Date: August 2007

UML Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF), Beta 1

OMG Adopted Specification

OMG Document Number: dtc/2007-08-02

This OMG document replaces the submission document (dtc/2007-06-11, Alpha). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to *issues@omg.org* by December 31, 2007.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/*.

The FTF Recommendation and Report for this specification will be published on March 21, 2008. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

LICENSES

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™, and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).

Table of Contents

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A catalog of all OMG Specifications is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM)

Platform Specific Model and Interface Specifications

- CORBAservices

- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. (as of January 16, 2006) at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

Note – Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

Part I - Overview of the UML Profile for DoDAF and MODAF

This part contains the following Clauses:

1. Scope
2. Conformance
3. References
4. Terms and Definitions
5. Symbols and Abbreviations
6. Additional Information

1 Scope

The purpose of this document is to specify a UML 2 profile to enable practitioners to express DoDAF and MODAF model elements and organize them in a set of specified viewpoints and views that support the specific needs of stakeholders in the US Department of Defense and the United Kingdom Ministry of Defense.

UPDM defines a set of UML and SysML stereotypes and model elements and associations to satisfy the requirements of the UPDM RFP. This specification documents the language architecture in terms of the parts of UML 2 that are reused and the extensions to UML 2 and SysML. The specification includes the concrete syntax (notation) for the complete language. The reusable portion of the UML 2 and SysML specification are not included directly in the specification but are included by reference. The specification also provides an example of how the language can be used to represent a DoD architecture or MODAF architecture.

The scope of UPDM includes the language extensions to enable the extraction of specified and custom views from an integrated architecture description. These views include a system's viewpoint (DoDAF Systems View) along with associated systems implementation standards (DoDAF Technical View) within the context of the business or enterprise viewpoint (DoDAF Operational View). The DoDAF AllView is also included. In addition, UPDM allows the architecture model to include representation of an enterprise capability and strategic intent (MODAF Strategic Viewpoint) and the process steps associated with the procurement of conformant systems (MODAF Acquisition Viewpoint). UPDM also includes mechanisms for designing ad hoc custom views and more formal extensions of new views of the model. More details on the specifics of each of these views and viewpoints are provided in the references cited in Annex C.

UPDM will support the capability to

- model architectures for a broad range of complex systems, which may include hardware, software, data, personnel, and facility elements;
- model consistent architectures for system-of-systems down to lower levels of design and implementation;
- model service oriented architectures;
- support the analysis, specification, design, and verification of complex systems; and
- improve the ability to exchange architecture information among related tools that are UML based and tools that are based on other standards.

The profile provides the modeling of operational capabilities, services, system activities, nodes, system functions, ports, protocols, interfaces, performance, and physical properties and units of measure. In addition, the profile enables the modeling of related architecture concepts such as DoD’s doctrine, organization, training, materiel, leadership & education, personnel, and facilities (DOTMLPF) and the equivalent UK Ministry of Defence Lines of Development elements.

UPDM, as illustrated in the following diagram, will address DoDAF and MODAF Viewpoints as well as enabling extensions to new architecture perspectives (e.g., Services views, Custom views, Logistics views, etc.).

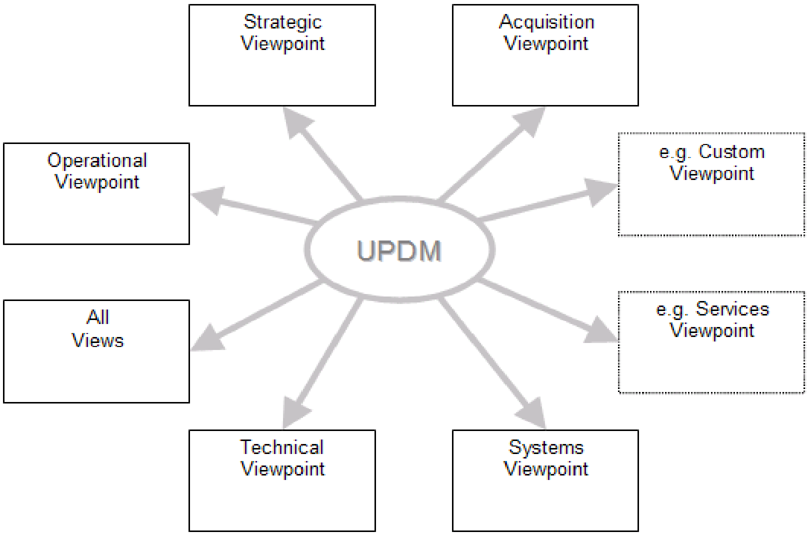


Figure 1.1 - UPDM Viewpoint Support Illustration

2 Compliance

2.1 Compliance Levels

UPDM specifies two compliance levels corresponding to supporting a UML-based profile and a UML+ OMG SysML™ profile.

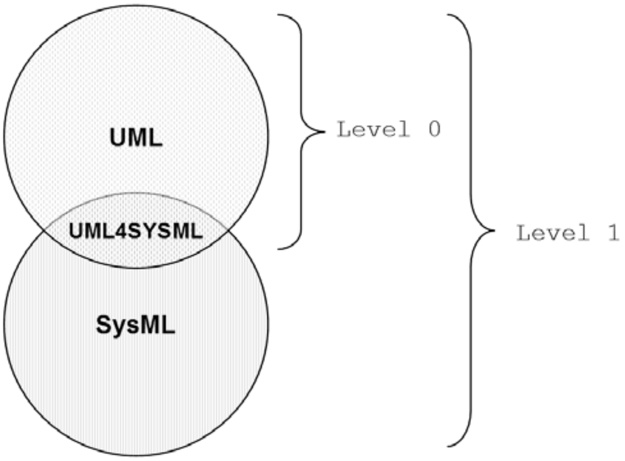


Figure 1.2 - UPDM Compliance Levels 0 and 1

2.1.1 Level 0: UPDM based on UML 2.1

UPDM Compliance Level 0 (UML profile) is the minimum but sufficient level of compliance that a tool vendor will need to meet to support this specification. The extensions for Level 0 only extend model elements that are included in the subset of UML that is referred to as UML4SysML. However, any UML 2 model element can be used when modeling with UPDM.

Compliance Level 0 is based on UML 2.1. Any part of UML may be used:

- Full metamodel compliance ; stereotypes, classes, attributes, constraints, associations, and package structure as set out in this specification.
- XMI serialization in both export and import.
- Stereotype definitions extend only those metaclasses found in SysML's UML2 subset called UML4SysML.
- A Level 0 compliant implementation must be able to import and export Level 0 UPDM models with 100% fidelity (i.e., no loss or transforms).

2.1.2 Level 1: UPDM based on UML 2.1 and SysML 1

UPDM Compliance Level 1 will reference the UML4SysML subset and import the SysML profile. It further extends Compliance Level 0 with additional SysML extensions.

Compliance Level 1 is based on UPDM Level 0 and SysML 1, (unrestricted mode). Any part of UML and SysML may be used.

- Level 1 UPDM includes Level 0 stereotype definitions. A subset of these definitions also specialize SysML stereotypes (see Clause 10).
- A Level 1 compliant implementation must be able to import and export Level 0 UPDM models with 100% fidelity (i.e., no loss or transforms).
- A Level 1 compliant implementation must be able to import and export Level 1 UPDM models with 100% fidelity (i.e., no loss or transforms). The UML models from Level 0 will be preserved. The SysML models from Level 1 will be preserved.

Note: Using SysML in restricted mode is not defined and may produce non-standard UPDM. See the supplementary documents for representations of the XMI for Level 0 and Level 1 models.

3 Normative References

The following normative documents contain provisions, which through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

UML 2.1.1 Superstructure Specification

UML 2.1.1 Infrastructure Specification

MOF 2.0 Core

UML 2.0 OCL Specification

OMG SysML Specification

XMI 2.1 Specification

DoD Architecture Framework, Version 1.0, Volume I: Definitions and Guidelines, February, 9, 2004; DoD Architecture Framework Working Group (http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_I.pdf)

DoD Architecture Framework, Version 1.0, Volume II: Product Descriptions, February, 9, 2004; DoD Architecture Framework Working Group (http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_II.pdf)

DoD Architecture Framework, Version 1.0, Deskbook, February, 9, 2004; DoD Architecture Framework Working Group (http://www.defenselink.mil/nii/doc/DoDAF_v1_Deskbook.pdf).

MODAF Architectural Framework Technical Handbook version 1.0, 31 MODAF-M07-022, August 2005.

4 Terms and Definitions

No new terms and definitions have been required to create this specification. All terms should be available in the normative references or bibliographic citations for detailed explanation.

5 Symbols/Acronyms

• AcV-*	Acquisition View
• AV-*	All View
• BPMN	Business Process Modeling Notation
• C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
• DoD	United States Department of Defense
• DoDAF	Department of Defense Architecture Framework
• DOTMLPF	Doctrine, Organization, Training, Material, Leadership, Personnel, Facilities
• IDEF0	Integrated DEFinition Methods: IDEF0 is the Functional Modeling Aspect of the IDEF methods.
• JCIDS	Joint Capabilities Integration and Development System
• JETL	Joint Essential Task List
• MOD	United Kingdom Ministry of Defence
• MODAF	Ministry of Defence Architecture Framework
• NCAT	NetCentric Assessment Tool
• NCOIC	NetCentric Operations Industry Consortium
• OV-*	Operational View
• POC	Proof of Concept

• SoS	System of Systems
• StV-*	Strategic View
• SV-*	System View
• TV-*	Technical View
• UPDM	UML Profile for DoDAF and MODAF

6 Additional Information

Accompanying this specification are XMI files for the Compliance Levels 0 and 1. These are:

• UPDM-L0 model.xmi	UPDM Compliance Level 0 formal metamodel
• UPDM-L1 model.xmi	UPDM Compliance Level 1 formal metamodel
• UPDM-L0 profile.xmi	UPDM Compliance Level 0 Profile
• UPDM-L1 profile.xmi	UPDM Compliance Level 1 Profile
• UPDM-L0 ClassLibrary.xmi	UPDM Compliance Level 0 Class library

6.1 Overview of this Specification

Intended Audience

This specification will be of interest to end users who expect to use this profile, and to tool vendors interested in developing tool support for the development of enterprise and system of systems architectures, and that can satisfy contract documentation requirements for DoD and MOD customers. Tool vendors will also be able to use this specification to support Model Driven Development of systems based on the architectural descriptions based on this profile. Developers and reviewers of the views will have a clearer understanding of the semantics behind specific views and viewpoints, which will support more precise evaluation and comparison.

Organization of this Specification

DoDAF and MODAF are specifically organized around a set of viewpoints and views that address the concerns of a well-defined set of stakeholders. This specification organizes the presentation of the UPDM abstract and concrete syntax around those viewpoints, so that the discussion is well-connected to their domain expertise. (See Section 1.5 for a more detailed description.)

The rest of this document contains the technical content of this specification. As background for this specification, readers may wish to review the UML and OMG SysML™ specifications that complement this specification.

Although the clauses are organized in a logical manner and can be read sequentially, this is a reference specification that can be read in a non-sequential manner.

Part I describes the details of this specification.

Part II provides the technical details essential to understanding the specification:

The specification of the Profile language. The profile includes both a Compliance Level 0 that extends UML and a Compliance Level 1 that extends UML and OMG SysML™. The elements of the profile are organized by the specific viewpoints required by DoDAF and MODAF. Within each of the viewpoint-specific sections, e.g., Operational Views (OVs), the elements are presented in alphabetical order. A reader interested in a UML only solution can read the section on the Compliance Level 0, whereas a reader interested in an additional SysML Solution should read both the Compliance Level 0 and Level 1 sections.

Annex A presents non-normative MODAF and DoDAF Artifact Guidance with usage examples. It presents a description of the key semantic relations that apply between the views and viewpoints that can be used to produce standard view deliverables.

Annex B presents a non-normative domain metamodel that defines the semantics behind the profile.

Annex C presents an example set of DoDAF artifacts based on an earlier version of the Compliance level 0 profile

Annex D contains the bibliography.

6.2 Acknowledgements

The following individuals submitted parts of this specification and/or have assisted the UPDM team in the development of the specification:

• Adaptive Inc	Pete Rivett
• ARTiSAN Software Tools Ltd.	Francis Thom
• BAE Systems	Charles Dickerson
• BCMS	Paul Bahrs
• CSCI	Bob Jacobs
• EmbeddesPlus Engineering	Cory Bialowas
• EuroStep	David Price
• General Dynamics	Joz
• IBM	Peter Bahrs
• IBM	Kevin Cornell
• IBM	Steven Hovater
• IBM	Sridhar Iyengar
• IBM	Fred Mervine
• IBM	Eric Nelson
• IBM	Bran Selic
• Lockheed Martin	Sanford Friedenthal
• Lockheed Martin	Laura Hart
• Mitre	Fatma Dandashi
• No Magic	Andrius Strazdauskas
• Northrup Grumman	Joe Kuncel
• OSD	Dwayne Hardy

Pathfinder Solutions	Greg Eakman
Raytheon	Ron Williamson
SAIC	Jonathan C. Bollers
SAIC	David H. Fado
SAIC	Steven Gordon
SAIC	Brand Niemann
Sparx Systems	Sam Mancarella
Telelogic AB	Morgan Björkander
Telelogic AB	Lou Varveris
THALES Group	Veronique Normand

Part II - Language Architecture

This part contains the following Clause and subclauses:

7 Language Architecture

7.1 Design Principles for the Architecture

This specification used the following design principles that support the domain needs and design rationale described above.

- UML. The specification contains a profile that extends UML, not a new MOF-based language. The profile is conformant with the UML Infrastructure Specification that states “A profile must be based on a metamodel such as the UML that it extends, and is not very useful standalone.” (page 13).
- SysML The specification contains a profile that extends SysML. This allows the models to be created and reused across the architecture and engineering lifecycles and seamlessly integrate the UPDM models with SysML models and seamlessly integrates the UPDM models with SysML models.
- Model Driven Design. The UPDM represents one module in a large set of language constructs that an enterprise will use for successful modeling. Communication with other modules in this broad set of constructs will take place through standardized interfaces and transformations. A Model-View-Constraints (MVCo) approach to modeling is used.
- Extensibility. The UPDM must have ways to extend the language to fit particular clients and to take advantage of new developments in modeling tools. Where possible, this profile looks to use simple mechanisms such as packages that can reference URIs to allow for the creation of new views without requiring the release of a new version of the profile. The profile also looks to provide a minimum set of constraints on the use of existing modeling techniques.
- Model Architecture. The UPDM is organized into a domain model expressed independently of the UML extensions that implement that model. The UPDM profile shows the UML extensions that provide a quickly realizable implementation. The UPDM profile has an integrated class library to provide additional extensibility options. The architecture follows the design for profiles as set out in the UML Superstructure and Infrastructure documents. The UPDM model transform recommendation for importing and exporting model views and legacy models is used in the production of the UPDM specification. Requirements drove the design of the domain metamodel and profile. The profile metamodel is an auto generated transform of the profile. The document specification sections for the domain metamodel, profile model and profile are auto generated transforms.

7.2 Domain Considerations for the Architecture

A large number of architectures for the US government use the DoDAF. DoDAF defines a common approach for describing, presenting, and comparing DoD architectures. The framework seeks to facilitate the use of common principles, assumptions, terminology, and representation in architectural descriptions. In application, the DoDAF has evolved a number of different implementations, some using UML and some using other modeling constructs. Even those that use UML often use the UML 1.x implementation as suggested in the DoDAF documentation. UML 2 offers a number of opportunities to improve the delivery of DoDAF work products. DoDAF implementations have started to use UML 2 features and custom profiles with success on the immediate project deliverables. Unfortunately, this proliferation of implementations has made the comparison and assessment of architectures difficult, so the larger goal of DoDAF supporting comparisons has not been realized.

To make architectural comparison even more difficult, DoDAF architectures must also comply with the Federal Enterprise Architecture Reference Models and some elements of the DoD require compliance with JCIDS, the Joint Capabilities Integration and Development System. The MODAF uses many of the DoDAF views and extends these with additional views related to strategic capabilities and acquisition; learning from some of the struggles users have had with DoDAF.

Given the wide variety of DoDAF implementations, the language architecture for UPDM does not need to invent new language mechanisms to add to the confusion, but to look for ways to allow existing tooling to produce DoDAF and MODAF documentation in an efficient manner. Many of the existing implementations are sufficient to work well within their context and don't require new architectural descriptions. Indeed, some DoD products have sophisticated implementations of MDA and executable[1] UML that should not be disrupted by the requirement to express the architectural descriptions with DoDAF or MODAF views. In order to satisfy the production of these views, many elements can use standard mechanisms for marking up model entities to allow expression of those constructs in automatically generated work products.

The review of existing DoDAF architectural descriptions and discussion of proposed changes to DoDAF underscored the importance of understanding the DoD context and the legacy of the architectural framework for C4ISR. Focus on defining an individual work product can lead to fruitless discussions where the actual disagreement is not with the modeling constructs but with the intended content of a particular DoDAF deliverable. This profile needs a domain model that provides an assessment of the core elements called for in DoDAF and MODAF, independent of any specific UML implementation (see Annex B). There are a number of ways for UML, or other modeling languages, to represent the information called for in a domain, so UPDM proposes a standard semantic foundation for the DoDAF and MODAF domains.

As the needs of the domain change, the implementation constructs in UML and other supporting language will need to change. The DoDAF domain does change rapidly, with requirements for top-end systems with low fault tolerance driving exacting modeling standards while the overhead of government bureaucracies enforcing contractual requirements does little to remove the ambiguity and distortion in the domain requirements. For example, the current domain model in this profile does not include constructs for handling the modeling of supply chains or logistics since the domain served by DoDAF has focused primarily on the exchange of information and the construction of software-based communication systems. Given the increasing reliance on DoDAF as an enterprise architecture framework, the domain has to handle the transfer of items other than information and systems that include non-software components. These will require new constructs. Similarly, the requirements for capability-based analysis will require new constructs in DoDAF. Given the uncertainty of the international climate, this profile assumes continued rapid change in the needs of this domain for the future. The language architecture and the supporting profile definition needs to address the instability in the domain elements by providing extension mechanisms in a coherent manner and a clear update path to allow for the continued evolution of DoD / MOD strategies.

This profile deals with domain instability by analyzing and defining a baseline so that both the domain model and UML extensions can rely on change control.

7.3 Basic Architecture

The UPDM package structure closely follows the DoDAF/MODAF product structure.

[1] Executable refers to the ability to simulate the execution of models diagramatically or with real interactions between systems. This is common in developing what-if scenarios and testing.

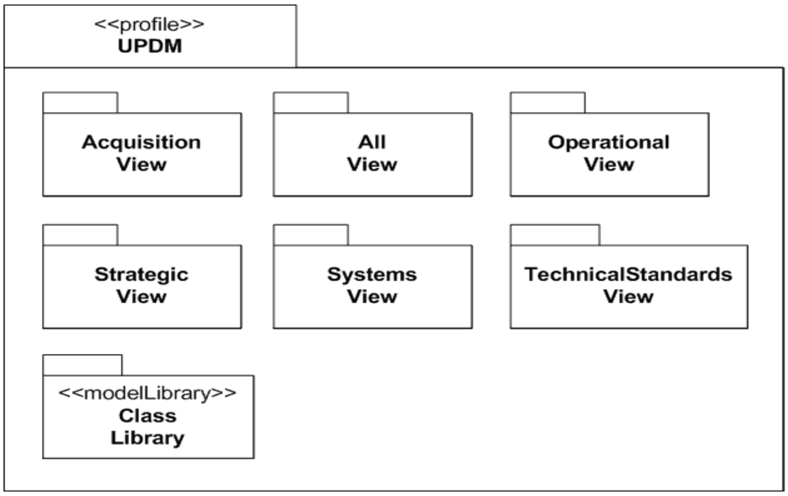


Figure 7.1 - UPDM Package Structure

7.3.1 Extension Mechanisms

The UPDM has been designed to be fully exploitable in a Model Driven Architecture paradigm. Transformations to and from UPDM are feasible. The UPDM design also includes extensibility for:

- UPDM Interoperability and Reuse
- Service Oriented Architecture
- Additional Viewpoints and Views
- Report and Image Artifacts
- Model Driven Architecture
- Executable UPDM

The following figure shows the UPDM architecture. The UPDM Profile is based on the UPDM Domain and Profile Metamodels that provide a semantically rich and precise definition of DoDAF and MODAF.

UPDM is based on UML and is stored in XMI. The representation in XMI (XML) allows easy manipulation by other tool and vendor implementations.

7.3.2 Extension Examples

The following figure shows the UPDM architecture. The UPDM Domain Metamodel expresses the DoDAF and MODAF concepts. The UPDM Profile defines an implementation of these concepts that is based on UML and SysML.

UPDM Models are exchanged as XMI. The representation in XMI (XML) allows easy manipulation by other tool and vendor implementations.

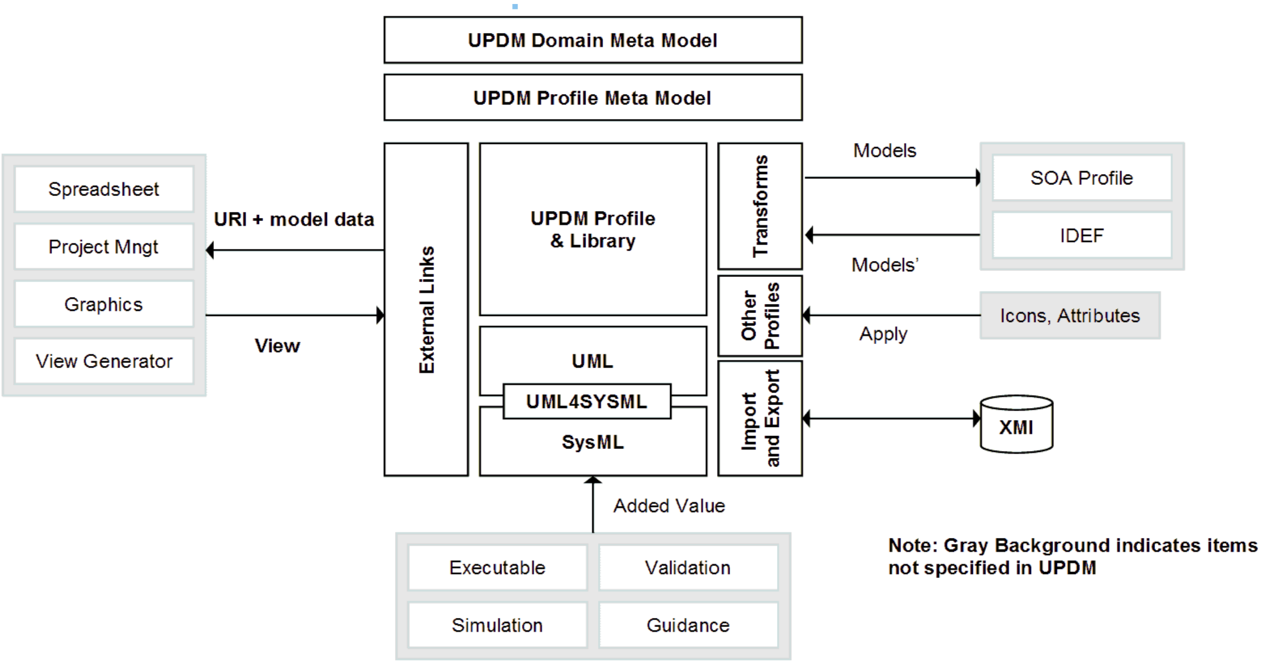


Figure 7.2 - UPDM Architecture

The UPDM Profile uses external links and model data to provide additional views that are not financially or technically feasible to model in UML. The links invoke, in an implementation specific manner, external productivity tools or systems to provide the views. Examples of views that would come from external tools are pie charts such as those used to monitor project progress (AcV-2) and conceptual representations (OV-1).

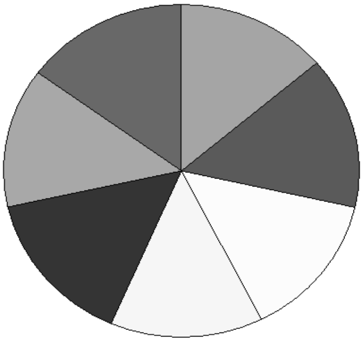


Figure 7.3 - AcV-2 View Example

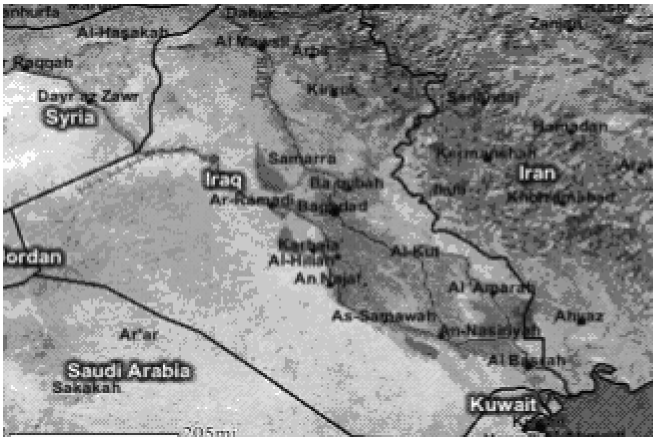


Figure 7.4- OV-1 View Example

External links can also be used in model elements to refer to externally published information on which the model depends. An example is an externally published taxonomy where it is not practical (or necessary) to copy the taxonomy into each model.

Transforms should be applied on UPDM models to enable importing and exporting to other repositories or tools. Transforms also allow UPDM models to be reused with other relevant standards including the OMG RFP for an SOA Profile, and SysML. Transforms can also be used to generate reports.

The UPDM Profile can also be extended to allow additional attributes to be applied. For example, icons of tanks, buildings or people can be applied differently across the Services without changing the semantics of the model.

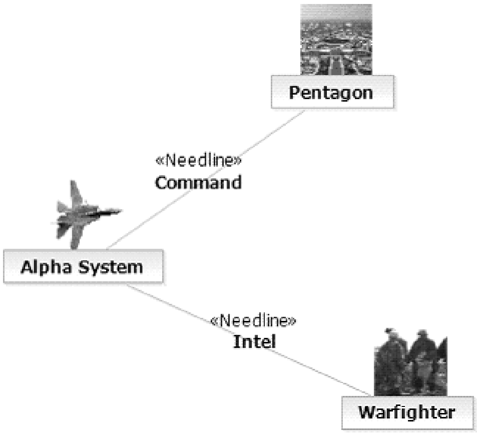


Figure 7.5 - Use of Multiple Concurrent Profiles

The final point in the architecture is that it is up to each implementation of the UPDM standard to provide vendor differentiating enablers such as model simulation, execution, and validation. Additionally, a vendor may provide tool or standard guidance on how to model UPDM models more effectively.

Part III - UPDM Profile Specification

This part contains the following Clauses:

8 UPDM Profile Specification Compliance Level 0

8.1 Overview

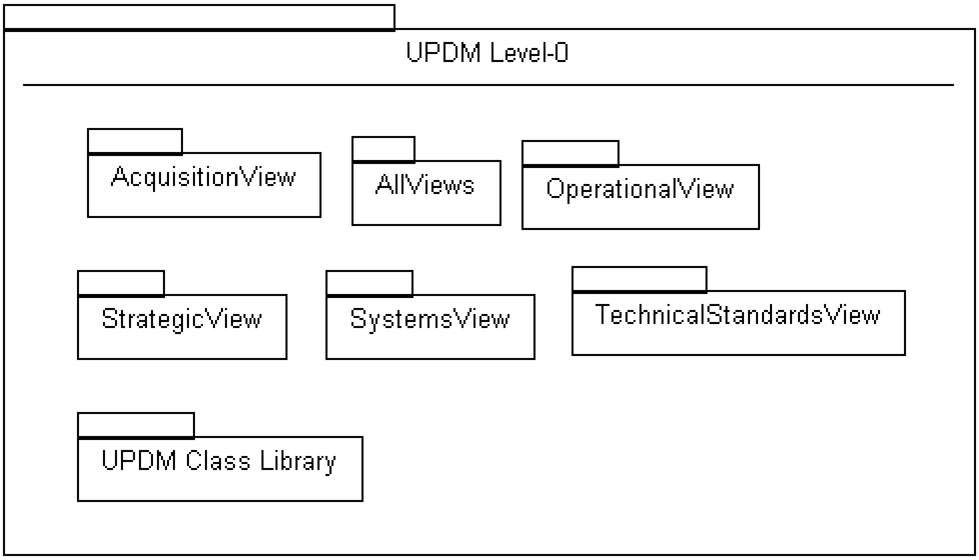


Figure 8.1- UPDM Packages

8.2 AcquisitionView Package

8.2.1 Overview

The Acquisition View Package contains UML stereotypes that assist the modeler in developing the views defined in the MODAF Acquisition Process. These views support the acquisition program dependencies, timelines, and DLOD status to inform program management. They have been introduced to describe programmatic details, including dependencies between projects and capability integration across the all the DLODs. These Views guide the acquisition and fielding processes.

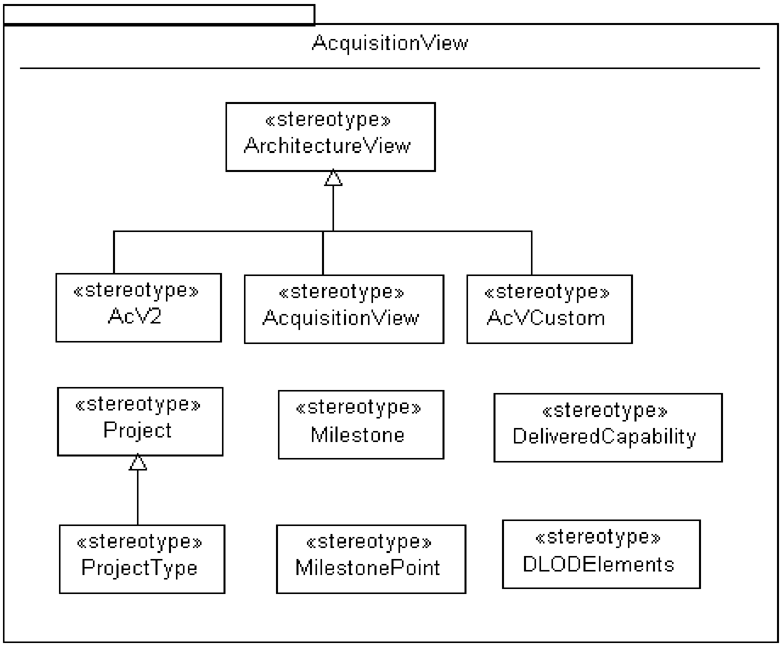


Figure 8.2 - Acquisition View Packages

8.2.2 AcquisitionView

8.2.2.1 Extension

8.2.2.2 Generalizations

- ArchitectureView (from AllViews)

8.2.2.3 Description

The AcquisitionView package provides three subpackages to contain the individual AcquisitionView deliverables.

