FIPS 184 1993]	*Integration Definition for Data Modeling (IDEF1X)*, Federal Information Processing Standards (FIPS) Publication 184 21 December 1993.
[HARLEY 1987a]	Harel, D., *Statecharts: A visual Formalism for complex systems*, The Science of Computer Programming, 1987, 8, pp. 231-274.
[HARLEY 1987b]	Harel, D., A. Pnueli, J.P. Schmidt, and R. Sherman, *On The Formal Semantics of Statecharts*, Proceedings, Second IEEE Symposium, Logic Comput Sci, Dorset House, New York, 1987, pp. 54-64.
[IEEE Std 1471-2000]	*IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, The Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, New York, NY, 2000.
[IEEE, 1990]	Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard*

Reference	Citation
	Glossary of Software Engineering Terminology, IEEE STD 610.12-1990, Piscataway, NJ, 1990.
[ITMRA, 1996]	Information Technology Management Report Act (Clinger-Cohen Act of 1996).
[NATO, 2000]	NATO C3 Board, *Architecture Framework for NATO C3 Systems (NC3S)*, 2 October 2000.
[NCDS, 2003]	"*DoD Net-Centric Data Strategy*", Memorandum, 9 May 2003, **http://www.defenselink.mil/nii/org/cio/doc/Net-Centric-Data-Strategy-2003-05-092.pdf**.
[NIST, 1993(1)]	National Institute of Standards and Technology, *Integration Definition for Function Modeling (IDEF0)*, FIPS PUB 183, Gaithersburg, MD, 21 December 1993.
[NIST, 1993(2)]	National Institute of Standards and Technology, *Integration Definition for Information Modeling (IDEF1X)*, FIPS PUB 184, Gaithersburg, MD, 21 December 1993.
[NRO, 2001]	National Reconnaissance Office, *National Reconnaissance Office Architecture Framework (DRAFT)*, Version 0.9, May 2001.
[OMB, 2000]	Office of Management and Budget, Circular A-130: *Management of Federal Information Resources*, 30 November 2000.
[OMG, 1998]	Object Management Group (OMG), UML 1998, *Unified Modeling Language Specification*, Framingham, Mass., Internet: http://www.omg.org, 1998.
[OMG, 2000]	Object Management Group, UML Primer 2000, "What Is OMG-UML and Why Is It Important?," Framingham, Mass., Internet: http://www.omg.org/news/pr97/umlprimer.html, 2000.
[USD(A&T), ASD(C3I), J6, 1998]	Under Secretary of Defense (Acquisition and Technology), Assistant Secretary of Defense (Command, Control, and Communications), and Joint Staff/J6 Memorandum, Subject: *Strategic Direction for a DoD Architecture Framework*, 23 February 1998.
[Yourdon 1989]	Yourdon, Edward., *Modern Structured Analysis*, ISBN 0-13-598624-9, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
[Zachman, 1987]	Zachman, J.A., *A Framework for Information Systems Architecture*, IBM Systems Journal, 26(3): 276-291, 1987.