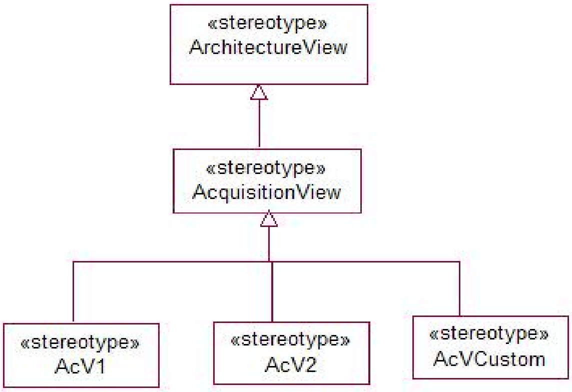


Figure 8.3- Acquisition View

8.2.2.4 Attributes

No additional attributes.

8.2.2.5 Associations

No additional associations.

8.2.2.6 Constraints

No additional constraints.

8.2.2.7 Semantics

The packages in the AcquisitionView will usually contain links to the output from external planning tools, such as a project planning tool. (See UPDMAttributes and ExternalReferences.)

8.2.2.8 Notation

No additional notational requirements.

8.2.3 AcV-1: System of Systems Acquisition Clusters

8.2.3.1 Extension

8.2.3.2 Generalizations

- AcquisitionView (from AcquisitionView)

8.2.3.3 Description

This package contains the AcV-1 System of Systems Acquisition Clusters (AcV-1) that describes how acquisition projects are organizationally grouped in order to form coherent acquisition programs.

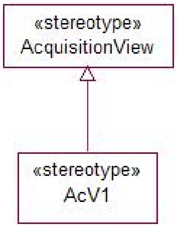


Figure 8.4 - AcV-1 System of Systems Acquisition Clusters View

8.2.3.4 Attributes

No additional attributes.

8.2.3.5 Associations

No additional associations.

8.2.3.6 Constraints

No additional constraints.

8.2.3.7 Semantics

No additional semantic requirements.

8.2.3.8 Notation

No additional notational requirements.

8.2.4 AcV-2: SoS Acquisition Programme

8.2.4.1 Extension

8.2.4.2 Generalizations

- AcquisitionView (from AcquisitionView)

8.2.4.3 Description

System of System Acquisition Programs - provides an overview of a program of individual projects, based on a time-line. It summarizes, for each of the projects illustrated, the level of maturity achieved across the DLODs at each stage of the CADMID lifecycle, and the interdependencies between the project stages.

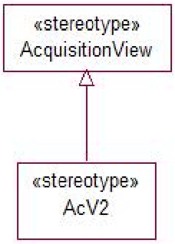


Figure 8.5 - AcV-2 SoS Acquisition Program View

8.2.4.4 Attributes

No additional attributes.

8.2.4.5 Associations

No additional associations.

8.2.4.6 Constraints

No additional constraints.

8.2.4.7 Semantics

No additional semantic requirements.

8.2.4.8 Notation

No additional notational requirements.

8.2.5 AcVCustom Custom Acquisition View

8.2.5.1 Extension

8.2.5.2 Generalizations

- AcquisitionView (from AcquisitionView)

8.2.5.3 Description

A user-defined view that applies to the AcquisitionView.

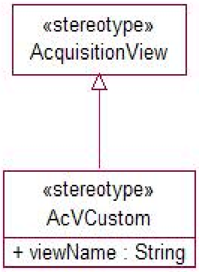


Figure 8.6 - AcVCustom View

8.2.5.4 Attributes

- viewName : String [1]
  Provides a name for the custom view so that it can be easily distinguished from other custom AcquisitionViews.

8.2.5.5 Associations

No additional associations.

8.2.5.6 Constraints

No additional constraints.

8.2.5.7 Semantics

No additional semantic requirements.

8.2.5.8 Notation

No additional notational requirements.

8.2.6 DeliveredCapability

8.2.6.1 Extension

- Association (from UML2)

8.2.6.2 Generalizations

8.2.6.3 Description

A DeliveredCapability indicates which Capabilities are to be delivered at the successful completion of a specific Project and which Project will deliver specific Capabilities.

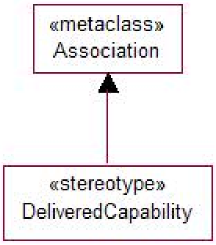


Figure 8.7- Delivered Capability

8.2.6.4 Attributes

No additional attributes.

8.2.6.5 Associations

No additional associations.

8.2.6.6 Constraints

[1] Asserts that the ends of the association are Capability and Project.

```
(self.endType->at(1).getAppliedStereotype('UPDM::Capability')->notEmpty() and
self.endType->at(2).getAppliedStereotype('UPDM::Project')->notEmpty()) or
(self.endType->at(2).getAppliedStereotype('UPDM::Capability')->notEmpty() and
self.endType->at(1).getAppliedStereotype('UPDM::Project')->notEmpty())
```

8.2.6.7 Semantics

No additional semantic requirements.

8.2.6.8 Notation

No additional notational requirements.

8.2.7 DLODElements

8.2.7.1 Extension

8.2.7.2 Generalizations

- Specification (from AllViews)

8.2.7.3 Description

Defence Logistics Operation Centre, has the attributes to show the status of a particular capability in terms of whether it is available. These are the attributes used in the pie charts to illustrate project maturity at lifelines:

- Training
- Equipment
- Logistics
- Infrastructure
- Organization
- Doctrine/concepts
- Information
- Personnel

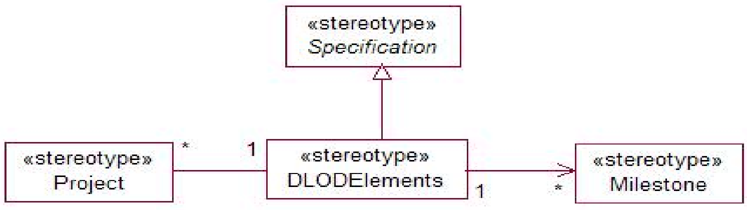


Figure 8.8 - DLODElements

8.2.7.4 Attributes

No additional attributes.

8.2.7.5 Associations

milestone : Milestone [*]	The Milestone to which these Elements apply.
project : Project [*]	The Project to which these elements apply.

8.2.7.6 Constraints

[1] Asserts that there is a Milestone that relates to a Project through this Element.

```
self.project-> forAll(getAppliedStereotype('UPDM::Project')->notEmpty())
```

[2] Asserts that there is a Milestone that relates to a Project through this Element.

```
self.milestone-> forAll(getAppliedStereotype('UPDM::Milestone')->notEmpty())
```

8.2.7.7 Semantics

No additional semantic requirements.

8.2.7.8 Notation

No additional notational requirements.

8.2.8 Milestone

8.2.8.1 Extension

- Class (from UML2)

8.2.8.2 Generalizations

8.2.8.3 Description

A Milestone is significant date for the evolution of a set of SystemGroups. For software, this could indicate the release of a new version of a suite of applications. It can also be a decision point that separates the phases of a directed, funded effort that is designed to provide a new or improved material capability in response to a validated need. It can also be an event in a Project by which progress is measured.

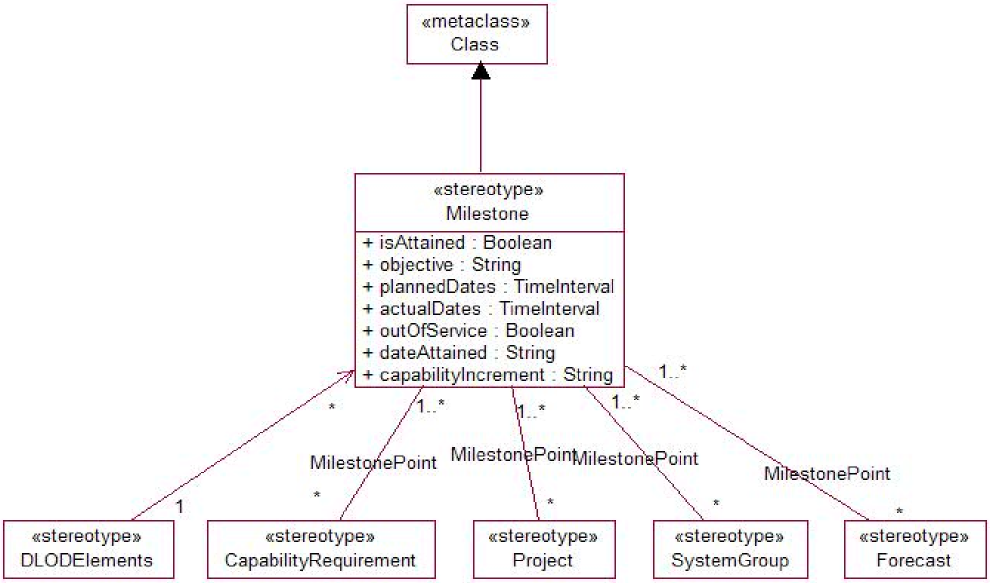


Figure 8.9- Milestone

8.2.8.4 Attributes

• actualDates : TimeInterval [1]	Actual start and end dates
• capabilityIncrement : String [1]	Level of Capability that has been deployed by the project
• dateAttained : String [1]	Date that the milestone was attained
• isAttained : Boolean [1]	Has the milestone been attained?
• objective : String [1]	A description of the objective of this milestone
• outOfService : Boolean [1]	Is the milestone out of service?
• plannedDates : TimeInterval [1]	planned start and end dates

8.2.8.5 Associations

- capabilityRequirement : CapabilityRequirement [*]See MilestonePoint for more details.
- forecast : Forecast [*]
- project : Project [*]
- systemgroup : SystemGroup [*]

8.2.8.6 Constraints

[1] Asserts that there is an association between Milestone and at least one of CapabilityRequirement, Project, Forecast or SystemGroup

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::MilestonePoint')-> notEmpty())->notEmpty()
```

[2] Asserts that the type of plannedDates is TimeInterval

```
self.plannedDates->forAll(classifier->forAll(name='TimeInterval')) or

self.plannedDates->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='TimeInterval')))
```

[3] Asserts that the type of actualDates is TimeInterval

```
self.actualDates->forAll(classifier->forAll(name='TimeInterval')) or

self.actualDates->forAll(classifier->forAll(allParents()->notEmpty() and
 allParents()->exists(name='TimeInterval')))
```

8.2.8.7 Semantics

A Milestone is a set of key dates for the systems analyzed in the ArchitecturalDescription. Milestone can apply to Capabilities and SystemGroups. Projects and Forecasts can be associated with a Milestone.

8.2.8.8 Notation

No additional notational requirements.

8.2.9 MilestonePoint

8.2.9.1 Extension

- Association (from UML2)

8.2.9.2 Generalizations

8.2.9.3 Description

Relates a Milestone to the elements that are governed by the Milestone.

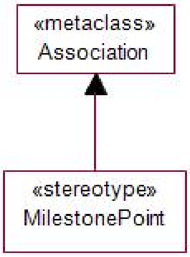


Figure 8.10 - MilestonePoint

8.2.9.4 Attributes

No additional attributes.

8.2.9.5 Associations

No additional associations.

8.2.9.6 Constraints

[1] Asserts that one end of the association is Milestone and the other end is one of:CapabilityRequirement, Forecast, Project, or SystemGroup.

```
(self.endType->at(1).getAppliedStereotype('UPDM::Milestone')->notEmpty()  and

self.endType->at(2).getAppliedStereotype('UPDM::CapabilityRequirement')-
>notEmpty()or

self.endType->at(2).getAppliedStereotype('UPDM::Forecast')->notEmpty()or

self.endType->at(2).getAppliedStereotype('UPDM::Project')->notEmpty()or

self.endType->at(2).getAppliedStereotype('UPDM::SystemGroup')->notEmpty())

   xor

(self.endType->at(2).getAppliedStereotype('UPDM::Milestone')->notEmpty()  and

self.endType->at(1).getAppliedStereotype('UPDM::CapabilityRequirement')-
>notEmpty()or

self.endType->at(1).getAppliedStereotype('UPDM::Forecast')->notEmpty()or

self.endType->at(1).getAppliedStereotype('UPDM::Project')->notEmpty()or

self.endType->at(1).getAppliedStereotype('UPDM::SystemGroup')->notEmpty())
```

8.2.9.7 Semantics

No additional semantic requirements.

8.2.9.8 Notation

No additional notational requirements.

8.2.10 Project

8.2.10.1 Extension

- Class (from UML2)

8.2.10.2 Generalizations

8.2.10.3 Description

A Project is a plan by an organizational unit to procure systems related to operations scheduled for a finite period of time. Implementation milestones can be assessed using the system evolution milestones associated with the system evolution description.

Figure 8.11 - Project

8.2.10.4 Attributes

No additional attributes.

8.2.10.5 Associations

- capability : Capability [1..*]
  See DeliveredCapability for more details.

- dlodelements : DLODElements [1]

- milestone : Milestone [1..*]

- organizationalResource : OrganizationalResource [*]
  The OrganizationalResource that owns this Project

8.2.10.6 Constraints

[1] Asserts that there is one OrganizationalResource required for this Project

```
self.organizationalResource.getAppliedStereotype
('UPDM::OrganizationalResource')->notEmpty()
```

[2] Asserts that zero or more Milestones govern this Project through an DOLDElement

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

    (getAppliedStereotype('UPDM::MilestonePoint')-> notEmpty())->notEmpty()
```

[3] Asserts that Capabilities are to be delivered by this Project

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

    (getAppliedStereotype('UPDM::DeliveredCapability')-> notEmpty())->notEmpty()
```

[4] Asserts that there is a set of DLODElements that describe this Project.

```
self.dlodElements.getAppliedStereotype('UPDM::DLODElements')->notEmpty()
```

8.2.10.7 Semantics

No additional semantic requirements.

8.2.10.8 Notation

No additional notational requirements.

8.2.11 ProjectType

8.2.11.1 Extension

8.2.11.2 Generalizations

- Project (from AcquisitionView)

8.2.11.3 Description

A ProjectType is a category of a Project, such as an acquisition project or program, and has much the same capabilities as a Project.

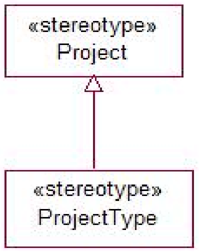


Figure 8.12 - ProjectType

8.2.11.4 Attributes

No additional attributes.

8.2.11.5 Associations

No additional associations.

8.2.11.6 Constraints

No additional constraints.

8.2.11.7 Semantics

No additional semantic requirements.

8.2.11.8 Notation

No additional notational requirements.

8.3 AllViews Package

8.3.1 Overview

The AllViews Package contains UML stereotypes that assist the modeler in developing the views defined in the AllViews viewpoint. These stereotypes support the description of some overarching aspects of an architecture that relate to all three views. The AllViews Package includes the subject area and time frame for the architecture. The setting in which the architecture exists comprises the interrelated conditions that compose the context for the architecture. These conditions include doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations (CONOPS); scenarios; and environmental conditions.

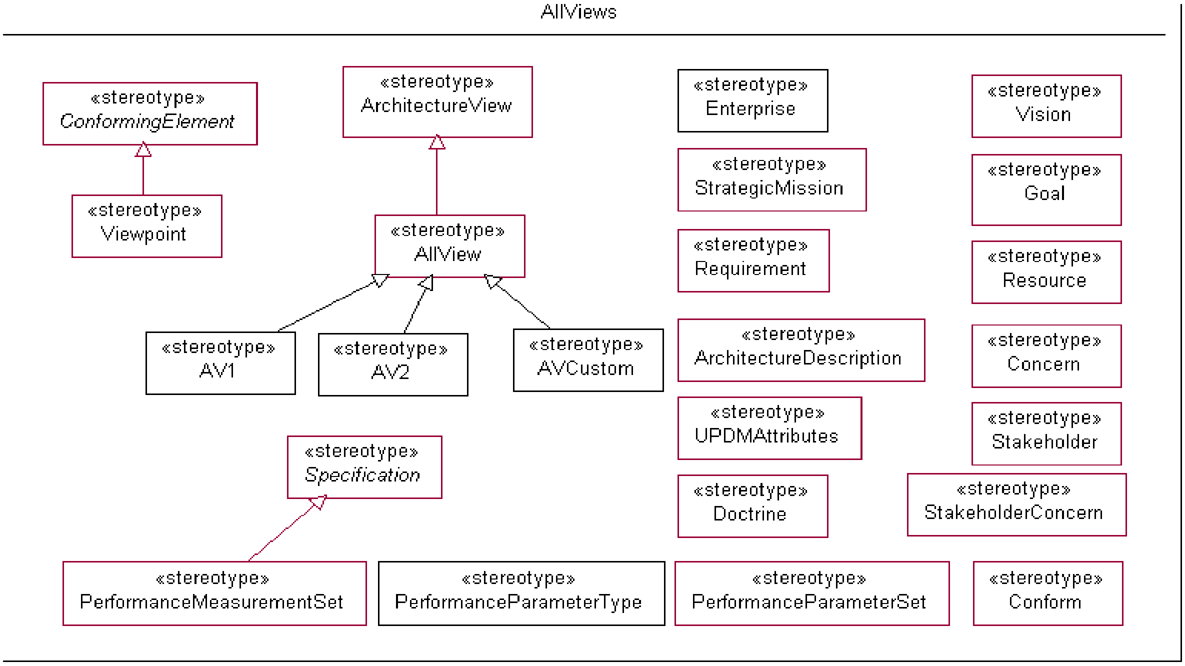

AllViews
«stereotype» ConformingElement
«stereotype» ArchitectureView
«stereotype» Enterprise
«stereotype» Vision
«stereotype» StrategicMission
«stereotype» Goal
«stereotype» Viewpoint
«stereotype» AllView
«stereotype» Requirement
«stereotype» Resource
«stereotype» AV1
«stereotype» AV2
«stereotype» AVCustom
«stereotype» ArchitectureDescription
«stereotype» Concern
«stereotype» UPDMAttributes
«stereotype» Stakeholder
«stereotype» Specification
«stereotype» Doctrine
«stereotype» StakeholderConcern
«stereotype» PerformanceMeasurementSet
«stereotype» PerformanceParameterType
«stereotype» PerformanceParameterSet
«stereotype» Conform

Figure 8.13 - AllViews Package Structure

8.3.2 AllViews

8.3.2.1 Extension

8.3.2.2 Generalizations

- ArchitectureView (from AllViews)

8.3.2.3 Description

AllViews is a package that contains information for one of the AllView deliverables. An AllView deliverable, such as the AV-2, might well consist of a number of packages. These packages do not have to be within the same parent package.

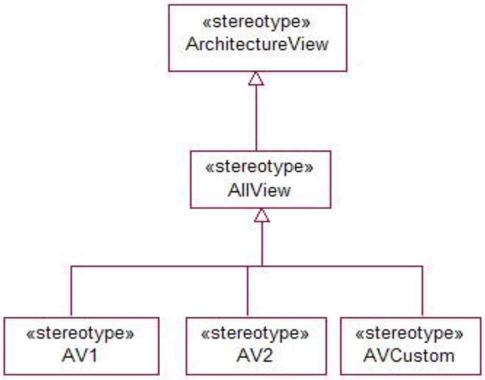


Figure 8.14 - AllView

8.3.2.4 Attributes

No additional attributes.

8.3.2.5 Associations

No additional associations.

8.3.2.6 Constraints

No additional constraints.

8.3.2.7 Semantics

The stereotyped package indicates to the user that the contents inside that package satisfy the indicated framework product or user defined product. The package is viewed as an organizing mechanism for views into the architecture, not as a package to hold DoDAF elements. Individual elements will typically reside outside of these packages with the diagrams that show the views for these elements inside the package.

8.3.2.8 Notation

No additional notational requirements.

8.3.3 ArchitectureDescription

8.3.3.1 Extension

- Package (from UML2)

8.3.3.2 Generalizations

8.3.3.3 Description

An ArchitectureDescription is specification of a system at a technical level that also provides the business context for the system. The ArchitectureDescription contains as many ArchitectureViews as are necessary to fully describe the architecture.

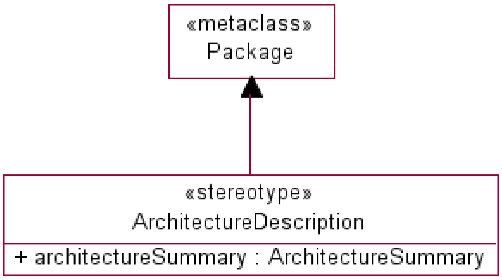


Figure 8.15 - Architecture Description

8.3.3.4 Attributes

- architectureSummary : ArchitectureSummary [1]
  Provides a summary of the architecture description.

8.3.3.5 Associations

No additional associations.

8.3.3.6 Constraints

[1] Asserts that this Model is defined by zero or more ArchitectureSummaries

```
self.architectureSummary->forAll(classifier->forAll(name='ArchitectureSummary'))
or
```

```
self.architectureSummary->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='ArchitectureSummary')))
```

[2] Asserts that this ArchitectureDescription is owned by an Enterprise

```
self.allOwningPackages()->exists(getAppliedStereotype('UPDM::Enterprise')-
>notEmpty())
```

8.3.3.7 Semantics

No additional semantic requirements.

8.3.3.8 Notation

No additional notational requirements.

8.3.4 ArchitectureView

8.3.4.1 Extension

- Package (from UML2)

8.3.4.2 Generalizations

8.3.4.3 Description

An ArchitectureDescription is a collection of ArchitectureViews, which includes Operational View (OV), Systems View (SV), and Technical Standards View (TV), Acquisition View (AcV), and Strategic View (StV). These views are integrated with each other. The Operational Activity to Systems Functionality Traceability Matrix (SV-5), for example, relates operational activities from the Operational Activity Model (OV-5) to system functions from the Systems Functionality Description (SV-4); the SV-4 system functions are related to systems in the Systems Interface Description (SV-1); thus bridging the OV and SV. An architecture is defined to be an integrated architecture when products and their constituent architecture data elements are developed such that architecture DataElements defined in one view are the same (i.e., same names, definitions, and values) as architecture DataElements referenced in another view.

The ArchitectureView is an abstract concept that groups the following kinds of views:

- All View
- Operational View
- System View
- Technical Standards View
- Acquisition View
- Strategic View
- Custom View

In some cases, it is desirable to assemble elements of the model and export them for processing by an external process; perhaps to produce custom presentations or to transform them into different forms. Using the UPDMAttributes, an external URI can be designated as the processing element, and the components within the View can be assembled as needed to complement the external process.

An ArchitectureView is a ConformingElement and thus can be related to Standards.

ArchitectureView can be used to implement the Viewpoint/View capability. As a UML Package, ArchitectureView is a nested construct. Using a CustomView, a new Viewpoint can be defined, and custom views can be defined within this new package. The result is a Viewpoint consisting of these custom views. The new views can be constructed using query capabilities.

The ArchitectureViews are defined based on Viewpoints, as specified in this specification. An ArchitectureView may conform to at most one Viewpoint. For all standard views, the viewpoints are implicit and defined as part of this specification. For example, this specification defines the viewpoint AV-1, and a product that adheres to this viewpoint is described in a view marked as AV-1. It is possible to define a custom viewpoint to which a custom views adheres.

An Address relationship may be used to indicate which Stakeholders or Concerns are addressed by an ArchitectureView.

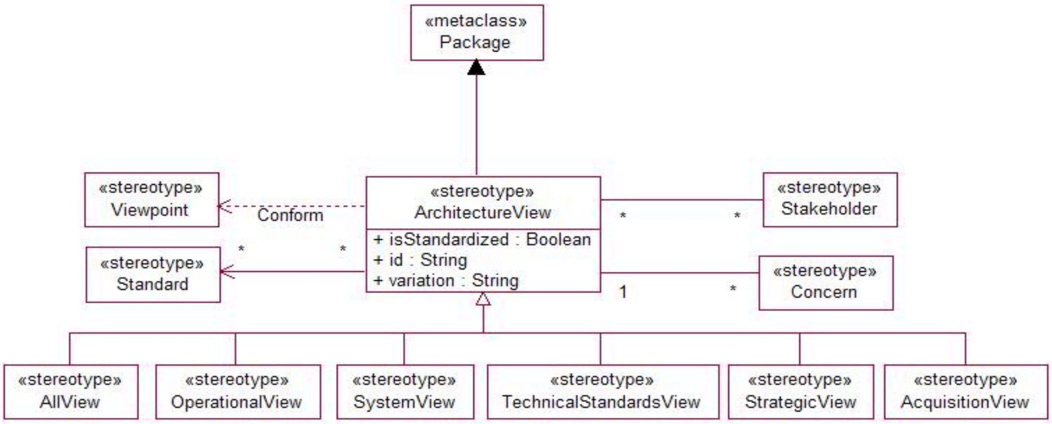


Figure 8.16 - ArchitectureView

8.3.4.4 Attributes

- id : String [1]
  An identifier for the ArchitectureView package. It should be a universal unique identifier.
- isStandardized : Boolean [1]
  An indication that this view is standardized.
- variation : String [1]

8.3.4.5 Associations

- concerns : Concern [*]
  Concerns drive viewpoint selection: each concern is addressed by an architectural view
- stakeholders : Stakeholder [*] Each viewpoint is specified by:
  - Viewpoint name
  - The stakeholders addressed by the viewpoint
  - The stakeholder concerns to be addressed by the viewpoint
  - The viewpoint language, modeling techniques, or analytical methods used
  - The source, if any, of the viewpoint (e.g., author, literature citation)
- standards : Standard [*]
  The Standards to which this ArchitectureView conforms

8.3.4.6 Constraints

[1] Asserts that Concerns is not empty

```
self.concerns-> forAll(getAppliedStereotype('UPDM::Concern')->notEmpty())
```

[2] Asserts that stakeholders are stereotyped Stakeholders

```
self.stakeholders-> forAll (getAppliedStereotype('UPDM::Stakeholder')->
notEmpty())
```

[3] Asserts that a package stereotyped ArchitectureView is contained in a package stereotyped either ArchitectureDescription or ArchitectureView.

```
self.owner->forAll(

getAppliedStereotype('UPDM::ArchitectureView')-> notEmpty() or

getAppliedStereotypes()->collect(allParents())-> any(qualifiedName =
'UPDM::ArchitectureView') -> notEmpty() or

getAppliedStereotype('UPDM::ArchitectureDescription')->notEmpty()

)
```

[4] Asserts that a package stereotyped ArchitectureView cannot be contained in a package that is stereotyped with a subclass of ArchitectureView.

```
self.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::ArchitectureView') ->notEmpty() and

self.allOwningPackages()->any(

   getAppliedStereotypes()->collect(allParents())->
      any(qualifiedName = 'UPDM::ArchitectureView') ->notEmpty()

   )->isEmpty())
```

[5] Asserts that an ArchitectureView may Conform to at most one Viewpoint.

```
self.client->forAll(getAppliedStereotype('UPDM::ArchitectureView')->notEmpty()
or

getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::ArchitectureView') ->notEmpty()) and

self.supplier->forAll(getAppliedStereotype('UPDM::Viewpoint')->notEmpty())
```

[6] Asserts that a package can only have one ArchitectureView stereotype applied

```
self.getAppliedStereotypes()->select(s| s.allParents()->any(qualifiedName =
'UPDM::ArchitectureView')->notEmpty() )->size()<=1
```

[7] Asserts that the element has at least one Standard associated with it.

```
self.standards->forAll(getAppliedStereotype('UPDM::Standard')->notEmpty())
```

8.3.4.7 Semantics

No additional semantic requirements.

8.3.4.8 Notation

No additional notational requirements.

8.3.5 AV-1: Overview and Summary

8.3.5.1 Extension

8.3.5.2 Generalizations

- AllView (from AllViews)

8.3.5.3 Description

The Overview and Summary Information provides 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.

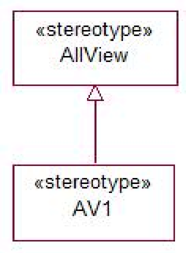


Figure 8.17 - AV-1 AllView Overview and Summary

8.3.5.4 Attributes

No additional attributes.

8.3.5.5 Associations

No additional associations.

8.3.5.6 Constraints

No additional constraints.

8.3.5.7 Semantics

No additional semantic requirements.

8.3.5.8 Notation

No additional notational requirements.

8.3.6 AV-2: Integrated Dictionary

8.3.6.1 Extension

8.3.6.2 Generalizations

- AllView (from AllViews)

8.3.6.3 Description

This view is the Integrated Dictionary that contains definitions of all the key concepts and model elements in an architectural model.

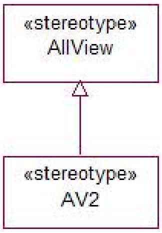


Figure 8.18 - AV-2 AllView Integrated Dictionary

8.3.6.4 Attributes

No additional attributes.

8.3.6.5 Associations

No additional associations.

8.3.6.6 Constraints

No additional constraints.

8.3.6.7 Semantics

No additional semantic requirements.

8.3.6.8 Notation

No additional notational requirements.

8.3.7 AVCustom: AllView Custom View

8.3.7.1 Extension

8.3.7.2 Generalizations

- AllView (from AllViews)

8.3.7.3 Description

A user-defined view that applies to the AllView.

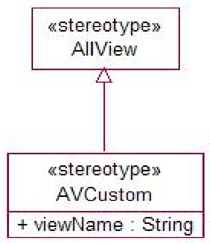


Figure 8.19 - AVCustom: AllView Custom View

8.3.7.4 Attributes

- viewName : String [1]
  Provides a name for the custom view so that it can be easily distinguished from other custom AllViews.

8.3.7.5 Associations

No additional associations.

8.3.7.6 Constraints

No additional constraints.

8.3.7.7 Semantics

No additional semantic requirements.

8.3.7.8 Notation

No additional notational requirements.

8.3.8 Concern

8.3.8.1 Extension

- UseCase (from UML2)

8.3.8.2 Generalizations

8.3.8.3 Description

An interest in a subject held by one or more Stakeholders.

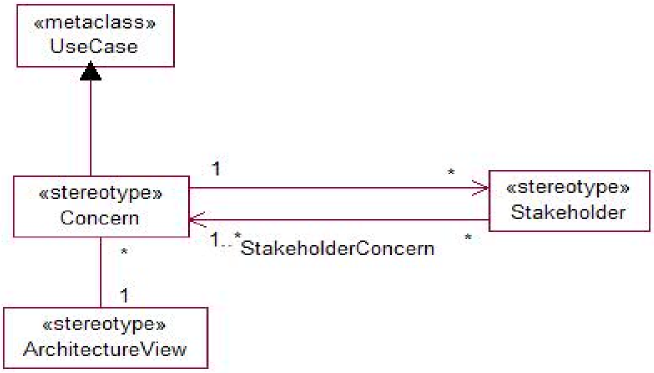


Figure 8.20 - Concern

8.3.8.4 Attributes

No additional attributes.

8.3.8.5 Associations

- architectureview : ArchitectureView [1]
- stakeholders : Stakeholder [*]

8.3.8.6 Constraints

[1] Asserts that this Concern links an ArchitecureView to a set of Stakeholders

```
self.architectureView.getAppliedSubstereotypes(self.getApplicableStereotypes()-
>any(qualifiedName='UPDM::ArchitectureView'))->notEmpty()
```

[2] Asserts that there are zero or more Stakeholders who have this Concern

```
self.stakeholders-> forAll(getAppliedStereotype('UPDM::Stakeholder')->notEmpty())
```

8.3.8.7 Semantics

No additional semantic requirements.

8.3.8.8 Notation

No additional notational requirements.

8.3.9 Conform

8.3.9.1 Extension

- Dependency (from UML2)

8.3.9.2 Generalizations

8.3.9.3 Description

A Conform relationship is a dependency between an ArchitectureView and a Viewpoint. The view conforms to the specified rules and conventions detailed in the Viewpoint.

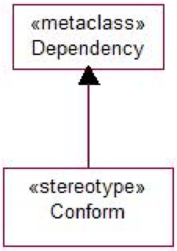


Figure 8.21 - Conform

8.3.9.4 Attributes

No additional attributes.

8.3.9.5 Associations

No additional associations.

8.3.9.6 Constraints

[1] Asserts that the client of a Conform dependency is an ArchitectureView and that the supplier of a Conform dependency is a Viewpoint.

```
self.client->forAll(getAppliedStereotype('UPDM::ArchitectureView')-> notEmpty()
   or

getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
   'UPDM::ArchitectureView') ->notEmpty()) and

self.supplier->forAll(getAppliedStereotype('UPDM::Viewpoint')->notEmpty())
```

8.3.9.7 Semantics

No additional semantic requirements.

8.3.9.8 Notation

No additional notational requirements.

8.3.10 ConformingElement

8.3.10.1 Extension

- Class (from UML2)

8.3.10.2 Generalizations

8.3.10.3 Description

ConformingElement provides a link from profile elements to specific standards in the context of a forecast.

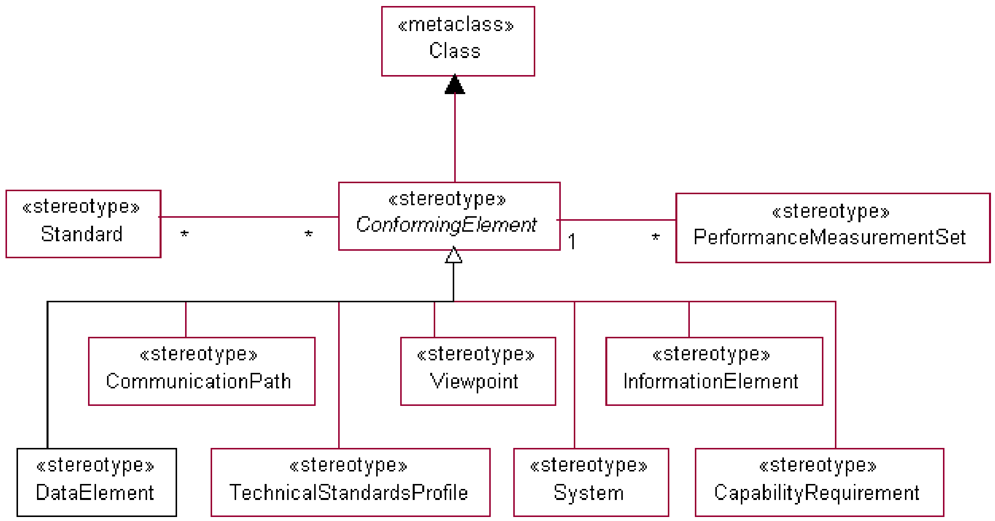

«metaclass» Class
«stereotype» Standard
*
*
«stereotype» ConformingElement
1
*
«stereotype» PerformanceMeasurementSet
«stereotype» CommunicationPath
«stereotype» Viewpoint
«stereotype» InformationElement
«stereotype» DataElement
«stereotype» TechnicalStandardsProfile
«stereotype» System
«stereotype» CapabilityRequirement

Figure 8.22 - Conforming Element

8.3.10.4 Attributes

No additional attributes.

8.3.10.5 Associations

- performanceMeasurements : PerformanceMeasurementSet [*]
  A CapabilityPerformanceSet that bounds the ConformingElement
- standards : Standard [*]
  The Standards to which this ConformingElement conforms
- performanceparameterset : PerformanceParamenterSet [*]
  The class that contains the specification for the performance parameters for this ConformingElement

8.3.10.6 Constraints

[1] Asserts that the element has at least one Standard associated with it.

```
self.standards->forAll(getAppliedStereotype('UPDM::Standard')->notEmpty())
```

[2] Asserts that a CapabilityPerformanceSet exists that bounds the ConformingElement

```
self.performanceMeasurements->forAll(getAppliedStereotype
   ('UPDM::PerformanceMeasurementSet')->notEmpty())
```

[3] Asserts that the entries are PerformanceParameterSets

```
self.performanceParameterSet-> forAll(getAppliedStereotype
   ('UPDM::PerformanceParameterSet')->notEmpty())
```

8.3.10.7 Semantics

No additional semantic requirements.

8.3.10.8 Notation

No additional notational requirements.

8.3.11 Doctrine

8.3.11.1 Extension

- Class (from UML2)

8.3.11.2 Generalizations

8.3.11.3 Description

The setting in which the architecture exists comprises the interrelated conditions that compose the context for the architecture. These conditions include Doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations (CONOPS); scenarios; and environmental conditions.

Doctrine is applied to CapabilityConfigurations and OperationalTask and SystemTask.

Doctrine is adopted by an Enterprise.

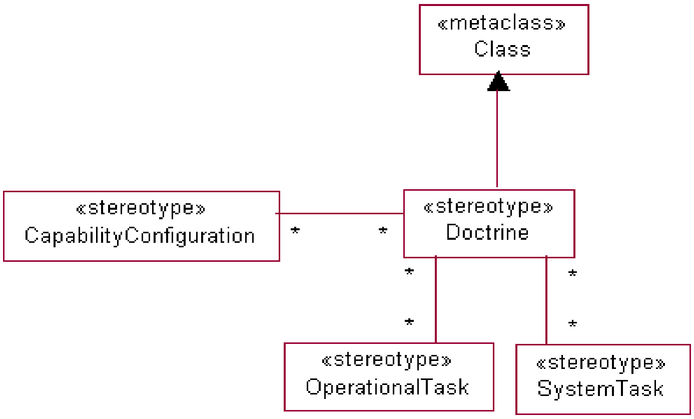


Figure 8.23 - Doctrine

8.3.11.4 Attributes

No additional attributes.

8.3.11.5 Associations

- capabilityConfigurations : CapabilityConfiguration [*]
  Capabilities governed by this Doctrine
- systemtask : SystemTask [*]
  The SystemTasks governed by this Doctrine
- tasks : OperationalTask [*]
  The OperationalTasks governed by this Doctrine

8.3.11.6 Constraints

[1] Asserts that there are SystemTasks or OperationalTasks governed by this Doctrine

```
self.tasks-> forAll(getAppliedStereotype('UPDM::OperationalTask')->notEmpty()
   or getAppliedStereotype('UPDM::SystemTask')->notEmpty())
```

[2] Asserts that this is a Doctrine of the Enterprise and that it is owned by the Enterprise package

```
self.allOwningPackages()->exists(getAppliedStereotype('UPDM::Enterprise')-
   >notEmpty())
```

[3] Asserts that this Doctrine governs zero or more CapabilityConfigurations

```
self.capabilityConfigurations-> forAll(getAppliedStereotype
   ('UPDM::CapabilityConfiguration')->notEmpty())
```

8.3.11.7 Semantics

No additional semantic requirements.

8.3.11.8 Notation

No additional notational requirements.

8.3.12 Enterprise

8.3.12.1 Extension

- Package (from UML2)

8.3.12.2 Generalizations

8.3.12.3 Description

An Enterprise is an endeavor of any size involving people, organizations and supporting systems. It is used to hold all the artifacts in a top level package including Doctrine references, Architecture Descriptions, Strategic Missions, etc.

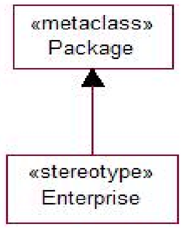


Figure 8.24 - Enterprise

8.3.12.4 Attributes

No additional attributes.

8.3.12.5 Associations

No additional associations.

8.3.12.6 Constraints

[1] Asserts that Enterprise has at least one Doctrine.

```
self.oclAsType(uml::Package).allOwnedElements()-> exists(getAppliedStereotype
   ('UPDM::Doctrine')->notEmpty())->notEmpty()
```

[2] Asserts that there are one or more StrategicMissions held by this Enterprise

```
self.oclAsType(uml::Package).allOwnedElements()-> exists(getAppliedStereotype
   ('UPDM::StrategicMission')->notEmpty())-> notEmpty()
```

[3] Asserts that the Enterprise owns at least one ArchitectureDescription

```
self.oclAsType(uml::Package).allOwnedElements()-> exists(getAppliedStereotype
   ('UPDM::ArchitectureDescription')->notEmpty())-> notEmpty()
```

8.3.12.7 Semantics

No additional semantic requirements.

8.3.12.8 Notation

No additional notational requirements.

8.3.13 Goal

8.3.13.1 Extension

- Class (from UML2)

8.3.13.2 Generalizations

8.3.13.3 Description

A Goal is a specific, required objective of the enterprise that the architecture represents. Some writers distinguish between goals (inexactly formulated aims that lack specificity) and objectives (aims formulated exactly and quantitatively as to time-frames and magnitude of effect).

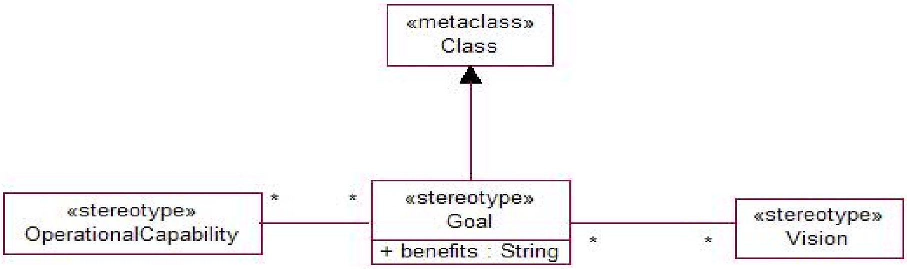


Figure 8.25 - Goal

8.3.13.4 Attributes

- benefits : String [1]

8.3.13.5 Associations

- operationalCapabilities : OperationalCapability [*]
  The OperationalCapabilities that will help achieve this Goal
- vision : Vision [*]
  Summary of Goals and Objectives

8.3.13.6 Constraints

[1] Asserts that there are OperationalCapabilities associated with this Goal.

```
self.operationalCapabilities-> forAll (getAppliedStereotype
   ('UPDM::OperationalCapability')->notEmpty())
```

[2] Asserts at lease one Vision statement that entails this Goal

```
self.vision-> forAll(getAppliedStereotype('UPDM::Vision')->notEmpty())
```

8.3.13.7 Semantics

No additional semantic requirements.

8.3.13.8 Notation

No additional notational requirements.

8.3.14 PerformanceMeasureSet

8.3.14.1 Extension

8.3.14.2 Generalizations

- Specification (from AllViews)

8.3.14.3 Description

A PerformanceMeasurementSet includes a set of PerformanceMeasures for each PerformanceMeasurePeriod. The set is applied to ConformingElements. (See UPDM Class Library). This stereotype is applied to instances of classes stereotyped PerformanceParameterSet.

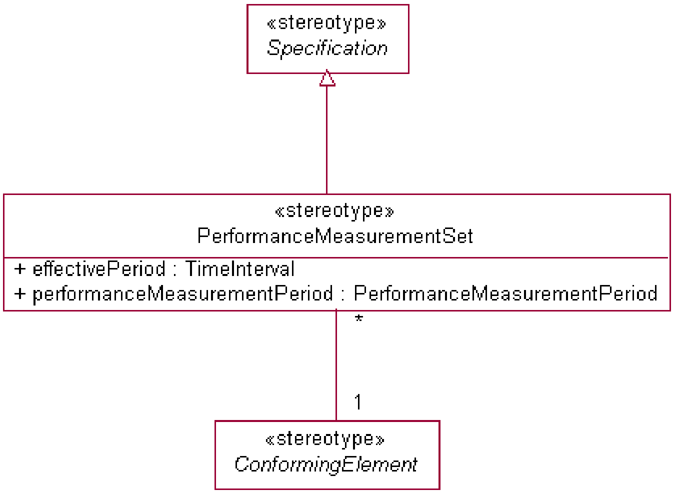

«stereotype»
Specification
«stereotype»
PerformanceMeasurementSet
+ effectivePeriod : TimeInterval
+ performanceMeasurementPeriod : PerformanceMeasurementPeriod
*
1
«stereotype»
ConformingElement

Figure 8.26 - PerformanceMeasureSet

8.3.14.4 Attributes

- performanceMeasurementPeriod: PerformanceMeasurementPeriod [1]
  One of Baseline, Target or Actual
- effectivePeriod: TimeInterval [1]
  Time period for which the measurement is valid

8.3.14.5 Associations

- conformingelement : ConformingElement [1]
  ConformingElement to which this measurement set applies

8.3.14.6 Constraints

[1] Asserts that the PerformanceMeasurementSet applies to a ConformingElement.

```
self.conformingElement-> forAll(
getApplicableStereotypes()->collect(allParents()->

any(qualifiedName = 'UPDM::ConformingElement')->notEmpty())
```

8.3.14.7 Semantics

In use, failure to live up to the PerformanceMeasures will result in a performance gap for the System.

8.3.14.8 Notation

No additional notational requirements.

8.3.15 PerformanceParameterSet

8.3.15.1 Extension

- Class (from UML2)

8.3.15.2 Generalizations

8.3.15.3 Description

This Stereotype is applied to the PerformanceMeasurementSet in the UPDM ClassLibrary.

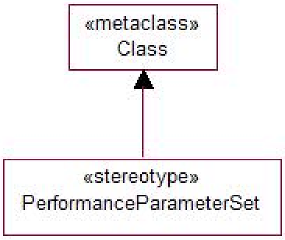


Figure 8.27 - PerformanceParameterSet

8.3.15.4 Attributes

No additional attributes.

8.3.15.5 Associations

- conformingelement : ConformingElement [1]
  ConformingElements to which this measurement set applies.

8.3.15.6 Constraints

No additional constraints.

8.3.15.7 Semantics

No additional semantic requirements.

8.3.15.8 Notation

No additional notational requirements.

8.3.16 PerformanceParameterType

8.3.16.1 Extension

- Property (from UML2)

8.3.16.2 Generalizations

8.3.16.3 Description

PerformanceParameterType provides details for the type of attribute for PerformanceParameterSet.

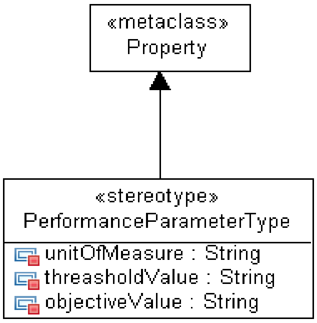


Figure 8.28 - PerformanceParameterType

8.3.16.4 Attributes

- unitOfMeasure : String [1]
- objectiveValue : String [1]
- thresholdValue : String [1]

8.3.16.5 Associations

No additional associations.

8.3.16.6 Constraints

No additional constraints.

8.3.16.7 Semantics

PerfromanceParameterType provides additional attributes to a type that is specified for an attribute of the PerformanceParameterSet.

8.3.16.8 Notation

No additional notational requirements.

8.3.16.9 Requirement

8.3.16.10 Extension

- Class (from UML2)

8.3.16.11 Generalizations

8.3.16.12 Description

A Requirement associates a textual statement of a requirement with one or more elements in the model and zero or more external references.

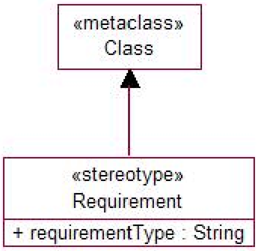


Figure 8.29 - Requirement

8.3.16.13 Attributes

- requirementType : String [1]

8.3.16.14 Associations

No additional associations.

8.3.16.15 Constraints

No additional constraints.

8.3.16.16 Semantics

A requirement traces a requirement to one or more elements that satisfy the requirement or specify the requirement in detail such as a Use Case or OperationalActivity.

8.3.16.17 Notation

No additional notational requirements.

8.3.17 Resource

8.3.17.1 Extension

- Class (from UML2)

8.3.17.2 Generalizations

8.3.17.3 Description

A Resource is something that is able to supply functionality, information or material. It can be a SystemsNode that represents facilities, platforms, units, and locations or OrganizationalResource that can contribute towards fulfilling a Capability.

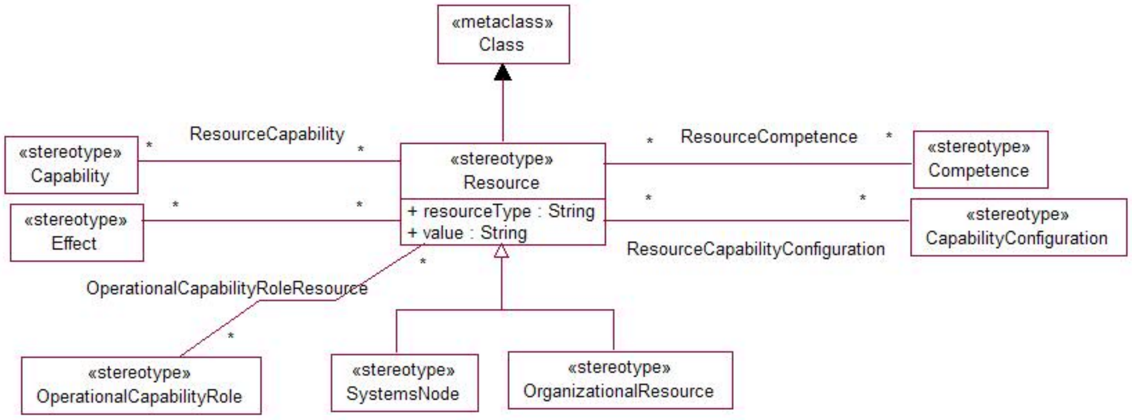


Figure 8.30 - Resource

8.3.17.4 Attributes

- resourceType : String [1]
  The type of resource - organization, organization type, human, etc.

- value : String [1]
  The value of the resource to the enterprise. This value is used in risk assessment and mitigation calculations. It could be a monetary amount, a critical supply scale, etc.

8.3.17.5 Associations

- capability : Capability [*]
  See ResourceCapability for more details.

- capabilityConfiguration : CapabilityConfiguration [*]
  See ResourceCapabilityConfiguration for more details.

- competence : Competence [*]
  See ResourceCompetence for more details.

- effect : Effect [*]
  The effects that affect the resource.

- operationalCapabilityRole : OperationalCapabilityRole [*]
  The OperationalCapabilityRoles that are filled by this Resource

8.3.17.6 Constraints

[1] Asserts that zero or more effects affect this resource.

```
self.effect-> forAll(getAppliedStereotype('UPDM::Effect')->notEmpty())
```

[2] Asserts that a ResourceCapability association exists between a Resource and the Capabilities that it provides

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()
   ).association->any

   (getAppliedStereotype('UPDM::ResourceCapability')-> notEmpty())->notEmpty()
```

[3] Asserts that a ResourceCapabilityConfiguration association exists between Resource and CapabilityConfiguration

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()
   ).association->any

(getAppliedStereotype('UPDM::ResourceCapabilityConfiguration')-> notEmpty())-
   >notEmpty()
```

[4] Asserts that a ResourceCompetence association exists between Resource and the Competence that it provides

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()
   ).association->any

   (getAppliedStereotype('UPDM::ResourceCompetence')-> notEmpty())->notEmpty()
```

[5] Asserts that there is an association between a Competence and the OperationalCapabilityRole that requires it.

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()
   ).association->any

(getAppliedStereotype('UPDM::OperationalCapabilityRoleResource')->
   notEmpty())->notEmpty()
```

8.3.17.7 Semantics

No additional semantic requirements.

8.3.17.8 Notation

No additional notational requirements.

8.3.18 Specification

8.3.18.1 Extension

- InstanceSpecification (from UML2)

8.3.18.2 Generalizations

8.3.18.3 Description

Specification is the ancestor of all the stereotypes that extend Instance Specification. It implements the common attribute, description.

In general, any of the classes supplied in the class library as the type for instance specifications can be extended to include additional attributes to be used in custom instance specifications.

Instances are created from classes that are defined in the UPDM ModelLibrary. The values of the attributes of the parent class are set in the slots of the instance. Associations among instances of classes that reside in the Model Library are specified by creating a link that is an instance of an association that is defined on the classes in the Model Library. In addition, the stereotyped instance may have stereotype properties that define relationships to other stereotyped classes.

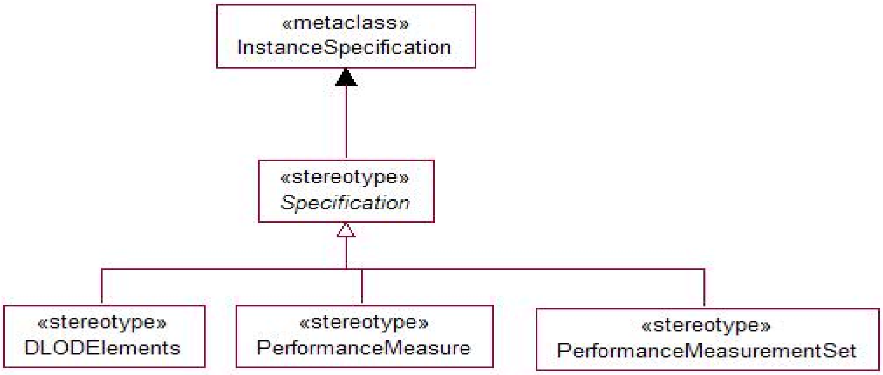


Figure 8.31 - Specification

8.3.18.4 Attributes

No additional attributes.

8.3.18.5 Associations

No additional associations.

8.3.18.6 Constraints

No additional constraints.

8.3.18.7 Semantics

No additional semantic requirements.

8.3.18.8 Notation

No additional notational requirements.

8.3.19 Stakeholder

8.3.19.1 Extension

8.3.19.2 Generalizations

- OrganizationalResource (from OperationalView)

8.3.19.3 Description

An individual, team or Organization (or classes thereof) with interests in, or concerns relative to, a system. Some typical stakeholders include: Client; Acquirer; Owner; User; Operator; Architect; System Engineer; Developer; Designer; Builder; Maintainer; Service Provider; Vendor; Subcontractor; Planned.

Stakeholder links the Concerns of the Stakeholder with the Viewpoints.

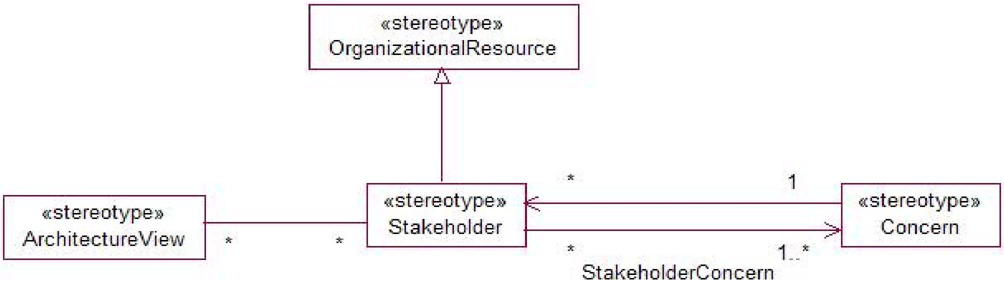


Figure 8.32 - Stakeholder

8.3.19.4 Attributes

No additional attributes.

8.3.19.5 Associations

- architectureView : ArchitectureView [*]
  ArchitectureViews associated with this Stakeholder
- concern : Concern [1..*]
  An interest in a subject held by one or more Stakeholders.

8.3.19.6 Constraints

[1] Asserts that there is at least one ArchitectureView associated with this Stakeholder

```
self.architectureView-> forAll(getAppliedStereotype('UPDM::ArchitectureView')-
   >notEmpty())
```

[2] Asserts that this Stakeholder has a Concern

```
self.getAllAttributes()->asOrderedSet()->
```

```
select(association->notEmpty()).association->select
   (getAppliedStereotype('UPDM::StakeholderConcern')-> notEmpty())->notEmpty()
```

8.3.19.7 Semantics

No additional semantic requirements.

8.3.19.8 Notation

No additional notational requirements.

8.3.20 StakeholderConcern

8.3.20.1 Extension

- Association (from UML2)

8.3.20.2 Generalizations

8.3.20.3 Description

A relationship that relates a Stakeholder with a Concern.

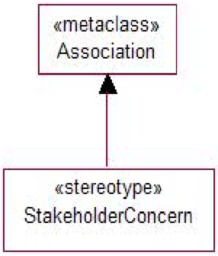


Figure 8.33 - StakeholderConcern

8.3.20.4 Attributes

No additional attributes.

8.3.20.5 Associations

No additional associations.

8.3.20.6 Constraints

[1] Asserts that the ends of the association are Stakeholder and Concern

```
(self.endType->at(1).getAppliedStereotype('UPDM::Stakeholder')->notEmpty() and
self.endType->at(2).getAppliedStereotype('UPDM::Concern')->notEmpty()) or
(self.endType->at(2).getAppliedStereotype('UPDM::Stakeholder')->notEmpty() and
self.endType->at(1).getAppliedStereotype('UPDM::Concern')->notEmpty())
```

8.3.20.7 Semantics

No additional semantic requirements.

8.3.20.8 Notation

No additional notational requirements.

8.3.21 StrategicMission

8.3.21.1 Extension

- UseCase (from UML2)

8.3.21.2 Generalizations

8.3.21.3 Description

A StrategicMission summarize goals and objectives into a mission statement.

A Mission statement may define the purpose or broader goal for being in existence or in the business. It serves as an ongoing guide without time frame. The mission can remain the same for decades if crafted well. Vision is more specific in terms of objective and future state. Vision is related to some form of achievement if successful.

Mission and Values go hand in hand. To make the mission statement effective, it needs to be aligned with the prevailing culture of its stakeholders, organization, market and political sphere. A lofty mission statement means nothing if it is not in congruence with the values practiced by the organization. A statement of values provides guiding principles when ethical issues related to realizing the Vision, and undertaking the Mission, arise.

A mission statement provides a path to realize the vision in line with its values. These statements have a direct bearing on the bottom line and success of the organization.

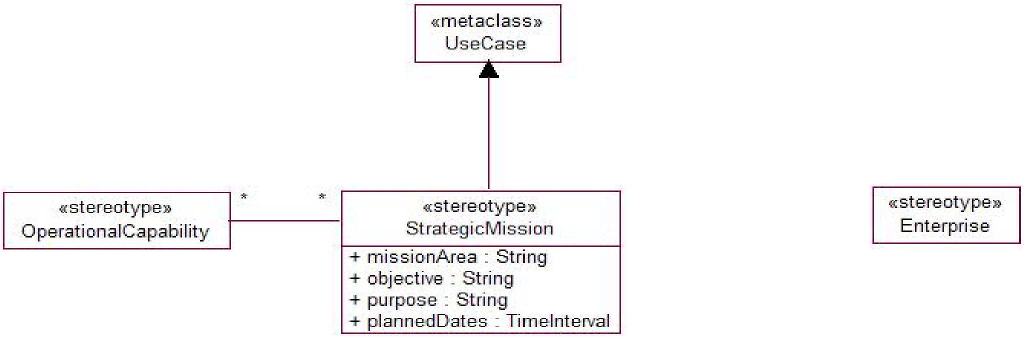


Figure 8.34 - StrategicMission

8.3.21.4 Attributes

- missionArea : String [1]

• objective : String [1]	Statement of the objectives to be achieved
• plannedDates : TimeInterval [1]	The planned start and end effective dates
• purpose : String [1]	A broader goal for being in existence

8.3.21.5 Associations

- operationalCapabilities : OperationalCapability [*]
  Multiple tasks, i.e., OperationalCapabilities, accomplish a mission. (Space and Naval Warfare Systems Command).

8.3.21.6 Constraints

[1] Asserts that zero or more OperationalCapabilities have been designated as instrumental in achieving the StrategicMission.

```
self.operationalCapabilities-> forAll(getAppliedStereotype
   ('UPDM::OperationalCapability')->notEmpty())
```

[2] Asserts that this is a Strategic Mission of the Enterprise and that it is owned by the Enterprise package

```
self.allOwningPackages()->exists(getAppliedStereotype('UPDM::Enterprise')-
   >notEmpty())
```

[3] The planned start and end effective dates

```
self.plannedDates->forAll(classifier->forAll(name='TimeInterval')) or
```

```
self.plannedDates->forAll(classifier->forAll(allParents()->notEmpty() and
   allParents()->exists(name='TimeInterval')))
```

8.3.21.7 Semantics

The StrategicMission along with the Vision and Goals of the Enterprise are the fundamental elements that provide the context for the Systems that are contained in the Model and described in the ArchitectureDescription.

(See: Vision, Goals, Enterprise, Model and ArchitectureDescription)

Military Mission

An objective together with the purpose of the intended action. (Extension of DDDS 1(A))

Note: Multiple tasks accomplish a mission. (Space and Naval Warfare Systems Command)

8.3.21.8 Notation

No additional notational requirements.

8.3.22 ExternalReferences

8.3.22.1 Extension

- NamedElement (from UML2)

8.3.22.2 Generalizations

8.3.22.3 Description

A general stereotype that can be applied to any element in a model that applies the profile.

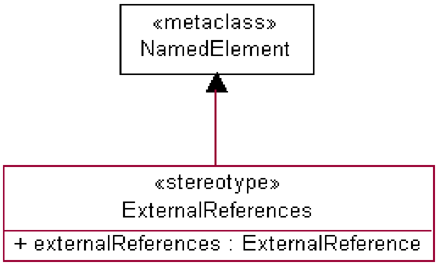


Figure 8.35 - ExternalReferences

8.3.22.4 Attributes

- externalReferences : ExternalReference [*]
  The ExternalReferences that apply to this UPDMElement

8.3.22.5 Associations

No additional associations

8.3.22.6 Constraints

[1] Asserts that zero or more ExternalReferences apply to this UPDM Element

```
self.externalReferences->forAll(classifier->forAll(name='ExternalReference'))
   or

self.externalReferences->forAll(classifier->forAll(allParents()->notEmpty() and
   allParents()->exists(name='ExternalReference')))
```

8.3.22.7 Semantics

Establishes a mechanism to associate any element in the model with any number of references that may be external to the model. (See ExternalReference)

8.3.22.8 Notation

No additional notational requirements.

8.3.23 Viewpoint

8.3.23.1 Extension

8.3.23.2 Generalizations

- ConformingElement (from AllViews)

8.3.23.3 Description

A Viewpoint is a set of conventions for constructing, interpreting, and analyzing a view (in accordance with IEEE 1471).

- A viewpoint is a pattern for constructing views
- Viewpoints define the rules on views. There is no fixed set of viewpoints.
- Each viewpoint used in an ArchitectureDescription is “declared” before use

A viewpoint may also include:

- Any consistency or completeness checks associated with the underlying method to be applied to models within the view
- Any evaluation or analysis techniques to be applied to models within the view
- Any heuristics, patterns, or other guidelines which aid in the synthesis of an associated view or its models

Each viewpoint is specified by:

- Viewpoint name
- The stakeholders addressed by the viewpoint
- The stakeholder concerns to be addressed by the viewpoint
- The viewpoint language, modeling techniques, or analytical methods used
- The source, if any, of the viewpoint (e.g., author, literature citation)

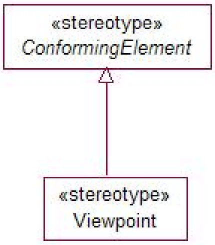


Figure 8.36 - Viewpoint

8.3.23.4 Attributes

No additional attributes.

8.3.23.5 Associations

No additional associations.

8.3.23.6 Constraints

No additional constraints.

8.3.23.7 Semantics

No additional semantic requirements.

8.3.23.8 Notation

No additional notational requirements.

8.3.24 Vision

8.3.24.1 Extension

- Class (from UML2)

8.3.24.2 Generalizations

8.3.24.3 Description

A Vision is a high-level textual description of the “aims” of the Enterprise over a given TimePeriod. Organizations sometimes summarize Goals and objectives into a mission statement and / or a vision statement.

A Vision statement describes, in graphic terms, where the goal-setters want to see themselves in the future. It may describe how they see events unfolding over 10 or 20 years given a set of assumptions.

The Vision describes a future complex state and the StrategicMission describes how it will be achieved. Vision is more specific in terms of objective and future state. Vision is related to some form of achievement or Effect if successful.

For example, “We help transport goods and people efficiently and cost effectively without damaging environment” is a mission statement. Ford’s brief but powerful slogan “Quality is Job 1” could count as a mission statement. “We will be one among the top three transporters of goods and people in North America by 2010” is a vision statement. It is very concrete and unambiguous goal.

The Vision statement can galvanize the people to achieve defined objectives, even if they are stretch objectives, provided the vision is SMART (Specific, Measurable, Achievable, Realistic and Time bound). A StrategicMission statement provides a path to realize the Vision in line with its values. These statements have a direct bearing on the bottom line and success of the organization.

Features of an effective Vision statement may include:

- Clarity and lack of ambiguity
- Paint a vivid and clear picture, not ambiguous
- Describing a bright future (hope)
- Memorable and engaging expression
- Realistic aspirations, achievable
- Alignment with organizational values and culture, Rational
- Time bound if it talks of achieving any goal or objective

In order to become really effective, an organizational Vision statement must (the theory states) become assimilated into the organization’s culture. Leaders have the responsibility of communicating the vision regularly, creating narratives that make the vision clear to the members of the enterprise.

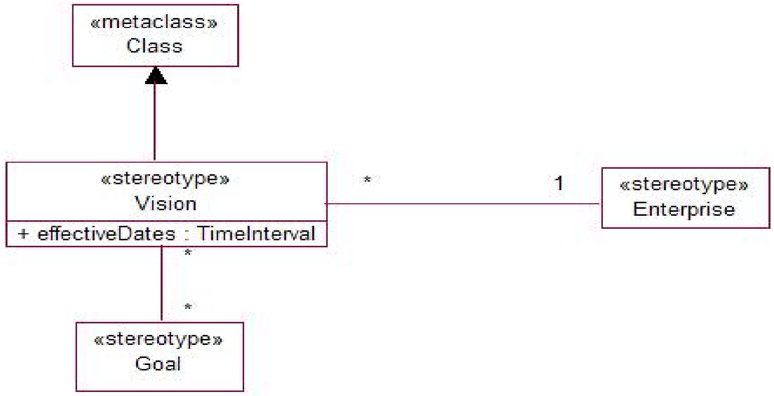


Figure 8.37 - Vision

8.3.24.4 Attributes

- effectiveDates : TimeInterval [1]
  Dates for which the Vision is applicable

8.3.24.5 Associations

- enterprise : Enterprise [1]
  The Enterprise that owns this Vision
- goal : Goal [*]

8.3.24.6 Constraints

[1] Asserts that the beginning and end dates are a TimeInterval

```
self.effectiveDates->forAll(classifier->forAll(name='TimeInterval')) or

self.effectiveDates->forAll(classifier->forAll(allParents()->notEmpty() and
   allParents()->exists(name='TimeInterval')))
```

[2] Asserts that there are zero or more Goals defined for the Vision

```
self.goal-> forAll(getAppliedStereotype('UPDM::Goal')->notEmpty())
```

[3] Asserts that this is a Vision of the Enterprise and that it is owned by the Enterprise package

```
self.allOwningPackages()->exists(getAppliedStereotype('UPDM::Enterprise')-
   >notEmpty())
```

8.3.24.7 Semantics

No additional semantic requirements.

8.3.24.8 Notation

No additional notational requirements.

8.4 OperationalView Package

8.4.1 Overview

The OperationalView Package contains UML stereotypes that assist the modeler in developing the views defined in the Operational View viewpoint. These stereotypes support the description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. The OperationalView Package also includes elements for the identification of the operational nodes, assigned tasks and activities, and information flows required between nodes.

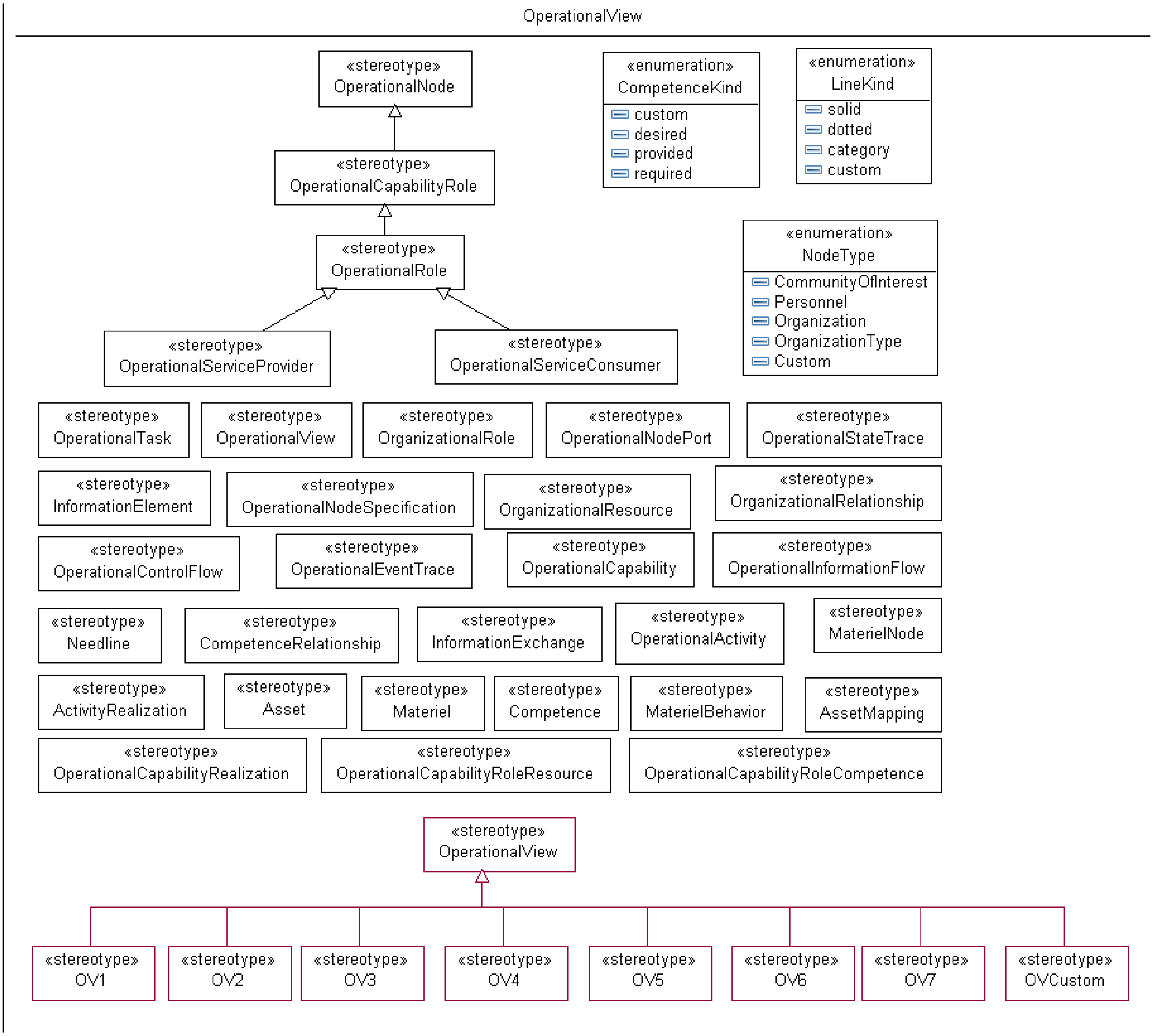

OperationalView
«stereotype» OperationalNode
«stereotype» OperationalCapabilityRole
«stereotype» OperationalRole
«stereotype» OperationalServiceProvider
«stereotype» OperationalServiceConsumer
«enumeration» CompetenceKind
custom
desired
provided
required
«enumeration» LineKind
solid
dotted
category
custom
«enumeration» NodeType
CommunityOfInterest
Personnel
Organization
OrganizationType
Custom
«stereotype» OperationalTask
«stereotype» OperationalView
«stereotype» OrganizationalRole
«stereotype» OperationalNodePort
«stereotype» OperationalStateTrace
«stereotype» InformationElement
«stereotype» OperationalNodeSpecification
«stereotype» OrganizationalResource
«stereotype» OrganizationalRelationship
«stereotype» OperationalControlFlow
«stereotype» OperationalEventTrace
«stereotype» OperationalCapability
«stereotype» OperationalInformationFlow
«stereotype» Needline
«stereotype» CompetenceRelationship
«stereotype» InformationExchange
«stereotype» OperationalActivity
«stereotype» MaterielNode
«stereotype» ActivityRealization
«stereotype» Asset
«stereotype» Materiel
«stereotype» Competence
«stereotype» MaterielBehavior
«stereotype» AssetMapping
«stereotype» OperationalCapabilityRealization
«stereotype» OperationalCapabilityRoleResource
«stereotype» OperationalCapabilityRoleCompetence
«stereotype» OperationalView
«stereotype» OV1
«stereotype» OV2
«stereotype» OV3
«stereotype» OV4
«stereotype» OV5
«stereotype» OV6
«stereotype» OV7
«stereotype» OVCustom

Figure 8.38 - OperationalView Package

8.4.2 ActivityRealization

8.4.2.1 Extension

- Association (from UML2)

8.4.2.2 Generalizations

8.4.2.3 Description

Maps the UPDM activities (OperationalActivity and SystemFunction) to a collection that contains the elements that collaborate to realize these behaviors. This collection owns a behavior that is expressed in terms of the activities or interactions.

8.4.2.4 Attributes

No additional attributes.

8.4.2.5 Associations

No additional associations.

8.4.2.6 Constraints

[1] Asserts that one end of the association is an Activity, either an OperationalActivity or a SystemFunction and the other is an OperationalActivityRealization if the first end is an **OperationalActivity** and an OperationalCapabilityRealization if the first end is a SystemFunction.

```
((self.endType->at(1).getAppliedStereotype('UPDM::OperationalActivity')-
   >notEmpty() and

self.endType->at(2).getAppliedStereotype
   ('UPDM::OperationalActivityRealization')->notEmpty() )

or

(self.endType->at(2).getAppliedStereotype('UPDM::OperationalActivity')-
   >notEmpty() and

self.endType->at(1).getAppliedStereotype
   ('UPDM::OperationalActivityRealization')->notEmpty()))

or

((self.endType->at(1).getAppliedStereotype('UPDM::SystemFunction')->notEmpty()
and

self.endType->at(2).getAppliedStereotype
   ('UPDM::OperationalCapabilityRealization')->notEmpty() )

or

(self.endType->at(2).getAppliedStereotype('UPDM::SystemFunction')->notEmpty()
and

self.endType->at(1).getAppliedStereotype
   ('UPDM::OperationalCapabilityRealization')->notEmpty()))
```

8.4.2.7 Semantics

For OperationalActivities, the OperationalActvityRealization’s owned behavior is expressed in terms of SystemFunctions.

For SystemFunctions, the OperationalCapabilityRealization is expressed in terms of OperationalActivities.

The result is a one to many mapping from an OperationalActivity to SystemFunctons that realize it, or from a SystemFunction to the OperationalActivities that it implements.

The activity owned by the realization can be decomposed into sub activities showing the desired detail.

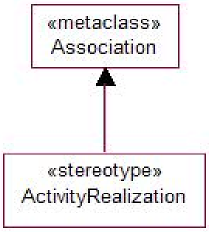


Figure 8.39 - ActivityRealization

8.4.2.8 Notation

No additional notational requirements.

8.4.3 Asset

8.4.3.1 Extension

- Class (from UML2)

8.4.3.2 Generalizations

8.4.3.3 Description

An Asset is a generic and very high-level description of an entity that appears in an OV-1 diagram, and can be used as an early representation of what will later become OperationalNodes, Systems, SystemNodes, OrganizationalResources, etc. The mappings from one of these constructs to an Asset can be modeled through an AssetMapping.

Assets are typically related to OperationalMissions and OperationalObjectives.

Asset is not intended to be used outside an OV-1.

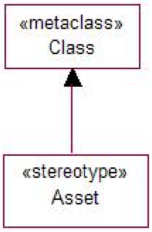


Figure 8.40 - Asset

8.4.3.4 Attributes

No additional attributes.

8.4.3.5 Associations

No additional associations.

8.4.3.6 Constraints

No additional constraints.

8.4.3.7 Semantics

No additional semantic requirements.

8.4.3.8 Notation

No additional notational requirements.

8.4.4 AssetMapping

8.4.4.1 Extension

- Dependency (from UML2)

8.4.4.2 Generalizations

8.4.4.3 Description

An AssetMapping is a relationship that indicates how an Asset has been realized, for example in terms of an OperationalNode, Service, or System.

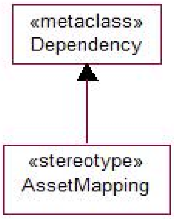


Figure 8.41 - AssetMapping

8.4.4.4 Attributes

No additional attributes.

8.4.4.5 Associations

No additional associations.

8.4.4.6 Constraints

[1] Asserts that the supplier of an AssetMapping Dependency is an Asset.

```
self.supplier->forAll(getAppliedStereotype('UPDM::Asset')->notEmpty())
```

8.4.4.7 Semantics

No additional semantic requirements.

8.4.4.8 Notation

No additional notational requirements.

8.4.5 Competence

8.4.5.1 Extension

- Class (from UML2)

8.4.5.2 Generalizations

8.4.5.3 Description

Competence in a specific discipline that can be provided by the indicated Resources. OperationalCapabilityRoles require Competencies to meet their declared responsibilities and perform their OperationalTasks.

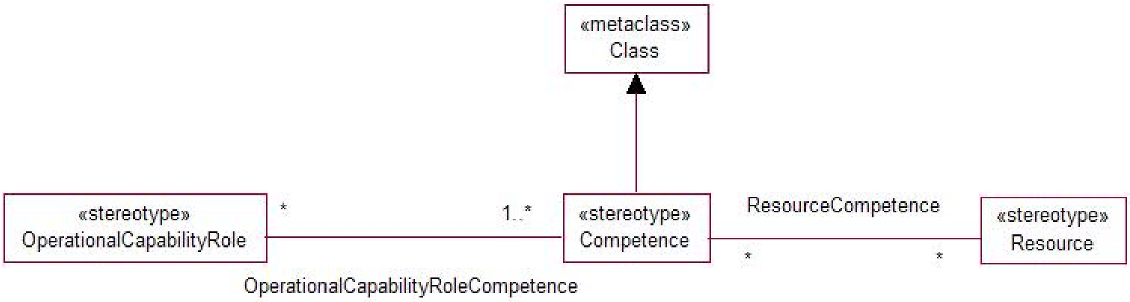


Figure 8.42 - Competence

8.4.5.4 Attributes

No additional attributes.

8.4.5.5 Associations

- operationalCapabilityRole : OperationalCapabilityRole [*]
  Roles that require this competence

- resource : Resource [*]
  Resources that can provide this Competence

8.4.5.6 Constraints

[1] Asserts that there is an association between a Competence and the OperationalCapabilityRole that requires it.

```
self.getAllAttributes()->asOrderedSet()->select(association-
   >notEmpty()).association->any

 (getAppliedStereotype('UPDM::OperationalCapabilityRoleCompetence')->
   notEmpty())->notEmpty()
```

[2] Asserts that a ResourceCompetence association exists between Resource and the Competence that it provides.

```
self.getAllAttributes()->asOrderedSet()->select(association-
   >notEmpty()).association->any

 (getAppliedStereotype('UPDM::ResourceCompetence')-> notEmpty())->notEmpty()
```

8.4.5.7 Semantics

No additional semantic requirements.

8.4.5.8 Notation

No additional notational requirements.

8.4.6 CompetenceRelationship

8.4.6.1 Extension

- Dependency (from UML2)

8.4.6.2 Generalizations

8.4.6.3 Description

A Resource or OrganizationalRole can have a set of related Competencies that are required or provided to meet their declared responsibilities and perform their OperationalTasks.

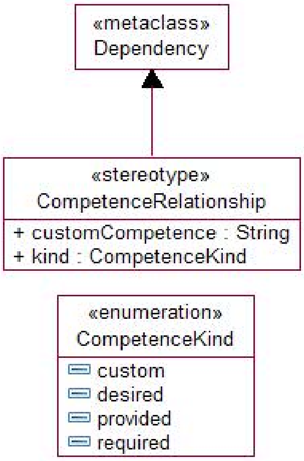


Figure 8.43 - CompetenceRelationship and CompetenceKind Enumeration

8.4.6.4 Attributes

- customCompetence : String [1]
  Indicates a custom meaning for the CompetenceRelationship when kind is set to custom.

- kind : CompetenceKind [1]
  The kind of competence relationships

8.4.6.5 Associations

No additional associations.

8.4.6.6 Constraints

[1] Asserts that the client of a CompetenceRelationship dependency is a Resource, one of its specializations, an OrganizationalRole or OperationalCapabilityRole.

```
self.client-> forAll(getAppliedStereotype('UPDM::Resource')-> notEmpty() or

getAppliedStereotypes()->collect(allParents())-> any(qualifiedName =
'UPDM::Resource') ->notEmpty() or

getAppliedStereotype('UPDM::OrganizationalRole')-> notEmpty() or

getAppliedStereotype('UPDM::OperationalCapabilityRole')->notEmpty())
```

[2] Asserts that the supplier of a CompetenceRelationship dependency is a Competence

```
self.supplier->forAll(getAppliedStereotype('UPDM::Competence')->notEmpty())
```

8.4.6.7 Semantics

No additional semantic requirements.

8.4.6.8 Notation

No additional notational requirements.

8.4.7 InformationElement

8.4.7.1 Extension

8.4.7.2 Generalizations

- ConformingElement (from AllViews)

8.4.7.3 Description

The collection of InformationElements and their performance attributes such as timeliness, quality, and quantity values.

DoDAF defines InformationElement as "... a formalized representation of information subject to an operational process (e.g., the information content that is required to be exchanged between nodes). In contrast an information exchange is comprised of the Needline, the information element, and other attributes such as Information Assurance attributes. The specific attributes of the information elements included are dependent on the objectives of the specific architecture effort and include the information scope, accuracy, and language. An information element may be used in one or more information exchanges."

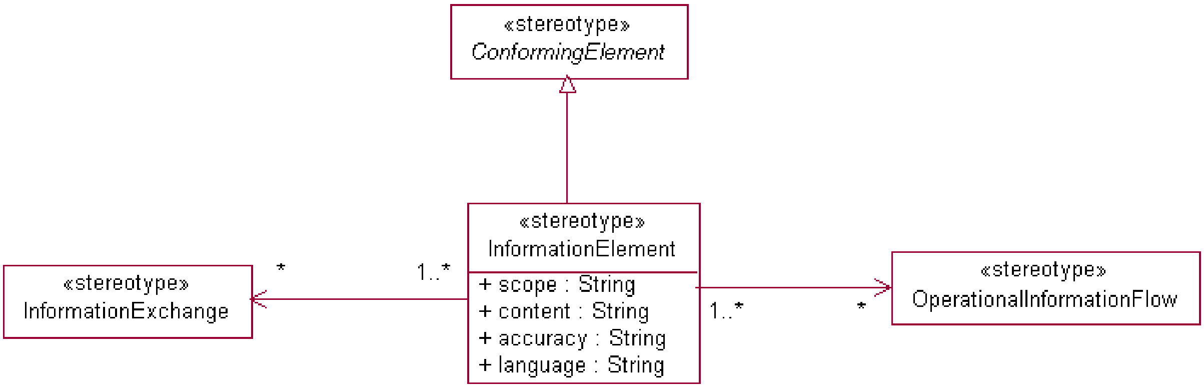


Figure 8.44 - InformationElement

8.4.7.4 Attributes

- accuracy : String [1]

  The required accuracy of the InformationElement.

- content : String [1]

  The content of the InformationElement.

- language : String [1]

  The language in which the information is expressed. For example, this could be a natural language, a computer language, mathematics, scientific, engineering or other discipline's nomenclature.

- scope : String [1]
  A description of the scope of the InformationElement.

8.4.7.5 Associations

- operationalInformationFlow : OperationalInformationFlow [*]
  The OperationalInformationFlows associated with this InformationElement.

- informationExchange : InformationExchange [*]
  The InformationExchanges that move these InformationElements

8.4.7.6 Constraints

[1] Asserts that this InformationElement is associated with an OperationalActivityFlow.

```
self.operationalInformationFlow.getAppliedStereotype
   ('UPDM::OperationalInformationFlow')->notEmpty()
```

[2] Asserts that there are zero or more InformationExchanges that utilize this InformationElement

```
self.informationExchange-> forAll(getAppliedStereotype
   ('UPDM::InformationExchange')->notEmpty())
```

8.4.7.7 Semantics

The identity of the individual InformationElements and their attributes must be documented in detail for many military missions. The InformationElements must be combined with InformationExchanges along with the InformationAssurance and Security definitions. InformationElements can also have relationships and constraints with other InformationElements outlined in a class diagram.

8.4.7.8 Notation

No additional notational requirements.

8.4.8 InformationExchange

8.4.8.1 Extension

- InformationFlow (from UML2)

8.4.8.2 Generalizations

8.4.8.3 Description

InformationExchange is an act of exchanging InformationElements between two distinct operational Nodes and the characteristics of the act, including the InformationElements that need to be exchanged and the attributes associated with the informationElement (e.g., Scope), as well as attributes associated with the exchange (e.g., Transaction Type). A Needline represents one or more InformationExchanges between the same two Nodes as supplier and consumer of the InformationExchange.

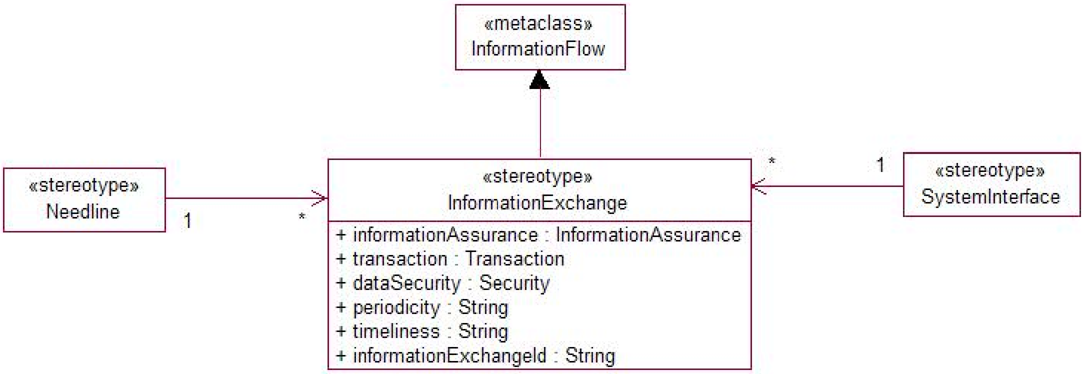


Figure 8.45 - InformationExchange

8.4.8.4 Attributes

- dataSecurity : Security [1]
  The Security elements that constrain this InformationExchange

- informationAssurance : InformationAssurance [1]
  The InformationAssurance elements that apply to this InformationExchange

- informationExchangeId : String [1]
  An identifier such as a unique id or serial number

- periodicity : String [1]
  A statement of the operational period of the InformationExchange

- timeliness : String [1]
  The schedule constraints for this InformationExchange

- transaction : Transaction [1]
  Specifies the Transaction if this InformationExchange is governed by transaction semantics

8.4.8.5 Associations

No additional associations.

8.4.8.6 Constraints

[1] Asserts that all the activity edges that realize this InformationFlow are either OperationalControlFlows or OperationalInformationFlows are owned by an OperationalActivity

```
self.realizingActivityEdge-> forAll((getAppliedStereotype
    ('UPDM::OperationalControlFlow')->notEmpty() or

getAppliedStereotype('UPDM::OperationalInformationFlow')->notEmpty()) and

owner.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty())
```

[2] Asserts that this InformationAssurance applies to the InformationExchange.

```
self.informationAssurance->forAll(classifier->forAll
   (name='InformationAssurance')) or
```

```
self.informationAssurance->forAll(classifier->forAll(allParents()->notEmpty()
   and allParents()->exists(name='InformationAssurance')))
```

[3] Asserts that the stereotype of transaction is Transaction

```
self.transaction->forAll(classifier->forAll(name='Transaction')) or
```

```
self.transaction->forAll(classifier->forAll(allParents()->notEmpty() and
   allParents()->exists(name='Transaction')))
```

[4] Asserts that the stereotype of realizingConnector or its type's stereotype is Needline

```
self.realizingConnector->notEmpty() and
```

```
self.realizingConnector->forAll(getAppliedStereotype('UPDM::Needline')-
   >notEmpty() or

   type.getAppliedStereotype('UPDM::Needline')->notEmpty())
```

[5] Asserts that the stereotype of dataSecurity is Security

```
self.dataSecurity->forAll(classifier->forAll(name='Security')) or
```

```
self.dataSecurity->forAll(classifier->forAll(allParents()->notEmpty() and
   allParents()->exists(name='Security')))
```

[6] Asserts that the Interaction that contains an InformationExchange must be an OperationalEventTrace

```
self.realizingMessage.getAppliedStereotype('UPDM::TaskInvocation')->notEmpty()
```

[7] Asserts that this InformationExchange occurs between two OperationalNodes.

```
(self.target->any(getAppliedStereotype( 'UPDM::OperationalNode')->notEmpty()) -
   >notEmpty() or
```

```
self.target->forAll(getAppliedStereotypes()-> collect(allParents())->
   any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty() ))
```

```
and
```

```
(self.source->any(getAppliedStereotype( 'UPDM::OperationalNode') ->notEmpty()
   )-> notEmpty() or self.source->forAll(getAppliedStereotypes()->
   collect(allParents())->any(qualifiedName = 'UPDM::OperationalNode') ->
   notEmpty() ))
```

[8] Asserts that the value provided by an OperationalActivity invoked by an InformationExchange is an InformationElement.

```
self.conveyed.oclAsType(uml::Classifier).getAppliedStereotype
   ('UPDM::InformationElement')->notEmpty()
```

8.4.8.7 Semantics

A set of InformationExchanges flowing from a providing OperationalNode to a consuming OperationalNode implies that a Needline exists between those two nodes with a directionality implied by the direction of the flow.

In the case where the InformationExchange is realized by a message, the signature of the message is an OperationalTask and the arguments of the message must correspond to the conveyed information element of the InformationExchange. The Behavior of the OperationalTask is the receiving OperationalActivity.

When a CompositeStructureDiagram is used to describe the structure of the encompassing OperationalNode, the connector between the parts that correspond to the lifelines can be assigned by the connector property.

If the InformationExchange ends are both OccurrenceSpecifications, then the connector must go between the Parts represented by the Lifelines of the two InformationExchange ends.

In the case where the InformationExchange is realized by Activity edges, the Activity edges will be stereotyped either OperationalInformationFlow or OperationalControlFlow.

OperationalInformationFlow

In this case, the ends of the flow are object pins whose type is stereotyped InformationElement. The activity elements connected by the OperationalInformationFlow must be either a CallBehaviorAction whose behavior is an OperationalActivity, or a CallOperationAction whose operation is an OperationalTask.

OperationalControlFlow

The source of the OperationalControlFlow is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an OperationalTask, or an InitialNode.

The target of the OperationalControlFlow is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an ActivityTask, or a FinalNode.

8.4.8.8 Notation

No additional notational requirements.

8.4.9 Materiel

8.4.9.1 Extension

- Class (from UML2)

8.4.9.2 Generalizations

8.4.9.3 Description

Materiel is tied to elements, where mechanisms may be used to represent Systems that support OperationalActivities. In addition, further materiel detail may be related to the activities, by relating those activities to the SystemFunctions that are executed by Systems that automate them (wholly or partially). An OperationalCapabilityRealization is defined in terms of the OperationalActivities. Materiel may be associated with UPDM behaviors: an OperationalActivity, a SystemFunction, an EventTrace, and StateTrace; and with other UPDM elements: an OperationalCapabilityRealization, an OperationalCapability, an OperationalNode, an OperationalNodeSpecification, a System, a SystemInterface, or a CapabilityConfiguration.

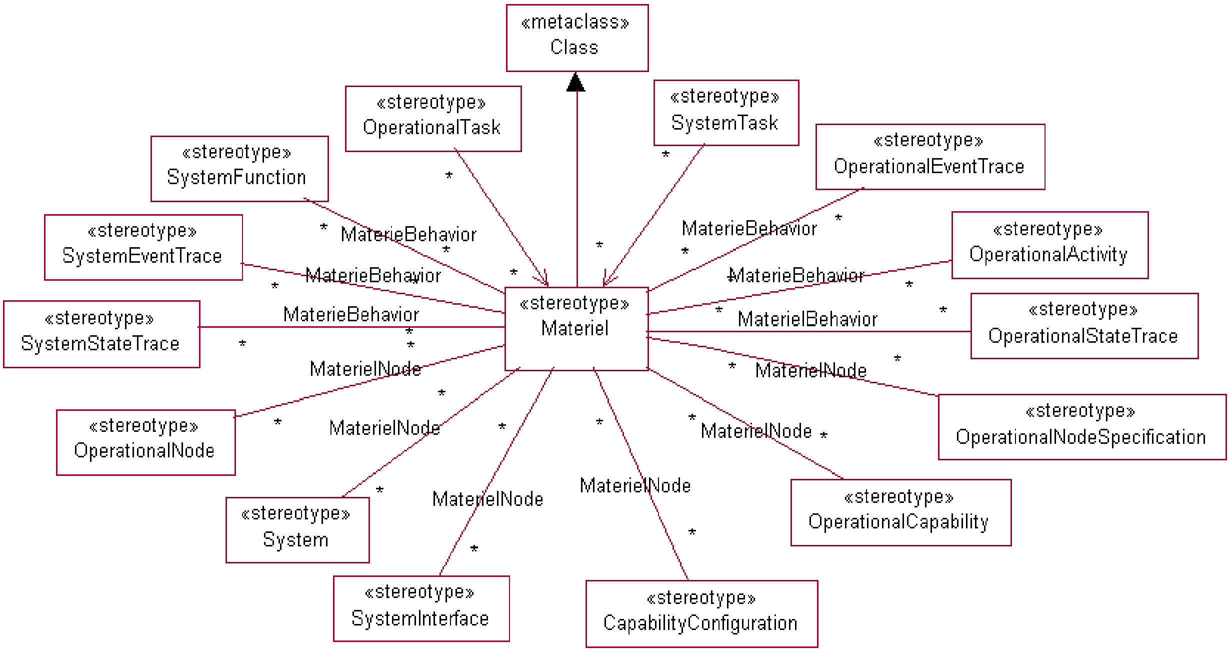

«metaclass» Class
«stereotype» OperationalTask
«stereotype» SystemTask
«stereotype» SystemFunction
«stereotype» OperationalEventTrace
«stereotype» SystemEventTrace
«stereotype» OperationalActivity
«stereotype» SystemStateTrace
«stereotype» Materiel
«stereotype» OperationalStateTrace
«stereotype» OperationalNode
«stereotype» OperationalNodeSpecification
«stereotype» System
«stereotype» OperationalCapability
«stereotype» SystemInterface
«stereotype» CapabilityConfiguration
MaterieBehavior
MaterieBehavior
MaterieBehavior
MaterieBehavior
MaterieBehavior
MaterielBehavior
MaterielNode
MaterielNode
MaterielNode
MaterielNode
MaterielNode
MaterielNode
*

Figure 8.46 - Materiel

8.4.9.4 Attributes

No additional attributes.

8.4.9.5 Associations

- capabilityconfiguration : CapabilityConfiguration [*]
- operationalactivity : OperationalActivity [*]
  Materiel used by this OperationalActivity
- operationalcapability : OperationalCapability [*]
  Materiel used by this OperationalCapability
- operationaleventtrace : OperationalEventTrace [*]
  Materiel used by this OperationalEventTrace
- operationalnode : OperationalNode [*]
  Materiel used by this OperationalNode
- operationalnodespecification : OperationalNodeSpecification [*]
  Materiel used by this OperationalNodeSpecification
- operationalstatetrace : OperationalStateTrace [*]
  Materiel used by this OperationalStateTrace
- system : System [*]
  Materiel used by this System

- systemeventtrace : SystemEventTrace [*]
  Materiel used by this OperationalEventTrace

- systemfunction : SystemFunction [*]
  Materiel used by this SystemFunction

- systeminterface : SystemInterface [*]
  Materiel used by this SystemInterface

- systemstatetrace : SystemStateTrace [*]
  Materiel used by this SystemStateTrace

8.4.9.6 Constraints

No additional constraints.

8.4.9.7 Semantics

No additional semantic requirements.

8.4.9.8 Notation

No additional notational requirements.

8.4.10 MaterielBehavior

8.4.10.1 Extension

- Association (from UML2)

8.4.10.2 Generalizations

8.4.10.3 Description

If this association exists, then one end is a Materiel and the other end is a Behavior stereotyped as either an OperationalActivity, a SystemFunction, an EventTrace, or StateTrace.

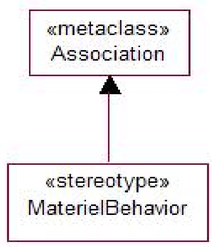


Figure 8.47 - MaterielBehavior

8.4.10.4 Attributes

No additional attributes.

8.4.10.5 Associations

No additional associations.

8.4.10.6 Constraints

[1] Materiel can be associated with the following UPDM Behavior elements stereotyped as either an OperationalActivity, OperationalEventTrace, OperationalStateTrace, SystemFunction, SystemEventTrace, SystemStateTrace

```
let e1:uml::Class = self.endType->at(1).oclAsType(uml::Class) in

let e2:uml::Class =  self.endType->at(2).oclAsType(uml::Class) in

if e2.oclIsKindOf(uml::Behavior)and e1.oclIsKindOf(uml::Class)

then

   e1.getAppliedStereotype('UPDM::Materiel')->notEmpty() and

   (e2.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty() or

   e2.getAppliedStereotype('UPDM::OperationalEventTrace')->notEmpty() or

   e2.getAppliedStereotype('UPDM::OperationalStateTrace')->notEmpty()or

   e2.getAppliedStereotype('UPDM::SystemFunction')->notEmpty() or

   e2.getAppliedStereotype('UPDM::SystemEventTrace')->notEmpty() or

   e2.getAppliedStereotype('UPDM::SystemStateTrace')->notEmpty())

else

if e1.oclIsKindOf(uml::Behavior)and e2.oclIsKindOf(uml::Class) then

   e2.getAppliedStereotype('UPDM::Materiel')->notEmpty() and

   (e1.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty() or

   e1.getAppliedStereotype('UPDM::OperationalEventTrace')->notEmpty() or

   e1.getAppliedStereotype('UPDM::OperationalStateTrace')->notEmpty()or

   e1.getAppliedStereotype('UPDM::SystemFunction')->notEmpty() or

   e1.getAppliedStereotype('UPDM::SystemEventTrace')->notEmpty() or

   e1.getAppliedStereotype('UPDM::SystemStateTrace')->notEmpty())

else false

endif

endif
```

8.4.10.7 Semantics

No additional semantic requirements.

8.4.10.8 Notation

No additional notational requirements.

8.4.11 MaterielNode

8.4.11.1 Extension

- Association (from UML2)

8.4.11.2 Generalizations

8.4.11.3 Description

Materiel can be associated with the following UPDM elements stereotyped as either an OperationalCapabilityRealization, an OperationalCapability, an OperationalNode, an OperationalNodeSpecification, a System, a SystemInterface, OperationalActivityRealization, or a CapabilityConfiguration.

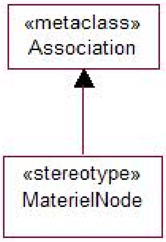


Figure 8.48 - MaterielNode

8.4.11.4 Attributes

No additional attributes.

8.4.11.5 Associations

No additional associations.

8.4.11.6 Constraints

[1] If this association exists, then one end is a Materiel and the other end is an OperationalCapabilityRealization, an OperationalCapability, an OperationalNode, anOperationalNodeSpecification, a System, a SystemInterface, or a CapabilityConfiguration.

```
let e1:uml::Class = self.endType-> at(1).oclAsType(uml::Class) in

let e2:uml::Class =  self.endType-> at(2).oclAsType(uml::Class) in

if e1.oclIsKindOf(uml::Class)and e2.oclIsKindOf(uml::Class) then

e1.getAppliedStereotype('UPDM::Materiel')-> notEmpty() and

(e2.getAppliedStereotype('UPDM::OperationalCapability')->notEmpty() or
e2.getAppliedStereotype('UPDM::OperationalNode')-> notEmpty() or

e2.getAppliedStereotypes()-> collect(allParents())->any(qualifiedName =
    'UPDM::OperationalNode') ->notEmpty() or

e2.getAppliedStereotype('UPDM::OperationalNodeSpecification')->notEmpty() or

e2.getAppliedStereotype('UPDM::System')-> notEmpty() or

e2.getAppliedStereotypes()-> collect(allParents())->any(qualifiedName =
    'UPDM::System') ->notEmpty() or

e2.getAppliedStereotype('UPDM::SystemInterface')-> notEmpty() or

e2.getAppliedStereotype('UPDM::CapabilityConfiguration')->notEmpty() )

 or

e2.getAppliedStereotype('UPDM::Materiel')-> notEmpty() and

(e1.getAppliedStereotype('UPDM::OperationalCapability')->notEmpty() or

e1.getAppliedStereotype('UPDM::OperationalNode')-> notEmpty() or

e1.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
    'UPDM::OperationalNode') ->notEmpty()or

e1.getAppliedStereotype('UPDM::OperationalNodeSpecification')->notEmpty() or

e1.getAppliedStereotype('UPDM::System')-> notEmpty() or

e1.getAppliedStereotypes()-> collect(allParents())->any(qualifiedName =
    'UPDM::System') ->notEmpty() or

e1.getAppliedStereotype('UPDM::SystemInterface')-> notEmpty() or

e1.getAppliedStereotype('UPDM::CapabilityConfiguration')->notEmpty() )

else false

endif
```

8.4.11.7 Semantics

No additional semantic requirements.

8.4.11.8 Notation

No additional notational requirements.

8.4.12 Needline

8.4.12.1 Extension

- Association (from UML2)
- Connector (from UML2)

8.4.12.2 Generalizations

8.4.12.3 Description

A Needline is a requirement that is the logical expression of the need to transfer information from a supplying operational Node to a consuming operational Node.

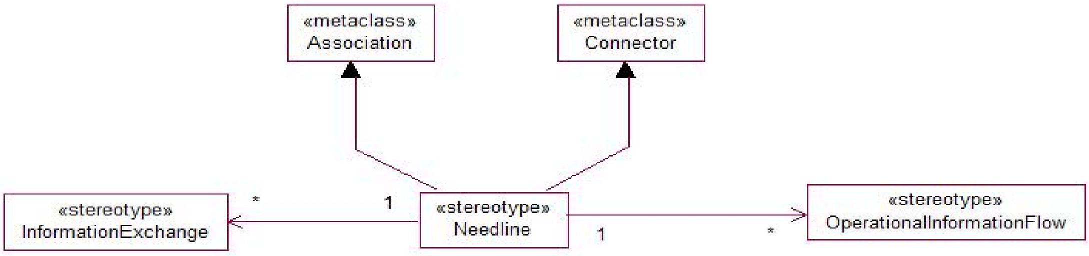


Figure 8.49 - Needline

8.4.12.4 Attributes

No additional attributes.

8.4.12.5 Associations

- informationExchanges : InformationExchange [*]
  The InformationExchanges that define this Needline.

- operationalInformationFlow : OperationalInformationFlow [*]
  The InformationFlows that are carried on this Needline

8.4.12.6 Constraints

[1] Asserts that this Needline is derived from one or more InformationExchanges

```
self.informationExchanges->notEmpty() and

self.informationExchanges->forAll(getAppliedStereotype
    ('UPDM::InformationExchange')->notEmpty())
```

[2] Asserts that both ends of the association must be stereotyped OperationalNode, one of its descendents or both ends must be sterotyped OperationalNodeSpecification

```
if self.oclIsTypeOf(uml::Association) then

let nl:uml::Association = self.oclAsType(uml::Association) in

(nl.endType->forAll(getAppliedStereotypes()-> collect(allParents())->
    any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty() or

getAppliedStereotype('UPDM::OperationalNode')-> notEmpty())) or
```

```
nl.endType-> forAll(getAppliedStereotype('UPDM::OperationalNodeSpecification')-
   > notEmpty())

else

 true

 endif
```

[3] Asserts that if the Needline is an Association then the multiplicity at both ends of the Needline is 1

```
if self.oclIsTypeOf(uml::Association) then

self.oclAsType(uml::Association).memberEnd.upper->forAll(a|a = 1) and
self.oclAsType(uml::Association).memberEnd.lower->forAll(a|a = 1)

else

true

endif
```

[4] Asserts that when Needline is modeled with a Connector, then both ends must connect OperationalNodePorts, or both ends must connect OperationalNodes or their specializations, or both ends must connect OperationalNodeSpecifications

```
if self.oclIsTypeOf(uml::Connector) then

   let con:uml::Connector = self.oclAsType(uml::Connector) in

con.end->forAll(role.oclIsTypeOf(uml::Port) and

role.getAppliedStereotype('UPDM::OperationalNodePort')->notEmpty()) or

con.end->forAll

(role.oclIsTypeOf(uml::Property) and

(role.type.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

role.type.getAppliedStereotypes()->collect(allParents())           ->
   any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty() )) or

con.end->forAll(role.oclIsTypeOf(uml::Property) and

role.type.getAppliedStereotype('UPDM::OperationalNodeSpecification')->
   notEmpty())

 else

 true

 endif
```

[5] Asserts that the elements in the operationalInformationFlow are stereotyped OperationalInformationFlow

```
self.operationalInformationFlow->
   forAll(getAppliedStereotype('UPDM::OperationalInformationFlow')->notEmpty())
```

8.4.12.7 Semantics

A Needline signifies a providing/consuming relationship between Nodes where the modification of the supplier may impact the client model elements.

Needlines and Information Exchanges

A Needline documents the requirement to exchange information between nodes. The Needline does not indicate how the information transfer is implemented. For example, if information produced at node A, is simply routed through node B, and is used at node C, then node B would not be shown on the OV-2 diagram - the Needline would go from Node A to Node C. OV-2 is not a communications link or communications network diagram. The System implementation (or what SystemsNodes or Systems are used to execute the transfer) is shown in the Systems Interface Description (SV-1). Furthermore, the Needline systems equivalent is the interface line depicted in SV-1. The actual implementation of an interface may take more than one form and is documented in a Systems Communications Description (SV-2). Therefore, a single Needline shown in the OV may translate into multiple interfaces in SV-1 and multiple physical links in SV-2.

Needlines may be generated automatically, derived from InformationExchanges, from ActivityEdges in OperationalActivities, and from Messages in OperationalEventTraces.

8.4.12.8 Notation

A Needline is represented by a line connecting the providing operational Node to the consuming operational Node with an arrowhead on the end of the line connected to and pointing at the consuming operational Node. Needlines may be drawn explicitly between two operational Nodes using an Association stereotyped as Needline.

When there are Needlines going in both directions between two OperationalNodes, they may be combined into a single bi-directional association.

In structure diagrams where two OperationalNodes are depicted with a Connector belonging to a message, the Connector should be typed with the appropriate Needline that represent the message in an OperationalEventTrace or control flow in an OperationalActivity.

8.4.13 OperationalActivity

8.4.13.1 Extension

- Activity (from UML2)

8.4.13.2 Generalizations

8.4.13.3 Description

The Operational Activity Model describes the operations that are normally conducted in the course of achieving a mission or a business goal. 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 Business Reference Model (DoDAF 1.5).

Guidance

There are two distinct sets of ActivityDiagrams: one to show the decomposition of the Activities independent of where they are performed and one to show the performing map. By factoring these two concerns, the Activity diagrams are easier to understand and model.

1. Define OperationalActivities either as owned by some OperationalNode or without a context. When OperationalNodes are defined, context free OperationalActivities may be assigned as owned by OperationalNodes.

OperationalNodes may perform any OperationalActivity. There are two approaches to model this:

- one is to show the Actions within partitions of Activity Diagrams where the partitions represent the OperationalNodes in which the OperationalActivity is performed.
- another is to type the lifelines of an Interaction with the OperationalNodes in which operations are performed whose Method is the desired Activity.

In the first case, the Activities performed by an OperationalNode cannot be determined by starting at the OperationalNode.

Two interesting characteristics regarding OperationalActivities are their composition, and who performs them.

2. To model the composition aspect, use CallBehaviorActions that invoke other OperationalActivities as a way to model the composition independent of who actually performs them. In this case, the partitions in the Activity diagrams would not necessarily represent OperationalActivities.

3. Then use CallOperationActions that are included in partitions that represent OperationalNodes that indicate that the OperationalNode is performing the Activity of which the called operation is the Method.

To indicate that an OperationalNode performs an OperationalActivity, include an OperationalTask - an operation - that results in the execution of the desired OperationalActivity.

- If the OperationalNode is the Owner of the OperationalActivity, then the OperationalTask's Method is set to the OperationalActivity, and the Specification of the OperationalActivity is set to the OperationalTask. (Actually this is a bidirectional association, and the tooling probably does the other half of this operation whichever is done first.)
- If the OperationalActivity is NOT owned by the OperationalNode, then a Proxy OperationalActivity is created that IS owned by the OperationalNode and whose Specification is set to the OperationalTask. This Proxy OperationalActivity includes a CallBehaviorAction to the desired OperationalActivity. In this case, the ultimate OperationalActivity may be owned by another OperationalNode, or it may be context free.

The reason is that the Specification of an OperationalActivity can be set to at most one operation, so to have multiple OperationalNodes performing an OperationalActivity, it is necessary to have each operation in the performing OperationalNode have its own Activity that can then use CallBehaviorAction to invoke the OperationalActivity to be performed.

Another feature of the Proxy OperationalActivity is that it can contain details that are specific to the performing OperationalNode, so it could enable a hierarchy of OperationalActivities by including specialized actions that are not part of a more general OperationalActivity.

Using this technique, the OperationalActivity structure can be modeled and the performing OperationalNodes can be modeled in a consistent and comprehensive way.

4. The other major behavior model, the EventTrace can use these OperationalTasks to model the sequence of events using the same OperationalTasks that are used to indicate which OperationalNodes perform the OperationalActivities and they are thus linked together to model a consistent whole.

Declaring the OperationalTasks and their Methods as the OperationalActivities is the first part, Putting the CallOperationActions into the Activity diagrams in the partitions is the second step.

A query on the model for all OperationalTasks results in the mapping of which OperationalNodes perform which OperationalActivities.

5. The OperationalTasks also have specific characteristics required by DoDAF. An OperationalActivity is the description of an activity, but the operation is the execution thereof. That execution may have different constraints in one OperationalNode from those in another performing the same OperationalActivity. The OperationalActivity stereotype provides the ability to differentiate these characteristics on a per OperationalNode basis.

An Activity can be expressed in terms of other Activities (callBehaviorActions) or Actions (callOperationActions). It includes the flow of objects and control among the actions or activities.

Interactions are expressed in terms of events, send and receive, that can cause the execution of behaviors or operations in the receiving class. These events also carry objects. In UPDM, when both of these expressions of behavior are used, the callBehaviors and callOperationActions in the Activities should correspond to behaviors and operations in the Interactions.

Behavior can be expressed from a single Class point of view, or from the point of view of a collaboration of Classes.

In the first case, the Activity performed at a particular OperationalNode can be derived from the business functions and carried forward into the System where it can become a service offered by the system.

UPDM can express both of these points of view using.

Many different kinds of OperationalNodes can perform an OperationalActivity, but they might do it in very different ways with very different actors and in different sequences. If an Activity and an Interaction are different ways to express the way in which an OperationalCapability is realized, then each OperationalCapabilityRealization can be achieved by a collaboration of different kinds of nodes that have different operations, yet all achieving the same OperationalCapability.

OperationalActivity can be modeled with three patterns to show the OperationalNodes at which they are performed.

OwnedBehavior

The OperationalActivity is defined as owned behavior of an OperationalNode. The OperationalNode is the Context of the OperationalActivity. Optionally, an OperationalTask can be defined on the OperationalNode that has its Method property set to the OperationalActivity. This OperationalActivity can be invoked from another OperationalActivity using a callBehaviorAction or a callOperationAction. In either case, the Partition in which these calls exist must be assigned to the OperationalNode that owns the OperationalActivity.

ProxyActivity

A context-free OperationalActivity is defined with no specific context. An OperationalNode is defined with an owning an OperationalActivity. This OperationalActivity contains at least a CallBehaviorAction referring to the context-free OperationalActivity. Optionally, an OperationalTask can be defined on the OperationalNode that has its Method property set to the owned OperationalActivity. In this case, any one can create an OwnedBehavior that references the context-free OperationalActivity. An OperationalActivity uses the context-free OperationalActivity and specifies that it is performed at the OperationalNode with the ProxyActivity. It is referenced using either a CallBehaviorAction or a CallOperationAction if the OperationalTask has been defined.

OperationalTask

An OperationalActivity is defined in a standalone context (some Classifier as desired). An OperationalTask is defined on the context. An OperationalInterface is defined with the OperationalTask as an operation and the context indicates that it implements the OperationalInterface. When the OperationalActivity is used in another OperationalActivity, any OperationalNode that implements the OperationalInterface can be the representation of the Partition and the OperationalActivity is invoked with a CallOperationAction on the desired OperationalActivity.

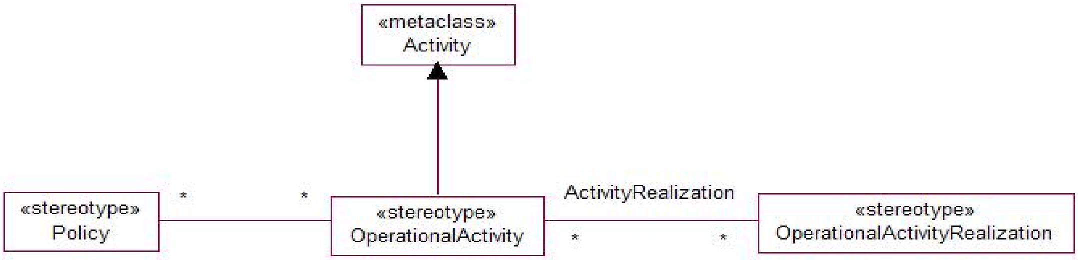


Figure 8.50 - OperationalActivity

8.4.13.4 Attributes

No additional attributes.

8.4.13.5 Associations

- materiel : Materiel [*]
  Materiel required for the OperationalActivity
- operationalActivityRealization : OperationalActivityRealization [*]
  See ActivityRealization for more details.
- policy : Policy [*]
  Policies that govern this activity

8.4.13.6 Constraints

[1] Asserts that an OperationalActivity that is invoked by a callBehaviorAction in a Partition of an OperationalActivity that represents an element, then that element must be stereotyped OperationalNodeSpecification or OperationalNode or one of its specializations or a Property that is typed by an OperationalNode or an OperationalNodeSpecification. The OperationalActivity specification must be an OperationalTask that is owned by the owner of the OperationalActivity.

If the partition represents an interface, then that interface must be implemented by the owner of the OperationalActivity and own an OperationalTask that has the same name as the OperationalActivity's specification.

If the partition represents a Property, then its type must obey the respective constraint.

Usage: This constraint assures the navigability from an OperationalNode to all of the OperationalActivities that it performs.

```
self.node->select(nde| nde.oclIsKindOf(uml::CallBehaviorAction) and
nde.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM::Operat
ionalActivity') -> notEmpty())->

forAll(op| op.inPartition.represents->forAll(x|not x.oclIsUndefined() ) and

(op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::OperationalNode')-> notEmpty() or
reps.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty()). oclAsType(uml::Class).ownedBehavior->

any(ownedop| ownedop=op.oclAsType(uml::CallBehaviorAction).behavior and
ownedop.specification->any(spec|

spec.oclIsTypeOf(uml::Operation) and

spec.getAppliedStereotype('UPDM::OperationalTask')-> notEmpty() and spec.owner =
ownedop.owner

      )->notEmpty()

)->notEmpty()

or

op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::OperationalNodeSpecification')->notEmpty() and

op.oclAsType(uml::CallBehaviorAction).behavior.owner.oclAsType(uml::Class).clientD
ependency.supplier->

any(x|x = reps)->notEmpty()). oclAsType(uml::Interface).ownedOperation->
any(ownedop|

ownedop.name = op.oclAsType(uml::CallBehaviorAction).behavior.specification.name
)-> notEmpty()

or

op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::OperationalNode')-
>notEmpty() or

reps.oclAsType(uml::Property).type.getAppliedStereotypes()->
collect(allParents())->any(qualifiedName = 'UPDM::OperationalNode') ->
notEmpty()). oclAsType(uml::Property).type.oclAsType(uml::Class).ownedBehavior->
any(ownedop| ownedop=op.oclAsType(uml::CallBehaviorAction).behavior and
ownedop.specification->any(spec|

    spec.oclIsTypeOf(uml::Operation) and

      spec.getAppliedStereotype('UPDM::OperationalTask')->notEmpty() and
       ownedop.owner

      )->notEmpty()
```

```
 )->notEmpty()

or

op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::OperationalNodeSpec
ification')->notEmpty() and
op.oclAsType(uml::CallBehaviorAction).behavior.owner.oclAsType(uml::Class).clientD
ependency.supplier->

any(x|x = reps)->
notEmpty()).oclAsType(uml::Property).type.oclAsType(uml::Interface).ownedOperation
->

any(ownedop|ownedop.name
=op.oclAsType(uml::CallBehaviorAction).behavior.specification.name)-> notEmpty()

))
```

[2] Asserts that for every callOperationAction in an OperationalActivity that refers to an Operation that is stereotyped OperationalTask, then:

- if the OperationalTask that is the operation of a callOperationAction that resides in a partition that represents an OperationalNode or a Property that is typed by an OperationalNode or one of its specializations, then
- the OperationalTask must be owned by that OperationalNode and the OperationalTask must have at least one method that specifies a corresponding OperationalActivity owned by the OperationalNode, (an OperationalTask could also specify Interactions or StateTraces)

OR

- if the OperationalTask that is the operation of a callOperationAction that resides in a partition that represents an OperationalNodeSpecification or a Property that is typed by an OperationalNodeSpecification, then
- the OperationalNodeSpecification must own the OperationalTask.

```
self.node->select(

   nde|nde.oclIsKindOf(uml::CallOperationAction) and
nde.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UPDM::Oper
ationalTask') ->notEmpty()

)->forAll(op|op.inPartition.represents->forAll(x|not x.oclIsUndefined()) and

(op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or
reps.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty()). oclAsType(uml::Class).ownedOperation->
any(ownedop| ownedop=op.oclAsType(uml::CallOperationAction).operation and
ownedop.method-> any(meth|meth.oclIsTypeOf(uml::Activity) and
meth.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty() and meth.owner
= ownedop.owner)-> notEmpty() )->notEmpty()

    or
op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::OperationalNodeSpecification')->
notEmpty()).oclAsType(uml::Interface).ownedOperation->
any(ownedop|ownedop=op.oclAsType(uml::CallOperationAction).operation)-> notEmpty()

      or op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::OperationalNode')-
>notEmpty() or reps.oclAsType(uml::Property).type.getAppliedStereotypes()->
collect(allParents())->any(qualifiedName = 'UPDM::OperationalNode') ->
notEmpty()).oclAsType(uml::Property).type.oclAsType(uml::Class).ownedOperation-
>any(ownedop| ownedop=op.oclAsType(uml::CallOperationAction).operation and
ownedop.method->any(meth|meth.oclIsTypeOf(uml::Activity) and
meth.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty() and meth.owner
= ownedop.owner

 )->notEmpty())->notEmpty()

or

op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::OperationalNodeSpec
ification')->notEmpty()

).oclAsType(uml::Property).type.oclAsType(uml::Interface).ownedOperation->

any(ownedop|ownedop=op.oclAsType(uml::CallOperationAction).operation)-> notEmpty()

     ))
```

[3] Asserts that the entries in policy are typed Policy

```
self.policy->forAll(getAppliedStereotype('UPDM::Policy')->notEmpty())
```

[4] Asserts that an OperationalActivity must be owned by an OperationalNode, one of its specializations or an OperationalCapabilityRealization.

```
let x:uml::Class = self.owner.oclAsType(uml::Class) in

x.getAppliedStereotype('UPDM::OperationalCapabilityRealization') ->notEmpty() or

x.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

x.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty()
```

8.4.13.7 Semantics

An OperationalActivity is the specification of parameterized behavior as the coordinated sequencing of activities whose individual elements are actions. An OperationalThread is the UML Element represented with an Activity Diagram, the OV-5

Constraints on OperationalActivity

OwnedBehavior

If an OperationalActivity includes a callBehaviorAction that refers to an Activity that is stereotyped OperationalActivity and the callBehaviorAction is included in a Partition and the Partition::Represents is not null, then the element that the Partition::Represents must be stereotyped OperationalNode and the OperationalNode::OwnedBehavior must include the OperationalActivity referenced by the callBehaviorAction.

OperationalTask

If an OperationalActivity includes a callOperationAction that refers to an Operation that is stereotyped OperationalTask and the callOperationAction is included in a Partition, then the Partition::Represents cannot be null and the element that the Partition::Represents must be stereotyped OperationalNode and the OperationalNode::OwnedMember must include the OperationalTask referenced by the callOperationAction. The constraints will validate these modeling techniques:

An OperationalActivity that uses callBehaviorActions initially to invoke OperationalActivities that are not allocated to a particular OperationalNode, and later Partitions are created in the OperationalActivity and their Representations set, then the Representations must be OperationalNodes, and the callBehaviorActions must refer to OperationalActivities owned by the OperationalNode Represented by the Partition. Otherwise, validation will indicate that there is a problem.

To model an OperationalActivity that is not owned by a particular OperationalNode (either context-free or owned by another element) that is to be performed by an OperationalNode using either an Activity or an Interaction (Sequence Diagram) or both:

The OperationalNode must implement an owned OperationalActivity (“Proxy”) that has a callBehaviorAction referring to that OperationalActivity that is either context-free or owned by another element that is to be performed by the OperationalNode. The OperationalNode must also implement an OperationalTask whose Method is set to the Proxy OperationalActivity.

Then to model the OperationalNode performing the OperationalActivity, either context-free or owned by another element, create an Activity with a Partition that Represents the OperationalNode and include a callOperationAction referencing the OperationalTask whose Method is set to the internal Proxy OperationalActivity.

And/Or, create an Interaction that has a Lifeline typed with the OperationalNode, and send it a Message whose signature is the OperationalTask.

The OperationalActivities Performed at an OperationalNode are those that are set as Methods of OperationalTasks.

8.4.13.8 Notation

No additional notational requirements.

8.4.14 OperationalCapability

8.4.14.1 Extension

- UseCase (from UML2)

8.4.14.2 Generalizations

8.4.14.3 Description

An OperationalCapability is a Use Case that specifies the requirements for a Capability.

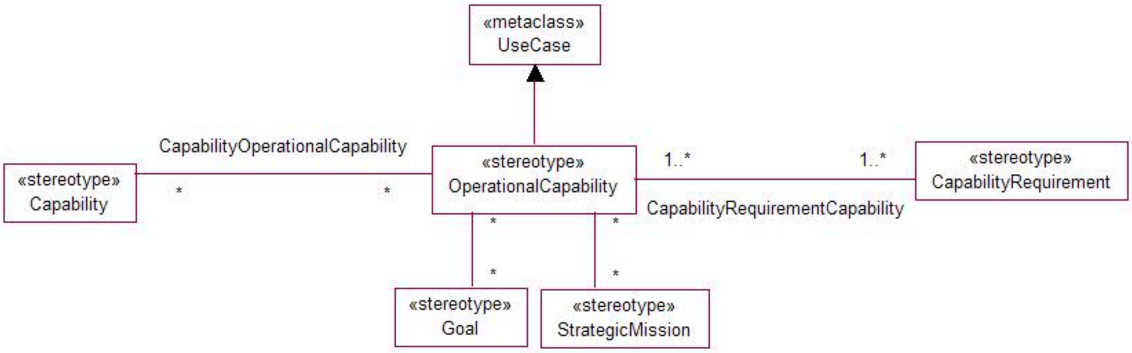


Figure 8.51 - OperationalCapability

8.4.14.4 Attributes

No additional attributes.

8.4.14.5 Associations

- capability : Capability [*]
  See CapabilityOperationalCapability for more details.
- capabilityRequirement : CapabilityRequirement [1..*]
  See CapabilityRequirementCapability for more details.
- goals : Goal [*]
  The Goals that are to be realized by the Strategic Mission and this capability
- materiel : Materiel [*]
  Materiel used in this OperationalCapability
- strategicMission : StrategicMission [*]
  The Strategic Missions supported by this OperationalCapability

8.4.14.6 Constraints

[1] Asserts that an OperationalCapability is realized by at least one Capability.

```
self.subject-> exists(getAppliedStereotype('UPDM::Capability')-> notEmpty() )
```

[2] Asserts that the OperationalCapability is related to a CapabilityRequirement

```
self.subject->exists (getAppliedStereotype('UPDM::CapabilityRequirement')->
notEmpty() )
```

[3] Asserts that the goals attribute has at least one entry.

```
self.goals.getAppliedStereotype('UPDM::Goal')->notEmpty()
```

[4] Asserts that the strategicMission attribute has at least one entry.

```
self.strategicMission.getAppliedStereotype('UPDM::StrategicMission')->notEmpty()
```

[5] Asserts that an OperationalCapability is realized by at least one OperationalCapabilityRealization.

```
self.subject->exists (getAppliedStereotype
('UPDM::OperationalCapabilityRealization')-> notEmpty() )
```

8.4.14.7 Semantics

OV-5 is a key product for describing capabilities and relating capabilities to mission accomplishment. The DoD Dictionary of Military Terms [DoD JP 1-02, 2001] defines a capability as “the ability to execute a specified course of action. (A capability may or may not be accompanied by an intention.)” A capability can be defined by one or more sequences of activities, referred to as operational threads or scenarios. A capability may be further described in terms of the attributes required to accomplish the set of activities (such as the sequence and timing of operational activities or materiel that enable the capability), in order to achieve a given mission objective. Capability-related attributes may be associated with specific activities or with the information flow between activities, or both. When represented by a set of operational activities, a capability can also be linked to an operational node in an OV-2.

In UPDM, the Capability activities and interactions are detailed in the OperationalCapabilityRealization.

8.4.14.8 Notation

No additional notational requirements.

8.4.15 OperationalCapabilityRealization

8.4.15.1 Extension

8.4.15.2 Generalizations

- OperationalNode (from OperationalView)

8.4.15.3 Description

OperationalCapabilityRealization is a collaboration of OperationalNodes that realize the OperationalCapability. It is the embodiment of the behavior of the OperationalCapability.

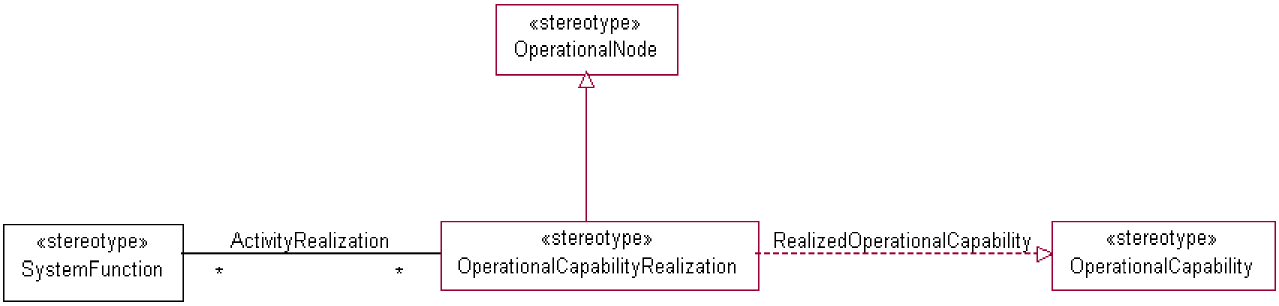


Figure 8.52 - OperationalCapabilityRealization

8.4.15.4 Attributes

No additional attributes.

8.4.15.5 Associations

- systemfunction : SystemFunction [*]
  SystemFunctions that realize the behavior of this OperationalCapabilityRealization

8.4.15.6 Constraints

[1] Asserts that OperationalCapabilityRealization is defined to realize an OperationalCapability.

```
self.clientDependency->asOrderedSet()->select(oclIsKindOf(uml::Realization))->
exists(getAppliedStereotype('UPDM::RealizedOperationalCapability')->notEmpty())
```

[2] Asserts that the use case for this realization is an OperationalCapability

```
self.useCase->exists(getAppliedStereotype('UPDM::OperationalCapability')-
>notEmpty())
```

8.4.15.7 Semantics

An OperationalCapability is realized by a collaboration among OperationalNodes. Each such collaboration is modeled by creating an OperationalCapabilityRealization. The behavior of an OperationalCapabilityRealization can be modeled as an OperationalActivity and a companion OperationalEventTrace.

The OperationalActivities have partitions that represent the collaborators. These collaborators may be any of the OperationalNodes or OperationalNodeSpecifications. (See OperationalActivity).

The Interaction and its Sequence diagram (OperationalEventTrace) specifies the interactions among the collaborators in terms of the OperationalTasks defined on the OperationalNodes(Specifications). These OperationalTasks are either the target of callOperationalActions or the specification of the activities invoked by callBehaviorActions that occur in the OperationalActivities.

8.4.15.8 Notation

No additional notational requirements.

8.4.16 OperationalCapabilityRole

8.4.16.1 Extension

8.4.16.2 Generalizations

- OperationalNode (from OperationalView)

8.4.16.3 Description

An OperationalCapabilityRole defines an association between the concept represented by an OperationalNode and the OrganizationalResource (OperationalActors) that actually perform the operations of the OperationalNodes to accomplish the node's capabilities. An OperationalCapabilityRole's operations are the OperationalTasks that are performed in that role.

An OperationalNode is an abstract representation of capabilities that are achieved through the OperationalActivities. The OperationalCapabilityRole is an explicit representation of that Role that enables modeling the assignment of the OrganizationalResources who will play that role. These OperationalCapabilityRole define a role in an Organization which is fulfilled by a post or by a sub-ordinate organization. The role may or may not be fulfilled (i.e., roleProvider is null) - e.g., it may be a vacant post

The term OperationalRole is used in DoDAF 1.5 to indicate Service provider and consumer in a net-centric architecture.

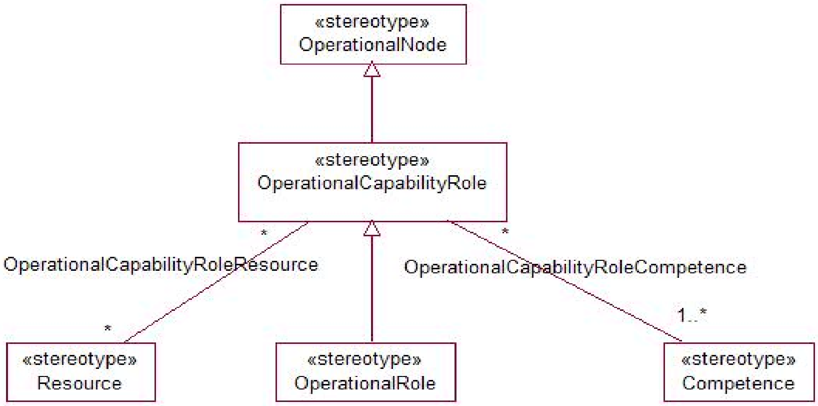


Figure 8.53 - OperationalCapabilityRole

8.4.16.4 Attributes

No additional attributes.

8.4.16.5 Associations

- competence : Competence [1..*]
  Competence (training, skill) required by this role to complete its tasks.
- resource : Resource [*]
  Resources that can play this role if they have the required competence

8.4.16.6 Constraints

[1] Asserts that there is an association between a Competence and the OperationalCapabilityRole that requires it.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

  (getAppliedStereotype('UPDM::OperationalCapabilityRoleCompetence')-> notEmpty())-
>notEmpty()
```

[2] Asserts that there is an association between an OperationalCapabilityRole and the Resource that provides it.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::OperationalCapabilityRoleResource')-> notEmpty())-
>notEmpty()
```

8.4.16.7 Semantics

No additional semantic requirements.

8.4.16.8 Notation

No additional notational requirements.

8.4.17 OperationalCapabilityRoleCompetence

8.4.17.1 Extension

- Association (from UML2)

8.4.17.2 Generalizations

8.4.17.3 Description

Specifies that an OperationalCapabilityRole requires zero or more Competencies and a Competence is required by zero or more OperationalCapabilityRole and that these are the elements associated with this association.

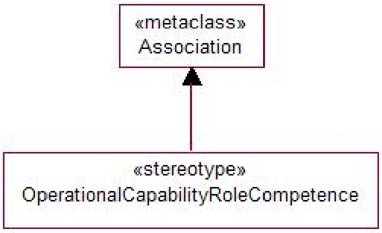


Figure 8.54 - OperationalCapabilityRoleCompetence

8.4.17.4 Attributes

No additional attributes.

8.4.17.5 Associations

No additional associations.

8.4.17.6 Constraints

[1] Asserts that Competencies are required by OperationalCapabilityRoles and that OperationalCapabilityRoles require Competence and that these are the elements related by this association.

```
(self.endType-> at(1).getAppliedStereotype('UPDM::Competence')-> notEmpty() and

(self.endType->at(2).getAppliedStereotype('UPDM::OperationalCapabilityRole')->
notEmpty()
or

self.endType->at(2).getAppliedStereotypes()-> collect(allParents())->
any(qualifiedName = 'UPDM::OperationalCapabilityRole') ->notEmpty() ))

or

(self.endType-> at(2).getAppliedStereotype('UPDM::Competence')-> notEmpty() and

(self.endType->at(1).getAppliedStereotype('UPDM::OperationalCapabilityRole')-
>notEmpty()

or

self.endType->at(1).getAppliedStereotypes()-> collect(allParents())->
any(qualifiedName = 'UPDM::OperationalCapabilityRole') ->notEmpty() ))
```

[2] Asserts that zero or more Competencies are required by zero or more OperationalCapabilityRoles.

```
self.memberEnd.upper->forAll(a|a = -1) and self.memberEnd.lower->forAll(a|a = 0)
```

8.4.17.7 Semantics

No additional semantic requirements.

8.4.17.8 Notation

No additional notational requirements.

8.4.18 OperationalCapabilityRoleResource

8.4.18.1 Extension

- Association (from UML2)

8.4.18.2 Generalizations

8.4.18.3 Description

An association showing the Resources that play the OperationalCapabilityRole.

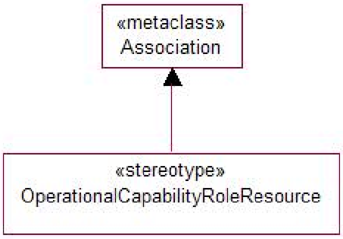


Figure 8.55 - OperationalCapabilityRoleResource

8.4.18.4 Attributes

No additional attributes.

8.4.18.5 Associations

No additional associations.

8.4.18.6 Constraints

[1] Asserts that zero or more Resources play zero or more OperationalCapabilityRoles and that zero or more OperationalCapabilityRoles are played by the Resource and that these are the elements associated by this association.

```
(self.base_Association.endType->at(1).getAppliedStereotype('UPDM::Resource')->
notEmpty()  or

self.base_Association.endType->at(1).getAppliedStereotypes()-
>collect(allParents())->any(qualifiedName = 'UPDM::Resource') ->notEmpty() and

(self.base_Association.endType-
>at(2).getAppliedStereotype('UPDM::OperationalCapabilityRole')->notEmpty() or

self.endType->at(2).getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::OperationalCapabilityRole') ->notEmpty() ))

or

(self.base_Association.endType->at(2).getAppliedStereotype('UPDM::Resource')->
notEmpty()or

self.base_Association.endType->at(1).getAppliedStereotypes()->
collect(allParents())->any(qualifiedName = 'UPDM::Resource') ->notEmpty() and

(self.base_Association.endType->
at(1).getAppliedStereotype('UPDM::OperationalCapabilityRole')->notEmpty() or

self.endType->at(1).getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::OperationalCapabilityRole') ->notEmpty() ))
```

[2] Asserts that the Resource provides the competency required by the OperationalCapabilityRole.

```
let role2:uml::Association = self.endType->
at(2).oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::OperationalCapabilityRoleCompetence')-> notEmpty()
or getAppliedStereotype('UPDM::ResourceCompetence')-> notEmpty()) in

letrole1:uml::Association = self.endType->
at(1).oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::ResourceCompetence')-> notEmpty()or
getAppliedStereotype('UPDM::OperationalCapabilityRoleCompetence')-> notEmpty() )in

 role1->notEmpty() and
```

```
role1.endType->asOrderedSet()->any(i| i=role2.endType->at(1) or
 i=role2.endType->at(2)
)->notEmpty()
```

[3] Asserts that a zero or more Resources are associated with zero or more OperationalCapabilityRoles.

```
self.memberEnd.upper->forAll(a|a = -1) and self.memberEnd.lower->forAll(a|a = 0)
```

8.4.18.7 Semantics

No additional semantic requirements.

8.4.18.8 Notation

No additional notational requirements.

8.4.19 OperationalControlFlow

8.4.19.1 Extension

- ControlFlow (from UML2)

8.4.19.2 Generalizations

8.4.19.3 Description

A flow of control of energy from one activity node to another.

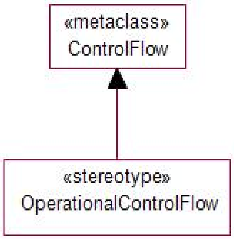


Figure 8.56 - OperationalControlFlow

8.4.19.4 Attributes

No additional attributes.

8.4.19.5 Associations

No additional associations.

8.4.19.6 Constraints

[1] Asserts that the source of the OperationalControlFlow is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an ActivityTask, or an InitialNode

```
if self.source.oclIsKindOf(uml::CallOperationAction) then
self.source.oclAsType(uml::CallOperationAction).operation.

getAppliedStereotype('UPDM::OperationalTask')->notEmpty()

else

if self.source.oclIsKindOf(uml::CallBehaviorAction) then


self.source.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM
::OperationalActivity')->notEmpty()

else

   self.source.oclIsKindOf(uml::InitialNode)

endif

endif
```

[2] Asserts that the target of the OperationalControlFlow is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an ActivityTask, or an FinalNode

```
if self.target.oclIsKindOf(uml::CallOperationAction) then


self.target.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UP
DM::OperationalTask')->notEmpty()

else

   if self.target.oclIsKindOf(uml::CallBehaviorAction) then
self.target.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM
::OperationalActivity')->notEmpty()

   else

      self.target.oclIsKindOf(uml::FinalNode)

   endif

endif
```

8.4.19.7 Semantics

OperationalControlFlows are the flow of control among activity nodes, usually represented on Activity Diagrams.

The flow of control from one action to another that specifies the actions within an OperationalActivity. The actions invoke either specific OperationalTasks or other OperationalActivities. An OperationalControlFlow has a source action (FROM), the source of the OperationalControlFlow that is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an OperationalTask, or an InitialNode. It has a target action (TO). the target of the OperationalControlFlow, that is one of a CallBehaviorAction whose behavior is an OperationalActivity. a CallOperationAction whose operation is an OperationalTask, or a FinalNode

A SystemControlFlow specifies partially or fully one InformationExchange.

8.4.19.8 Notation

No additional notational requirements.

8.4.20 OperationalEventTrace

8.4.20.1 Extension

- Interaction (from UML2)

8.4.20.2 Generalizations

8.4.20.3 Description

Interactions describe behavior as a sequence of events of the system's parts and other actors. The behavior of an interaction as depicted in a sequence diagram that shows the ordered exchange of information between OperationalNodes and other elements through the activation of their OperationalTasks that participate in the interaction. The OV-6c is a sequence diagram that shows the behavior of the OperationalCapabilityRealization's OperationalNodes.

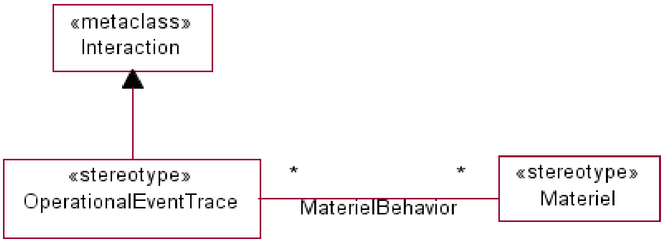


Figure 8.57 - OperationalEventTrace

8.4.20.4 Attributes

No additional attributes.

8.4.20.5 Associations

- materiel : Materiel [*] Materiel required for this Interaction

8.4.20.6 Constraints

[1] Asserts that the owner of the OperationalEventTrace is either an OperationalNode or an OperationalCapabilityRealization.

```
self.owner.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

self.owner.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty() or

self.owner.getAppliedStereotype('UPDM::OperationalCapabilityRealization') ->
notEmpty()
```

8.4.20.7 Semantics

No additional semantic requirements.

8.4.20.8 Notation

No additional notational requirements.

8.4.21 OperationalInformationFlow

8.4.21.1 Extension

- ObjectFlow (from UML2)

8.4.21.2 Generalizations

8.4.21.3 Description

OperationalInformationFlow is the flow of objects, InformationElements, within OperationalActivities. A flow of information, energy, or materiel from one activity node to another.

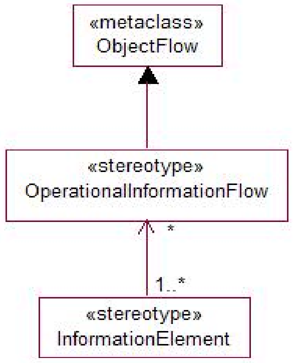


Figure 8.58 - OperationalInformationFlow

8.4.21.4 Attributes

No additional attributes.

8.4.21.5 Associations

No additional associations.

8.4.21.6 Constraints

[1] Asserts that if the source of the object flow is an outputpin, then if the owner of the output pin is a CallBehaviorAction then its behavior is an OperationalActivity, or a CallOperationAction then its operation is an OperationalTask.

```
if self.oclAsType(uml::ObjectFlow).source.oclIsKindOf(uml::OutputPin) then
```

```
   let src:uml::OutputPin = self.source.oclAsType(uml::OutputPin) in

   if (src.owner.oclIsKindOf(uml::CallOperationAction)) or
(src.owner.oclIsKindOf(uml::CallBehaviorAction)) then

   ((src.owner.oclIsKindOf(uml::CallOperationAction) and


src.owner.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UPDM
::OperationalTask')->notEmpty()) or

      (src.owner.oclIsKindOf(uml::CallBehaviorAction) and


src.owner.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM::
OperationalActivity')->notEmpty()))

   else true

   endif

else true

endif
```

[2] Asserts that if the target of the object flow is an Input Pin, then the owner of the Input pin must be an OperationalAction - a CallBehaviorAction whose behavior is an OperationalActivity, or a CallOperationAction whose operation is an OperationalTask.

```
if self.oclAsType(uml::ObjectFlow).target.oclIsKindOf(uml::InputPin) then

   (let trg:uml::InputPin = self.target.oclAsType(uml::InputPin) in

      (trg.owner.oclIsKindOf(uml::CallOperationAction) and


trg.owner.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UPDM
::OperationalTask')->notEmpty())

   or (trg.owner.oclIsKindOf(uml::CallBehaviorAction) and


trg.owner.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM::
OperationalActivity')->notEmpty())

   )

else

   true

endif
```

[3] The type of the input or output pin must be an InformationElement.

```
if self.target.oclIsKindOf(uml::InputPin) then
```

```
self.target.oclAsType(uml::InputPin).type.getAppliedStereotype('UPDM::InformationE
lement')->notEmpty()

else

if self.source.oclIsKindOf(uml::OutputPin) then


self.source.oclAsType(uml::OutputPin).type.getAppliedStereotype('UPDM::Information
Element')->notEmpty()

else

   false

endif

endif
```

8.4.21.7 Semantics

OperationalInformationFlows are the flow of information among activity nodes, usually represented on Activity Diagrams. In the case of flow of physical entities, they may include parameters or constraints representing their physical characteristics..

In an IT environment information flow objects' size is not a factor and transmission times are usually considered instantaneous unless the model is, for example, modeling real communications systems that must include factors such as latency, bandwidth, and packet size considerations.

In physical flows, the volume or size of the object is a consideration, and the time to traverse the Needline is not instantaneous. In this case, constraints can be applied that indicate the size of the objects (e.g., volume) and the rate of flow.

8.4.21.8 Notation

No additional notational requirements.

8.4.22 OperationalNode

8.4.22.1 Extension

- Class (from UML2)

8.4.22.2 Generalizations

8.4.22.3 Description

An OperationalNode performs a role or mission; a conceptual architectural element that produces, consumes, or processes information. An OperationalNode exists to realize one or more capabilities. Deployment of the capabilities in systems that help to realize those capabilities is allocated to SystemsNodes. An OperationalNode can be a composite structure that is decomposed into finer grained OperationalNodes. To model OperationalNode composition, designate component OperationalNodes as parts of the composite OperationalNode. This is a recursive structure. Level of decomposition is indicated in the decompositionLevel attribute. The level of decomposition, beginning with zero, the default, is recorded in the decompositionLevel attribute. The concept of an actor or resource that realizes an OperationalNode has been made explicit by using OrganizationalResource that plays an OperationalCapabilityRole in the context of an OperationalNode. The OperationalCapabilityRole is an Class that associates Competencies required for that OperationalCapabilityRole.

Information is consumed by an OperationalNode by ingesting information through required DataFlows, exchanged from another OperationalNode, provided DataFlows, that produces the information. If an OperationalNode consumes information it implies that it needs a channel for communication. These channels are called Needlines. They are directional indicating the supplier and consumer of the information exchange.

DoDAF (V1) states: “The relationship between architecture data elements across the System View to the Operational View (OV) can be exemplified as systems are procured and fielded to support organizations and their operations. SV-1 links together the OV and SV by depicting the assignments of systems and systems nodes (and their associated interfaces) to the OperationalNodes (and their associated Needlines) described in OV-2. OV-2 depicts the OperationalNodes representing organizations, organization types, and/or human roles, while SV-1 depicts the physical assets that house OperationalNodes (e.g., platforms, units, facilities, and locations) and the corresponding systems resident at these physical assets and which support the OperationalNodes.”

An OperationalNode is an abstract representation of capabilities that are achieved through the OperationalTasks (the operations on the OperationalNode). Interactions among OperationalNodes are represented as InformationExchanges between the OperationalNodes. These InformationExchanges are accomplished by the execution of an OperationalActivity in the receiving OperationalNode. In an interaction, an OperationalNode may perform one or more OperationalActivities. That group of OperationalTasks defines the role that the OperationalNode plays in the context of that interaction or in the accomplishment of that capability. (See OperationalCapabilityRole).

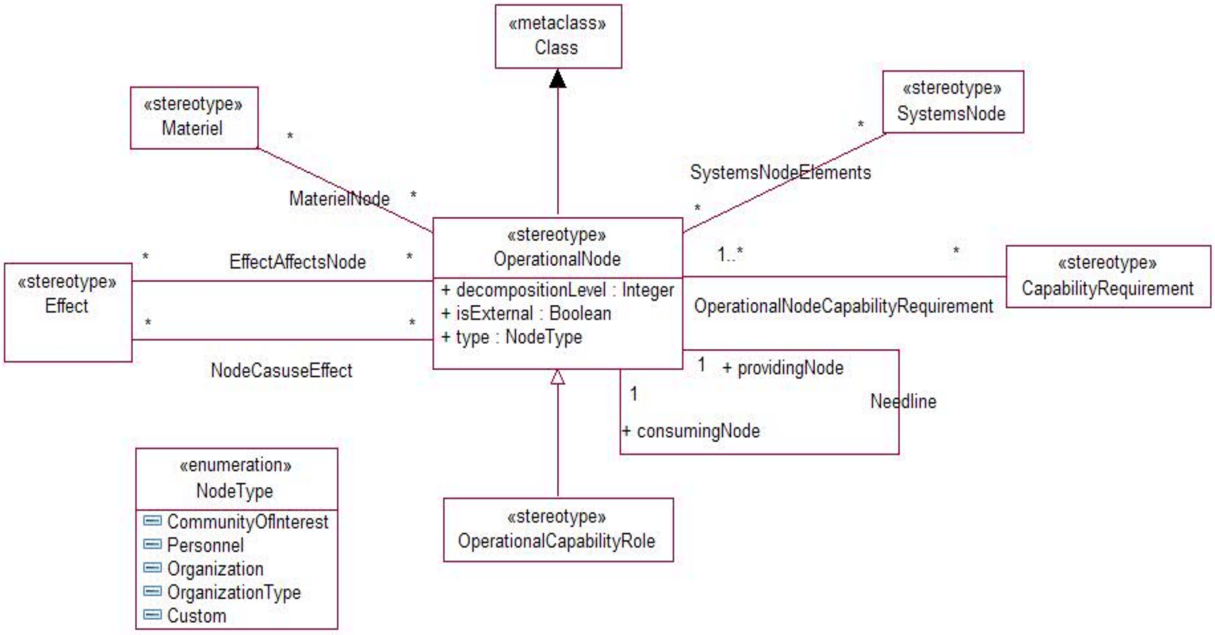


Figure 8.59 - OperationalNode

8.4.22.4 Attributes

- decompositionLevel : Integer [1]

OperationalNode is a composite element. Level of decomposition of this node. Default value = 0

- isExternal : Boolean [1]
  Is this OperationalNode external to the system being described? Default value = false
- type : NodeType [1]
  An indicator of a type of OperationalNode

8.4.22.5 Associations

- capabilityRequirement : CapabilityRequirement [*]
  See OperationalNodeCapabilityRequirement for more details.
- consumingNode : OperationalNode [1]
- effect : Effect [*]
  See NodeCausesEffect for more details.
- effect : Effect [*]
  See EffectAffectsNode for more details.
- materiel : Materiel [*]
  Materiel required for the OperationalNode
- providingNode : OperationalNode [1]
- systemsnode : SystemsNode [*]
  SystemsNode in which the OperationalNode is housed.

8.4.22.6 Constraints

[1] Asserts that there is at least one SystemsNode (a physical asset) that houses this OperationalNode.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

[2] Asserts that this Node participates in at least one Needline. (If no other Node needs this one or it doesn't need any other Node, then what is its purpose?)

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::Needline')-> notEmpty())->notEmpty()
```

[3] Asserts that an OperationalNode must have at least one element that defines its behavior. One of OperationalEventTrace, OperationalActivity or OperationalStateTrace

```
self.oclAsType(uml::Class).ownedBehavior->asOrderedSet()->exists(a|

a.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty() or

a.getAppliedStereotype('UPDM::OperationalEventTrace')->notEmpty()or

a.getAppliedStereotype('UPDM::OperationalStateTrace')->notEmpty())
```

[4] Asserts that this Node is required for delivery of an OperationalCapability

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::OperationalNodeCapability')-> notEmpty())-
>notEmpty()
```

[5] Asserts that there is at least one OperationalTask defined on an OperationalNode.

```
self.oclAsType(uml::Class).ownedOperation->asOrderedSet() ->

exists(getAppliedStereotype('UPDM::OperationalTask')->notEmpty())
```

[6] Asserts that an OperationalNode realizes interfaces that are OperationalNodeSpecifications, i.e., interfaces

```
self.clientDependency->forAll(spec|

if spec.oclIsKindOf(uml::InterfaceRealization)then

spec.getAppliedStereotype('UPDM::RealizedOperationalSpecification')->notEmpty()

else

true

endif)
```

8.4.22.7 Semantics

To specify the EventTrace of an OperationalNode, create an Interaction in the OperationalNode and stereotype it with EventTrace. Then create a Sequence diagram to populate it. (OV 6-c)

To specify the OperationalActivities of an OperationalNode, create an Activity in the OperationalNode and stereotype it with OperationalActivity. Then create an Activity diagram to populate it.

To determine the required DataFlows (Needlines) of an OperationalNode, refer to ClientDependencies that are stereotyped as Needlines.

Provided Dataflows are a stereotype property, a metaclass association with Needline.

Operations defined on an OperationalNode may be stereotyped as OperationalTask. OperationalTasks are defined as stereotyped operations on OperationalNodes.

8.4.22.8 Notation

No additional notational requirements.

8.4.23 OperationalNodePort

8.4.23.1 Extension

- Port (from UML2)

8.4.23.2 Generalizations

8.4.23.3 Description

An OperationalNodePort is used to model details of Node connections when a Needline is modeled as a connector.

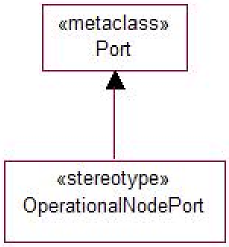


Figure 8.60 - OperationalNodePort

8.4.23.4 Attributes

No additional attributes.

8.4.23.5 Associations

No additional associations.

8.4.23.6 Constraints

[1] Asserts that the owner of the ends of an OperationalPort is a Needline

```
self.oclAsType(uml::Port).end-
>forAll(owner.getAppliedStereotype('UPDM::Needline')->notEmpty())
```

[2] Asserts that the owner of the OperationalPort is an OperationalNode or one of its specializations.

```
self.oclAsType(uml::Port)owner.getAppliedStereotype('UPDM::OperationalNode')->
notEmpty() or
```

```
self.oclAsType(uml::Port).owner.getAppliedStereotypes()->collect(allParents())-
>any(qualifiedName = 'UPDM::OperationalNode') -> notEmpty()
```

8.4.23.7 Semantics

No additional semantic requirements.

8.4.23.8 Notation

No additional notational requirements.

8.4.24 OperationalNodeSpecification

8.4.24.1 Extension

- Interface (from UML2)

8.4.24.2 Generalizations

8.4.24.3 Description

OperationalNodeSpecification provides a separation of the declaration of OperationalTasks from the OperationalNodes that implement them.

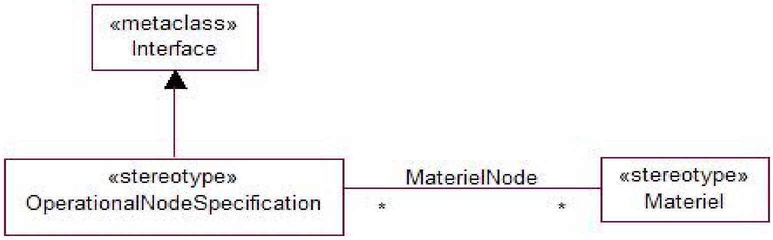


Figure 8.61 - OperationalNodeSpecification

8.4.24.4 Attributes

No additional attributes.

8.4.24.5 Associations

- materiel : Materiel [*]
  Materiel required for this OperationalNodeSpecification

8.4.24.6 Constraints

[1] Asserts that there is at least one OperationalTask defined on an OperationalNodeSpecification.

```
self.oclAsType(uml::Interface).ownedOperation->asOrderedSet() ->

exists(getAppliedStereotype('UPDM::OperationalTask')->notEmpty())
```

8.4.24.7 Semantics

No additional semantic requirements.

8.4.24.8 Notation

No additional notational requirements.

8.4.25 OperationalRole

8.4.25.1 Extension

8.4.25.2 Generalizations

- OperationalCapabilityRole (from OperationalView)

8.4.25.3 Description

The use of operational role applied to operational nodes stresses the responsibility the node plays in interacting with external organizations and human roles. There are three types of operational roles:

- The operational role of Service Provider helps to illustrate the set of known sources of information. Needlines associated with Nodes that have a Service Provider role indicates that information is provided by that source. The operational role of Service Provider may be applied to both the internal system or to external systems that provide needed information.
- The operational role of Service Consumer helps to illustrate the set of expected users of information. Needlines associated with Nodes that have a Service Consumer role indicates that information is required by that consumer. The operational role of Service Consumer may be applied to both the internal system or to external systems that require needed functionality.
- The operational role of Unanticipated User helps to identify the set of unknown external consumers who may be interested in obtaining information the internal system provides. The Unanticipated User operational role will transition to a service consumer role and allows the architecture to roughly categorize the community of unexpected users of services.
- The Needlines identify the desire of the service provider to support other targeted audiences.

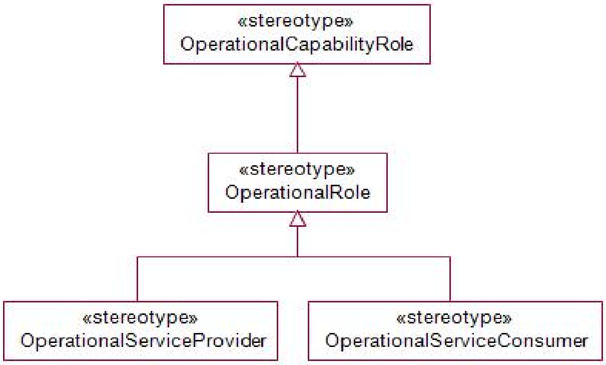

«stereotype»
OperationalCapabilityRole
«stereotype»
OperationalRole
«stereotype»
OperationalServiceProvider
«stereotype»
OperationalServiceConsumer

Figure 8.62 - OperationalRole

8.4.25.4 Attributes

No additional attributes.

8.4.25.5 Associations

No additional associations.

8.4.25.6 Constraints

No additional constraints.

8.4.25.7 Semantics

No additional semantic requirements.

8.4.25.8 Notation

No additional notational requirements.

8.4.26 OperationalServiceConsumer

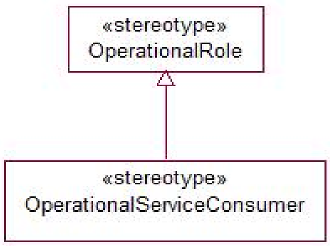
8.4.26.1 Extension

8.4.26.2 Generalizations

- OperationalRole (from OperationalView)

8.4.26.3 Description

Any classifier (class, component, and so on) may act as the consumer of a service, and that includes another service. While this stereotype is most definitely optional, it may help you identify elements of a model -- that are not services themselves -- as clients of services. On the other hand, it may just be overhead and you don't need to use it.



OperationalServiceConsumer

8.4.26.4 Attributes

No additional attributes.

8.4.26.5 Associations

No additional associations.

8.4.26.6 Constraints

[1] Asserts that an OperationalServiceConsumer must have at least one port that is stereotyped Service and that it requires an interface that is stereotyped ServiceSpecification

```
self.ownedPort->notEmpty() and

self.ownedPort->exists(p|

p.getAppliedStereotype('UPDM::Service')->notEmpty() and
p.required->notEmpty() and

p.required->exists(inf|inf.getAppliedStereotype('UPDM::ServiceSpecification')-
>notEmpty()))
```

8.4.26.7 Semantics

No additional semantic requirements.

8.4.26.8 Notation

No additional notational requirements.

8.4.27 OperationalServiceProvider

8.4.27.1 Extension

8.4.27.2 Generalizations

- OperationalRole (from OperationalView)

8.4.27.3 Description

The service provider provides one or more services. A service provider has a property that captures information about its location, although the meaning of this is implementation-dependent. The class acting as the service provider may not expose any attributes or operations directly: only public ports may be provided (stereotyped as service), and these are typed by service specifications.

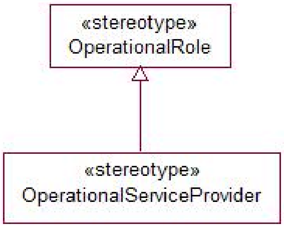


Figure 8.63 - OperationalServiceProvider

8.4.27.4 Attributes

No additional attributes.

8.4.27.5 Associations

No additional associations.

8.4.27.6 Constraints

[1] Asserts that an OperationalServiceProvider must have at least one port that is stereotyped Service and that it provides an interface that is stereotyped ServiceSpecification.

```
self.ownedPort->notEmpty() and

self.ownedPort->exists(p|

p.getAppliedStereotype('UPDM::Service')->notEmpty() and

p.provided->notEmpty() and

p.provided-> exists(inf|inf.getAppliedStereotype('UPDM::ServiceSpecification')->
notEmpty()))
```

8.4.27.7 Semantics

No additional semantic requirements.

8.4.27.8 Notation

No additional notational requirements.

8.4.28 OperationalStateTrace

8.4.28.1 Extension

- StateMachine (from UML2)

8.4.28.2 Generalizations

8.4.28.3 Description

This is the state machine for OV-5.

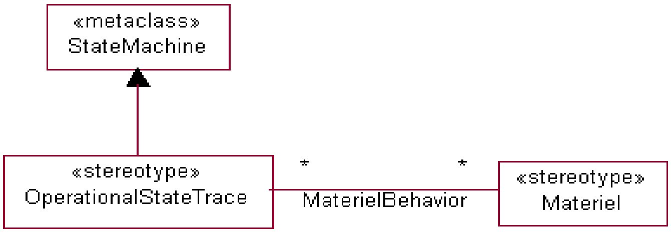


Figure 8.64 - OperationalStateTrace

8.4.28.4 Attributes

No additional attributes.

8.4.28.5 Associations

- materiel : Materiel [*]
  Materiel required for the StateMachine

8.4.28.6 Constraints

[1] Asserts that this StateMachine is owned either by an OperationalNode or an OperationalCapabilityRealization.

```
self.owner.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

self.owner.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty() or

self.owner.getAppliedStereotype('UPDM::OperationalCapabilityRealization') -
>notEmpty()
```

8.4.28.7 Semantics

No additional semantic requirements.

8.4.28.8 Notation

No additional notational requirements.

8.4.29 OperationalTask

8.4.29.1 Extension

- Operation (from UML2)

8.4.29.2 Generalizations

8.4.29.3 Description

OperationalTask is an operation on OperationalNodes that provides the means to invoke OperationalActivities.

OperationalTasks defined and standardized by the Joint Staff are in the form of Mission Essential Tasks [CJCSM 3500.04C, 2002]. OperationalTasks are also specified (and sometimes standardized) in the form of process activities arising from process modeling. It is sometimes convenient to merge these sets, either as activities or tasks.

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 (e.g., 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 Operational Activity Model). It is used to portray operational actions not hardware/software system functions.

The operational activities (from the OV-5 Operational Activity Model) performed by a given OperationalNode may be listed on the graphic, if space permits. OV-2, in effect, turns OV-5 inside out, focusing first-order on the Operational Nodes and second-order on the Activities. OV-5, on the other hand, places first-order attention on OperationalActivities and only second-order attention on OperationalNodes, which can be shown as annotations on the activities.

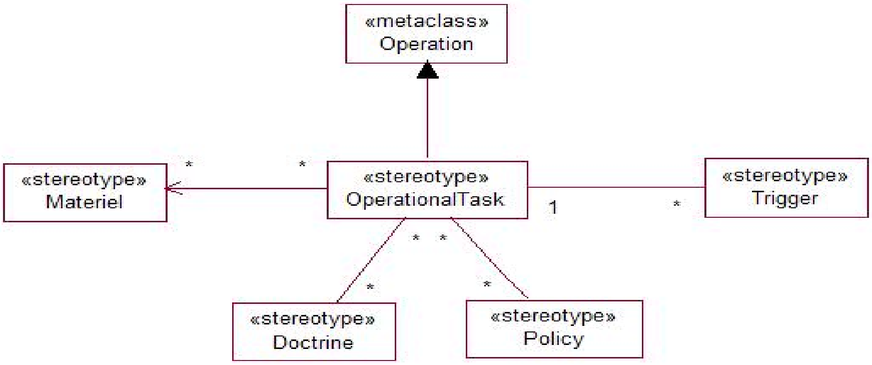


Figure 8.65 - OperationalTask

8.4.29.4 Attributes

No additional attributes.

8.4.29.5 Associations

- adheresToPolicy : Policy [*]
  Policies to which this OperationalTask adheres

- governedByDoctrine : Doctrine [*]
  Doctrines that govern this OperationalTask

- triggeredByEvents : Trigger [*]
  Triggers, callEvents, that trigger this OperationalTask

- utilizesMateriel : Materiel [*]
  Materiel utilized by this OperationalTask

8.4.29.6 Constraints

[1] Asserts that this OperationalTask adheres to zero or more Rules

```
self.rules-> forAll(getAppliedStereotype('UPDM::Rule')->notEmpty())
```

[2] Asserts that this OperationalTask adheres to zero or more Policies

```
self.adheresToPolicy-> forAll(getAppliedStereotype('UPDM::Policy')->notEmpty())
```

[3] Asserts that there are zero or more Doctrines that govern this OperationalTask

```
self.governedByDoctrine->forAll(getAppliedStereotype('UPDM::Doctrine')-
>notEmpty())
```

[4] Asserts that Materiel is utilized by this OperationalTask

```
self.utilizesMateriel-> forAll(getAppliedStereotype('UPDM::Materiel')->notEmpty())
```

[5] Asserts that the Triggers of this OperationalTask can be stereotyped Trigger, a callEventAction

```
self.triggeredByEvents->forAll(getAppliedStereotype('UPDM::Trigger') ->
notEmpty())
```

[6] Asserts that the Method property of this OperationalTask is not null and that its stereotype is OperationalActivity or OperationalStateTrace or OperationalEventTrace

```
self.method->notEmpty() and self.method->forAll(x|
(x.oclIsKindOf(uml::Activity)and
x.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty())

or (x.oclIsKindOf(uml::StateMachine)and
x.getAppliedStereotype('UPDM::OperationalStateTrace')->notEmpty())

or (x.oclIsKindOf(uml::StateMachine)and
x.getAppliedStereotype('UPDM::OperationalEventTrace')->notEmpty())

)
```

[7] Asserts that this OperationalTask is owned by an OperationalNode, one of its specializations or an OperationalNodeSpecification

```
self.owner.getAppliedStereotype('UPDM::OperationalNodeSpecification')-> notEmpty()
or

self.owner.getAppliedStereotype('UPDM::OperationalNode')-> notEmpty() or

self.owner.getAppliedStereotypes()->collect(allParents())-> any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty()
```

8.4.29.7 Semantics

OperationalTasks redefines operation in Node. In the EventTrace, the messages invoke these operations in the receiving Node.

An action is a named element that is the fundamental unit of executable functionality. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise. An action represents a single step within an activity, that is, one that is not further decomposed within the activity. An action has pre- and post-conditions.

A CallOperationAction is an action that transmits an operation call request to the target object, where it may cause the invocation of associated operation. The argument values of the action are available to the execution of the invoked behavior. If the action is marked synchronous, the execution of the CallOperationAction waits until the execution of the invoked behavior completes and a reply transmission is returned to the caller; otherwise execution of the action is complete when the invocation of the operation is established and the execution of the invoked operation proceeds concurrently with the execution of the calling behavior. Any values returned as part of the reply transmission are put on the result output pins of the CallOperationAction. Upon receipt of the reply transmission, execution of the CallOperationAction is complete.

The activity diagram that represents the OperationalActivity should have its specification set to the OperationalActivity that it models.

OperationalTasks are modeled as operations on OperationalNodes. OperationalCapabilitiesRealizations are collaborations among OperationalNodes exchanging information through the OperationalActivities and OperationalTasks. The OperationalActivities and OperationalTasks that are used in the OperationalCapabilityRealization contribute to that OperationalCapability. The OperationalCapabilities can be explicitly noted in the OperationalActivity by recording them in the contributesToCapability property.

If an OperationalTask is owned by an OrganizationalRole, then the OrganizationalRole is said to perform that task.

8.4.29.8 Notation

No additional notational requirements.

8.4.30 OperationalView

8.4.30.1 Extension

8.4.30.2 Generalizations

- ArchitectureView (from AllViews)

8.4.30.3 Description

OperationalView is a package that contains information for one or more of the OperationalView deliverables or information used for the OperationalView.

An Operational View (OV) describes the tasks and activities, operational elements and exchanges of information required to conduct operations. A pure OV is materiel independent. However, operations and their relationships may be influenced by new technologies such as collaboration technology, where process improvements are in practice before policy can reflect the new procedures. There may be some cases, as well, in which it is necessary to document the way processes are performed given the restrictions of current systems, in order to examine ways in which new systems could facilitate streamlining the processes. In such cases, an OV may have materiel constraints and requirements that must be addressed. For this reason, it may be necessary to include some high-level Systems View (SV) architecture data as overlays or augmenting information onto the OV products.

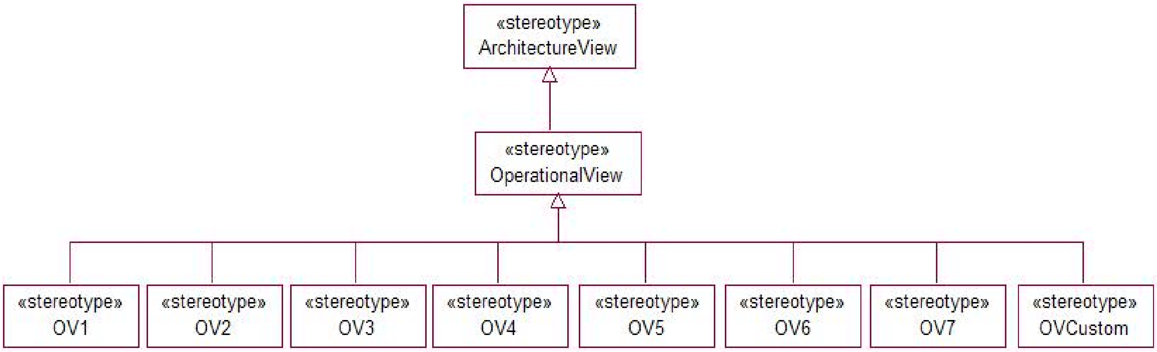


Figure 8.66 - OperationalView

8.4.30.4 Attributes

No additional attributes.

8.4.30.5 Associations

No additional associations.

8.4.30.6 Constraints

No additional constraints.

8.4.30.7 Semantics

OperationalViews are UML packages that contain information that is to be assembled into the OperationalView deliverable. The stereotyped package indicates to the user that the contents inside that package satisfy the indicated framework product or user defined product. The package here is viewed as an organizing mechanism for views into the architecture, not as a package to hold DoDAF elements. Individual elements will typically reside outside of these packages with the diagrams that show the views for these elements inside the package.

8.4.30.8 Notation

No additional notational requirements.

8.4.31 OrganizationalRelationship

8.4.31.1 Extension

- Association (from UML2)

8.4.31.2 Generalizations

8.4.31.3 Description

OrganizationalRelationship describes the relationship between two OrganizationalResources.

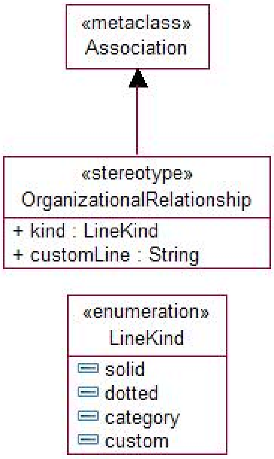


Figure 8.67 - OrganizationalRelationship and LineKind Enumeration

8.4.31.4 Attributes

- customLine : String [1]
- kind : LineKind [1]

8.4.31.5 Associations

No additional associations.

8.4.31.6 Constraints

[1] Asserts that both ends of this association are typed by an OrganizationalResource.

```
self.memberEnd->at(1).type.getAppliedStereotype('UPDM::OrganizationalResource')-
>notEmpty() and
```

```
self.memberEnd->at(2).type.getAppliedStereotype('UPDM::OrganizationalResource')-
>notEmpty()
```

[2] Asserts that if kind is set to solid, the OrganizationalRelationship association is composite.

```
(self.kind=LineKind::solid) implies
```

```
self.memberEnd->select(m | m.isComposite)->notEmpty()
```

[3] Asserts that if kind is set to dotted, the OrganizationalRelationship association is not composite.

```
(self.kind=LineKind::dotted) implies
```

```
self.memberEnd->select(m | m.isComposite)->isEmpty()
```

8.4.31.7 Semantics

No additional semantic requirements.

8.4.31.8 Notation

No additional notational requirements.

8.4.32 OrganizationalResource

8.4.32.1 Extension

8.4.32.2 Generalizations

- Resource (from AllViews)

8.4.32.3 Description

An OrganizationalResource is a Resource that can be deployed to SystemsNodes and can play OperationalCapabilityRoles. OrganizationalResources own Projects and are used to configure Capabilities.

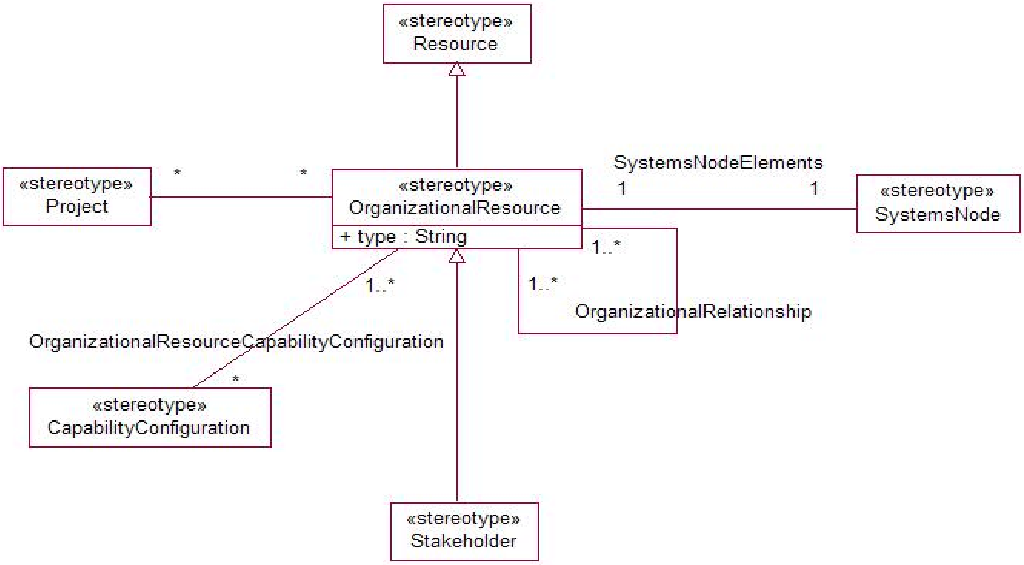


Figure 8.68 - OrganizationalResource

8.4.32.4 Attributes

- type : String [1]
  A user defined type of OrganizationalResource

8.4.32.5 Associations

- capabilityConfiguration : CapabilityConfiguration [*]
  See OrganizationalResourceCapabilityConfiguration for more details.

- organizationalRelationship1 : OrganizationalResource [1..*]
  See OrganizationalRelationship for more details.

- organizationalRelationship2 : OrganizationalResource [1..*]
  See OrganizationalRelationship for more details.

- projects : Project [*]
  The Projects that are owned by this OrganizationalResource

- systemsnode : SystemsNode [1]
  SystemsNodes on which this Resource is deployed

8.4.32.6 Constraints

[1] Asserts that there is at least one Project owned by this OrganizationalResource.

```
self.projects.getAppliedStereotype('UPDM::Project')->notEmpty()
```

[2] Asserts that an association exists between this OrganizationalResource and another.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::OrganizationalRelationship')-> notEmpty())-
>notEmpty()
```

[3] Asserts that there is an association between the OperationalNode and a SystemsNode that houses the OperationalNodes.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

(getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

8.4.32.7 Semantics

Use the DeployedToSystemsNode association to indicate deployment of OrganizationalResource to a SystemsNode.

Use the OperationalCapabilityRoleResource association to indicate an OrganizationalRole played by the OrganizationalResource.

Use the OrganizationalResourceCapabilityConfiguration association to indicate inclusion in a CapabilityConfiguration

Use the DeployedToSystemsNode association to indicate deployment of OrganizationalResource to a SystemsNode.

Use the DeployedToSystemsNode association to indicate deployment of OrganizationalResource to a SystemsNode.

8.4.32.8 Notation

No additional notational requirements.

8.4.32.9 OrganizationalRole

8.4.32.10 Extension

- Property (from UML2)

8.4.32.11 Generalizations

8.4.32.12 Description

An OrganizationalRole indicates a role played by a Resource or OperationalNode in a specific context. That context is typically provided by another OperationalNode. This also allows for hierarchical decomposition of in particular OperationalNodes.

The same OperationalNode or Resource can play multiple roles within a single context, or may be used differently across multiple contexts.

In the case where the OrganizationalRole is typed by an OperationalNode it can also be connected to other such OrganizationalRoles using Needlines. Needlines attached to OrganizationalRoles are always contextual as well, i.e., they are only valid in the specific context where OrganizationalRoles are used.

An OrganizationalRole may be related to any number of Competencies through CompetenceRelationships.

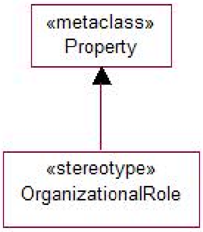


Figure 8.69 - OrganizationalRole

8.4.32.13 Attributes

No additional attributes.

8.4.32.14 Associations

No additional associations.

8.4.32.15 Constraints

[1] Asserts that an OrganizationalRole is typed by an OperationalNode, a Resource or a CapabilityConfiguration.

```
self.type-> forAll(getAppliedStereotype('UPDM::OperationalNode')-> notEmpty() or

getAppliedStereotypes()->collect(allParents())-> any(qualifiedName =
'UPDM::OperationalNode') -> notEmpty() or

getAppliedStereotype('UPDM::Resource')->notEmpty() or

getAppliedStereotypes()->collect(allParents())-> any(qualifiedName =
'UPDM::Resource') ->notEmpty() or

getAppliedStereotype('UPDM::CapabilityConfiguration')->notEmpty())
```

8.4.32.16 Semantics

No additional semantic requirements.

8.4.32.17 Notation

No additional notational requirements.

8.4.33 OV-1: High Level Operational Concept

8.4.33.1 Extension

8.4.33.2 Generalizations

- OperationalView (from OperationalView)

8.4.33.3 Description

High-Level Operational Concept Graphic - High-level graphical/textual description of operational concept.

Variation a) High-Level Operational Concept Graphic - High-level graphical/textual description of operational concept.

Variation b) Operational Concept Description - Provides a supplementary description of the High-Level Operational Concept Graphic

Variation c) Operational Performance Attributes - Provides detail of the operational performance attributes associated with the scenario / use case represented in the High-Level Operational Concept Graphic

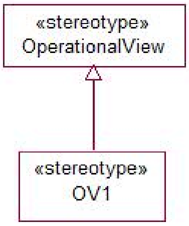


Figure 8.70 - OV-1: High Level Operational Concept

8.4.33.4 Attributes

No additional attributes.

8.4.33.5 Associations

No additional associations.

8.4.33.6 Constraints

No additional constraints.

8.4.33.7 Semantics

No additional semantic requirements.

8.4.33.8 Notation

No additional notational requirements.

8.4.34 OV-2: Operational Node Connectivity Description

8.4.34.1 Extension

8.4.34.2 Generalizations

- OperationalView (from OperationalView)

8.4.34.3 Description

Operational Node Connectivity - Description of operational nodes, connectivity, and information exchange Needlines between operational nodes.

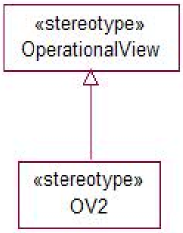


Figure 8.71 - OV-2: Operational Node Connectivity Description

8.4.34.4 Attributes

No additional attributes.

8.4.34.5 Associations

No additional associations.

8.4.34.6 Constraints

No additional constraints.

8.4.34.7 Semantics

No additional semantic requirements.

8.4.34.8 Notation

No additional notational requirements.

8.4.35 OV-3: Operational Information Exchange Matrix

8.4.35.1 Extension

8.4.35.2 Generalizations

- OperationalView (from OperationalView)

8.4.35.3 Description

Operational Information Exchange Matrix - Information exchanged between operational nodes and the relevant attributes of that exchange.

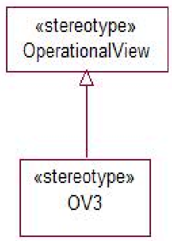


Figure 8.72 - OV-3: Operational Information Exchange Matrix

8.4.35.4 Attributes

No additional attributes.

8.4.35.5 Associations

No additional associations.

8.4.35.6 Constraints

No additional constraints.

8.4.35.7 Semantics

No additional semantic requirements.

8.4.35.8 Notation

No additional notational requirements.

8.4.36 OV-4: Organizational Relationships Chart

8.4.36.1 Extension

8.4.36.2 Generalizations

- OperationalView (from OperationalView)

8.4.36.3 Description

Organizational Relationships Chart - Organizational, role, or other relationships among organizations.

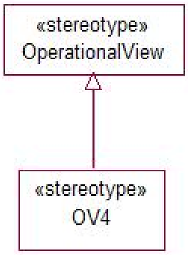


Figure 8.73 - OV-4: Organizational Relationships Chart

8.4.36.4 Attributes

No additional attributes.

8.4.36.5 Associations

No additional associations.

8.4.36.6 Constraints

No additional constraints.

8.4.36.7 Semantics

No additional semantic requirements.

8.4.36.8 Notation

No additional notational requirements.

8.4.37 OV-5: Operational Activity Model

8.4.37.1 Extension

8.4.37.2 Generalizations

- OperationalView (from OperationalView)

8.4.37.3 Description

Operational Activity Model - Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing operational nodes, or other pertinent information.

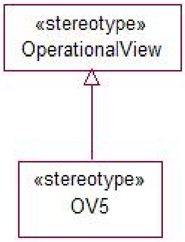


Figure 8.74 - OV-5: Operational Activity Model

8.4.37.4 Attributes

No additional attributes.

8.4.37.5 Associations

No additional associations.

8.4.37.6 Constraints

No additional constraints.

8.4.37.7 Semantics

No additional semantic requirements.

8.4.37.8 Notation

No additional notational requirements.

8.4.38 OV-6: Operational Activity Sequence and Timing Descriptions (OV-6A, 6B and 6C)

8.4.38.1 Extension

8.4.38.2 Generalizations

- OperationalView (from OperationalView)

8.4.38.3 Description

Variation a) Operational Rules Model - Identifies business rules that constrain the operations.

Variation b) Operational State Transition Description - Identifies business process responses to events.

Variation c) Operational Event-Trace Description - Traces actions in a scenario or sequence of events.

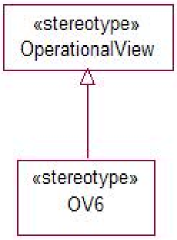


Figure 8.75 - OV-6: Operational Activity, Sequence and Timing Descriptions

8.4.38.4 Attributes

No additional attributes.

8.4.38.5 Associations

No additional associations.

8.4.38.6 Constraints

No additional constraints.

8.4.38.7 Semantics

No additional semantic requirements.

8.4.38.8 Notation

No additional notational requirements.

8.4.39 OV-7: Logical Data Model

8.4.39.1 Extension

8.4.39.2 Generalizations

- OperationalView (from OperationalView)

8.4.39.3 Description

Logical Data Model - Documentation of the system data requirements and structural business process rules of the OperationalView.

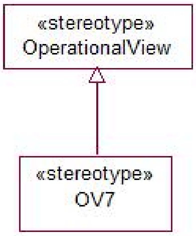


Figure 8.76 - OV-7: Logical Data Model

8.4.39.4 Attributes

No additional attributes.

8.4.39.5 Associations

No additional associations.

8.4.39.6 Constraints

No additional constraints.

8.4.39.7 Semantics

No additional semantic requirements.

8.4.39.8 Notation

No additional notational requirements.

8.4.40 OVCustom: Custom Operational View

8.4.40.1 Extension

8.4.40.2 Generalizations

- OperationalView (from OperationalView)

8.4.40.3 Description

A user-defined view that applies to the OperationalView.

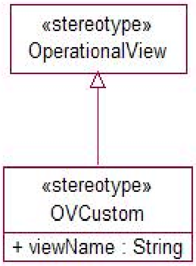


Figure 8.77 - OVCustom: Custom Operational View

8.4.40.4 Attributes

- viewName : String [1]
  Provides a name for the custom view so that it can be easily distinguished from other custom OperationalViews.

8.4.40.5 Associations

No additional associations.

8.4.40.6 Constraints

No additional constraints.

8.4.40.7 Semantics

No additional semantic requirements.

8.4.40.8 Notation

No additional notational requirements.

8.4.41 Policy

8.4.41.1 Extension

- Class (from UML2)

8.4.41.2 Generalizations

8.4.41.3 Description

An established course of action that must be followed. Policies can be the source of rules that show up in the operational activities.

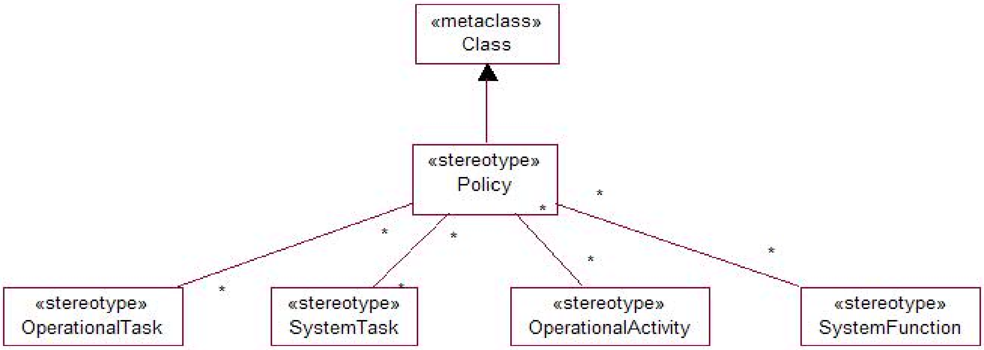


Figure 8.78 - Policy

8.4.41.4 Attributes

No additional attributes.

8.4.41.5 Associations

- activity : OperationalActivity [*]
  This policy governs these Activities
- operationaltask : OperationalTask [*]
  OperationalTasks governed by this Policy
- SystemFunction : SystemFunction [*]
  SystemFunctions governed by this Policy
- systemtask : SystemTask [*]
  SystemTasks governed by this Policy

8.4.41.6 Constraints

[1] Asserts that this policy governs the OperationalActivities and SystemFunctions

```
self.activity->forAll(getAppliedStereotype('UPDM::OperationalActivity')-
>notEmpty() or

getAppliedStereotype('UPDM::SystemFunction')->notEmpty())
```

8.4.41.7 Semantics

No additional semantic requirements.

8.4.41.8 Notation

No additional notational requirements.

8.4.42 RealizedOperationalCapability

8.4.42.1 Extension

- Realization (from UML2)

8.4.42.2 Generalizations

8.4.42.3 Description

Specifies that an OperationalCapabilityRealization realizes an OperationalCapability.

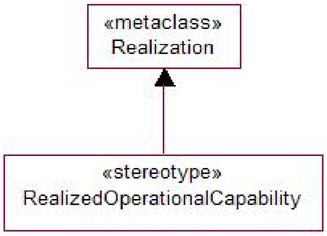


Figure 8.79 - RealizedOperationalCapability

8.4.42.4 Attributes

No additional attributes.

8.4.42.5 Associations

No additional associations.

8.4.42.6 Constraints

[1] Asserts that the supplier or target of the realization is an OperationalCapability and the client or source is an OperationalCapabilityRealization.

```
let real:uml::Realization=self.oclAsType(uml::Realization) in real->
any(supplier.getAppliedStereotype('UPDM::OperationalCapability')-> notEmpty())-
>notEmpty() and real->
any(client.getAppliedStereotype('UPDM::OperationalCapabilityRealization')->
notEmpty())->notEmpty()
```

8.4.42.7 Semantics

No additional semantic requirements.

8.4.42.8 Notation

No additional notational requirements.

8.4.43 RealizedOperationalSpecification

8.4.43.1 Extension

- Realization (from UML2)

8.4.43.2 Generalizations

8.4.43.3 Description

Specifies that a OperationalNode or one of its sub stereotypes realizes an interface stereotyed OperationalNodeSpecification or one of its sub stereotypes.

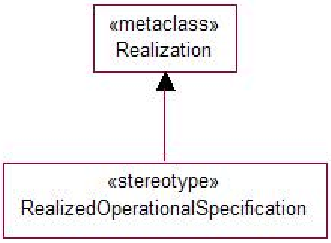


Figure 8.80 - RealizedOperationalSpecification

8.4.43.4 Attributes

No additional attributes.

8.4.43.5 Associations

No additional associations.

8.4.43.6 Constraints

[1] Asserts that the source or client of the realization is an OperationalNode or one of its specializations and the target or supplier is an OperationalNodeSpecification

```
self.supplier.getAppliedStereotype('UPDM::OperationalNodeSpecification')-
>notEmpty()

and

(self.client.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

self.client.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty())
```

8.4.43.7 Semantics

No additional semantic requirements.

8.4.43.8 Notation

No additional notational requirements.

8.4.44 ResourceCompetence

8.4.44.1 Extension

- Association (from UML2)

8.4.44.2 Generalizations

8.4.44.3 Description

Specifies that zero or more Competencies are provided by a Resource.

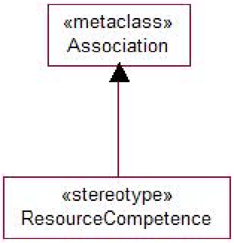


Figure 8.81 - ResourceCompetence

8.4.44.4 Attributes

No additional attributes.

8.4.44.5 Associations

No additional associations.

8.4.44.6 Constraints

[1] Asserts that there are zero or more Competencies provided by the Resource and that zero or more Resources provide a Competence and that these are the elements associated by this association

```
(self.endType-> at(1).getAppliedStereotype('UPDM::Competence')-> notEmpty() and

self.endType-> at(2).getAppliedStereotype('UPDM::Resource')-> notEmpty() or

self.base_Association.endType-> at(2).getAppliedStereotypes()->
collect(allParents())-> any(qualifiedName = 'UPDM::Resource') ->notEmpty()

) or

(self.endType-> at(2).getAppliedStereotype('UPDM::Competence')-> notEmpty() and

self.endType-> at(1).getAppliedStereotype('UPDM::Resource')-> notEmpty() or
self.base_Association.endType-> at(1).getAppliedStereotypes()->
collect(allParents())-> any(qualifiedName = 'UPDM::Resource') ->notEmpty() )
```

8.4.44.7 Semantics

No additional semantic requirements.

8.4.44.8 Notation

No additional notational requirements.

8.4.45 ResultsOfEffect

8.4.45.1 Extension

- State (from UML2)

8.4.45.2 Generalizations

8.4.45.3 Description

In effects based planning, the focus is on the outcome of an Effect. This is that end state, the ResultsOfEffect.

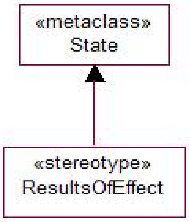


Figure 8.82 - ResultsOfEffect

8.4.45.4 Attributes

No additional attributes.

8.4.45.5 Associations

No additional associations.

8.4.45.6 Constraints

[1] Asserts that the state transition that resulted in this state is a CausesEffect transition.

```
self.incoming.getAppliedStereotype('UPDM::CasuseEffect')->notEmpty()
```

8.4.45.7 Semantics

No additional semantic requirements.

8.4.45.8 Notation

No additional notational requirements.

8.4.46 Rule

8.4.46.1 Extension

- Constraint (from UML2)

8.4.46.2 Generalizations

8.4.46.3 Description

A constraint to be applied to elements of special interest to DoDAF and MODAF models.

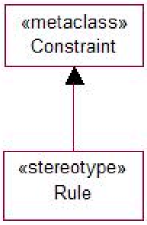


Figure 8.83 - Rule

8.4.46.4 Attributes

No additional attributes.

8.4.46.5 Associations

No additional associations.

8.4.46.6 Constraints

No additional constraints.

8.4.46.7 Semantics

No additional semantic requirements.

8.4.46.8 Notation

No additional notational requirements.

8.4.47 TaskInvocation

8.4.47.1 Extension

- Message (from UML2)

8.4.47.2 Generalizations

8.4.47.3 Description

The event that invokes either an OperationalTask or a SystemTask in the EventTrace.

The ends of the connector associated with the message are the source and target of the information flow and the argument is the information element. If there is no argument, then the invocation, itself, is the information element - i.e., a command.

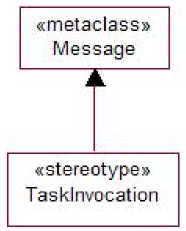


Figure 8.84 - TaskInnovation

8.4.47.4 Attributes

No additional attributes.

8.4.47.5 Associations

No additional associations.

8.4.47.6 Constraints

[1] Asserts that the source and target are either both OperationalNodes, one of its specializations or an OperationalNodeSpecification; or System, one of its specializations or a SystemFunctionSpecification. The source and target are determined from the connector. The target is the owner of the operation. Asserts that the signature of the message is an OperationalTask or a SystemTask.

```
let left:uml::Class = self.connector.end->asOrderedSet()-
>at(1).role.type.oclAsType(uml::Class) in

let right:uml::Class = self.connector.end->asOrderedSet()-
>at(2).role.type.oclAsType(uml::Class)in

(self.signature.getAppliedStereotype('UPDM::OperationalTask')->notEmpty() and

  ((left.getAppliedStereotype('UPDM::OperationalNodeSpecification')->notEmpty() or

left.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

     left.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty())

      and

(right.getAppliedStereotype('UPDM::OperationalNodeSpecification')->notEmpty() or
     right.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

     right.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty())))

or

(self.signature.getAppliedStereotype('UPDM::SystemTask')->notEmpty() and

  (((left.getAppliedStereotype('UPDM::System')->notEmpty() or

      left.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty() or

      left.getAppliedStereotype('UPDM::SystemFunctionSpecification')->notEmpty() )

          and

    ( right.getAppliedStereotype('UPDM::System')->notEmpty() or

      right.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty() or

right.getAppliedStereotype('UPDM::SystemFunctionSpecification')->notEmpty() ))))
```

8.4.47.7 Semantics

No additional semantic requirements.

8.4.47.8 Notation

No additional notational requirements.

8.4.48 NodeType Enumerated Type

8.4.48.1 Description

Nodes can represent many different types of entities. NodeType is an enumeration of common types including a Custom type to be defined by the user.

8.4.48.2 Literals

Table 8.1 - NodeType Enumeration Literals

Literal	Definition
CommunityOfInterest	Community of Interests that participate in the enterprise.
Personnel	Man in the loop. This is used when the Node is generally realized as a human.
Organization	Organizations are composed of departments, divisions, etc. and ultimately people who perform tasks. More than one organization can realize a Node.
OrganizationType	OrganizationType is a logical or functional grouping of Organizations (e.g., Logistics, Intelligence, etc.).
Custom	User defined

8.4.49 LineKind Enumerated Type

8.4.49.1 Description

Indicates how one OrganizationalResource relates to another

8.4.49.2 Literals

Table 8.2 - LineKind Enumeration Literals

Literal	Defintion
solid	Indicates a solid line between organizational resources
dotted	Indicates a dotted line between organizational resources
category	Indicates a categorization relationship between organizational resource.
custom	Indicates a custom relationship between organizational resources

8.4.50 CompetenceKind Enumerated Type

8.4.50.1 Description

CompetenceKind is an enumeration that is used to indicate how a Resource, OrganizationalNeed, or OrganizationalRole relates to a Competence.

8.4.50.2 Literals

Table 8.3 - CompetenceKink Enumeration Literals

Literal	Defintion
custom	Indicates a custom relationship to a competence
desired	Indicates that a competence is desired
provided	Indicates that a competence is provided
required	Indicates that a competence is required

8.4.51 PerformanceMeasurePeriod

8.4.51.1 Description

The three periods for which PerformanceMeasures are defined

8.4.51.2 Literals

Table 8.4 - PerformanceMeasurePeriod

Literal	Defintion
baseline	Baseline for the measurement

Table 8.4 - PerformanceMeasurePeriod

actual	Actual measurement
target	Target for the measurement

8.5 StrategicView Package

8.5.1 Overview

The StrategicView Package contains UML stereotypes that assist the modeler in developing the views defined in the MODAF Capability Management process. These elements include capability and configuration, effects and the relationship between capabilities and the resources required to realize them.

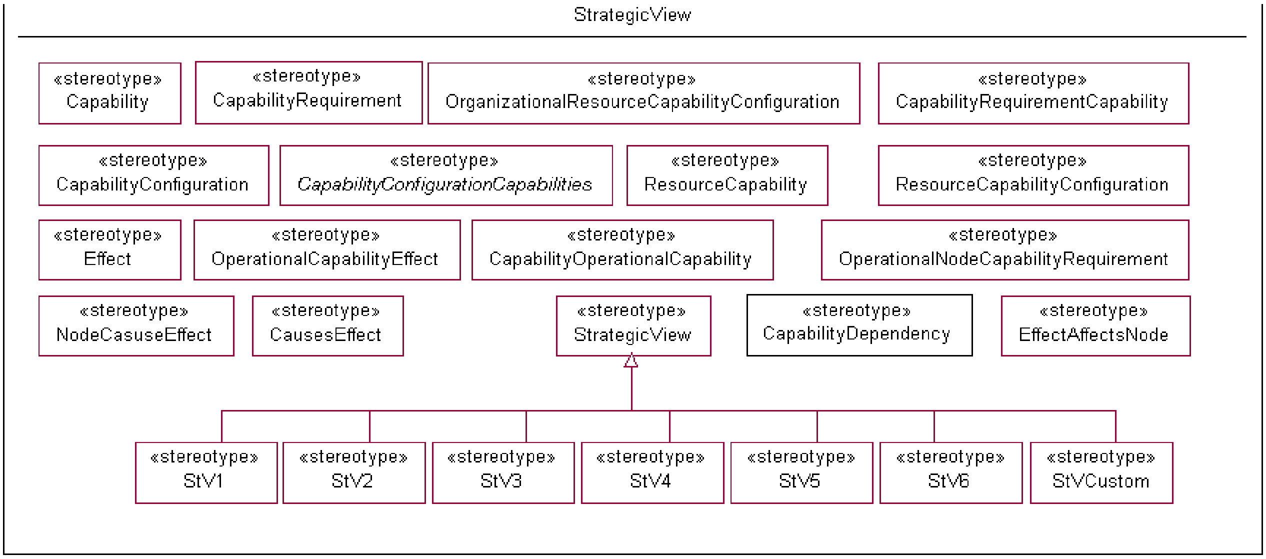

StrategicView
«stereotype» Capability
«stereotype» CapabilityRequirement
«stereotype» OrganizationalResourceCapabilityConfiguration
«stereotype» CapabilityRequirementCapability
«stereotype» CapabilityConfiguration
«stereotype» *CapabilityConfigurationCapabilities*
«stereotype» ResourceCapability
«stereotype» ResourceCapabilityConfiguration
«stereotype» Effect
«stereotype» OperationalCapabilityEffect
«stereotype» CapabilityOperationalCapability
«stereotype» OperationalNodeCapabilityRequirement
«stereotype» NodeCasuseEffect
«stereotype» CausesEffect
«stereotype» StrategicView
«stereotype» CapabilityDependency
«stereotype» EffectAffectsNode
«stereotype» StV1
«stereotype» StV2
«stereotype» StV3
«stereotype» StV4
«stereotype» StV5
«stereotype» StV6
«stereotype» StVCustom

Figure 8.85 - StrategicView Package Overview

8.5.2 Capability

8.5.2.1 Extension

- Class (from UML2)

8.5.2.2 Generalizations

8.5.2.3 Description

A high level specification of the enterprise's ability to execute a specified course of action.

The Capability is expressed in terms of the Resources required to implement the Capability, the OperationalCapabilities that contribute to the Capability and a CapabilityConfiguration that specifies the other aspects of the Capability.

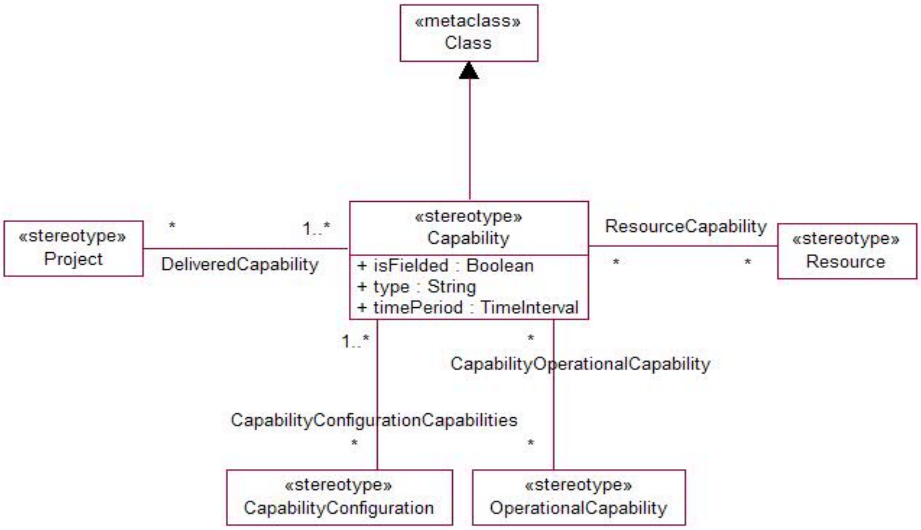


Figure 8.86 - Capability

8.5.2.4 Attributes

- isFielded : Boolean [1]
  Indicates whether the Capability has been fielded.
- timePeriod : TimeInterval [1]
  The TimePeriod that bounds this Capability
- type : String [1]
  A user defined type designation of the Capability

8.5.2.5 Associations

- capabilityConfiguration : CapabilityConfiguration [*]
  CapabilityConfigurations specify Resources that combine to provide a Capability.
- operationalCapability : OperationalCapability [*]
  OperationalCapabilities that express this strategic Capability
- project : Project [*]
  Projects that are designed to deliver this Capability
- resource : Resource [*]
  Resources required for this Capability

8.5.2.6 Constraints

[1] Asserts that a Project delivers this Capability

```
self.getAllAttributes()->asOrderedSet()->
```

```
select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::DeliveredCapability')-> notEmpty())->notEmpty()
```

[2] Asserts that a TimePeriod bounds this Capability

```
self.timePeriod->forAll(classifier->forAll(name='TimeInterval')) or

self.timePeriod->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='TimeInterval')))
```

[3] Asserts that a Capability is realized by at least one OperationalCapability.

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::CapabilityOperationalCapability')-> notEmpty())-
>notEmpty()
```

[4] Asserts that a CapabilityConfigurationCapabilities association exists between CapabilityConfiguration and Capability.

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::CapabilityConfigurationCapabilities')-> notEmpty())-
>notEmpty()
```

[5] Asserts that a ResourceCapability association exists between Resource and the Capabilities that it provides.

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::ResourceCapability')-> notEmpty())->notEmpty()
```

[6] Asserts that there is a dependency relationship between Capabilities

```
self.clientDependency->notEmpty() and

self.clientDependency->any(getAppliedStereotype('UPDM::CapabilityDependency')->
notEmpty())->notEmpty()
```

[7] Asserts that an OperationalCapability exists for this Capability

```
self.useCase->exists (getAppliedStereotype('UPDM::OperationalCapability')->
notEmpty() )
```

8.5.2.7 Semantics

The MODAF StrategicView encourages a rigorous decomposition of the capabilities for the organization to be used for a number of architectural descriptions. OperationalCapabilities and OperationalCapabilityRealizations can be used to relate the Strategic Capabilities to the Operational Views.

8.5.2.8 Notation

No additional notational requirements.

8.5.3 CapabilityConfiguration

8.5.3.1 Extension

- Class (from UML2)

8.5.3.2 Generalizations

8.5.3.3 Description

A CapabilityConfiguration is a combination of organizational aspects with their competencies and equipment, i.e., Resources that combine to provide a Capability. A CapabilityConfiguration is a physical asset or organization configured to provide a capability, and must be guided by Doctrine. (Source MODAF)

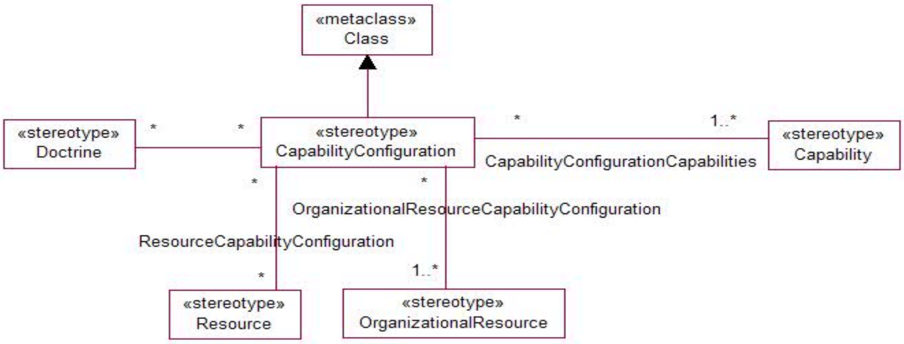


Figure 8.87 - CapabilityConfiguration

8.5.3.4 Attributes

No additional attributes.

8.5.3.5 Associations

- capability : Capability [1..*]
  The Capability that is configured by this configuration. See CapabilityConfigurationCapabilities for more details.

- doctrine : Doctrine [*]
  Doctrines that govern this CapabilityConfiguration

- materiel : Materiel [*]
  Materiel used in this CapabilityConfiguration

- organizationalResource : OrganizationalResource [1..*]
  The OrganizationalResource that owns or is responsible for the Capability See OrganizationalResourceCapabilityConfiguration for more details.

- resource : Resource [*]
  Resources that combine to provide a Capability. See ResourceCapabilityConfiguration for more details.

8.5.3.6 Constraints

[1] Asserts that all the Doctrine entries are types of Doctrines

```
self.doctrine->forAll(getAppliedStereotype('UPDM::Doctrine')->notEmpty())
```

[2] Asserts that a System is used by an OrganizationalResource. The context for the usage must be specified in terms of the CapabilityConfiguration that the system is contributing to.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::OrganizationalResourceCapabilityConfiguration')->
notEmpty())->notEmpty()
```

[3] Asserts that a CapabilityConfigurationCapabilities association exists between CapabilityConfiguration and Capability.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::CapabilityConfigurationCapabilities')-> notEmpty())-
>notEmpty()

self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::ResourceCapabilityConfiguration')-> notEmpty())-
>notEmpty()
```

8.5.3.7 Semantics

CapabilityConfiguration. associates Resources with Capabilities. In addition, it has a specific association for the OrganizationalResource that owns or is responsible for the Capability.

8.5.3.8 Notation

No additional notational requirements.

8.5.4 CapabilityConfigurationCapabilities

8.5.4.1 Extension

- Association (from UML2)

8.5.4.2 Generalizations

8.5.4.3 Description

Defines the association between a CapabilityConfiguration and the Capabilities that are configured by it.

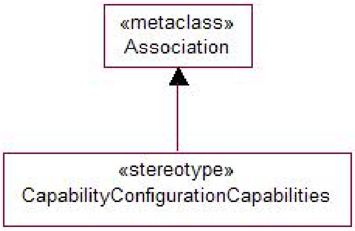


Figure 8.88 - CapabilityConfigurationCapabilities

8.5.4.4 Attributes

No additional attributes.

8.5.4.5 Associations

No additional associations.

8.5.4.6 Constraints

[1] Asserts that there are zero or more Capabilities configured by the CapabilityConfiguration and that these are the elements associated with this association

```
(self.endType->at(1).getAppliedStereotype('UPDM::Capability')->notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::CapabilityConfiguration')-
>notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::Capability')->notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::CapabilityConfiguration')-
>notEmpty())
```

8.5.4.7 Semantics

No additional semantic requirements.

8.5.4.8 Notation

No additional notational requirements.

8.5.5 CapabilityDependency

8.5.5.1 Extension

- Dependency (from UML2)

8.5.5.2 Generalizations

8.5.5.3 Description

Specifies that one Capability depends on another Capability (at the strategic level).

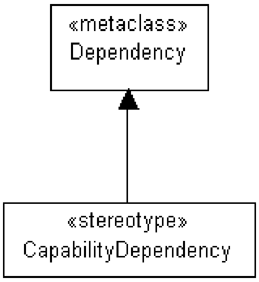


Figure 8.89 - CapabilityDependency

8.5.5.4 Attributes

No additional attributes.

8.5.5.5 Associations

No additional associations.

8.5.5.6 Constraints

[1] Asserts that the provider and the client are Capabilities.

```
self.client.getAppliedStereotype('UPDM::Capability')-> notEmpty()and
self.supplier.getAppliedStereotype('UPDM::Capability')-> notEmpty()
```

8.5.5.7 Semantics

No additional semantic requirements.

8.5.5.8 Notation

No additional notational requirements.

8.5.6 CapabilityOperationalCapability

8.5.6.1 Extension

- Association (from UML2)

8.5.6.2 Generalizations

8.5.6.3 Description

Defines the association between a Capability and the OperationalCapabilities that are supported by it.

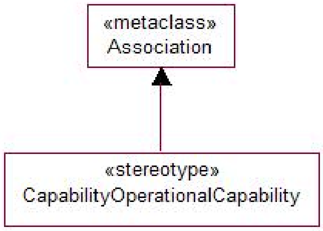


Figure 8.90 - CapabilityOperationalCapability

8.5.6.4 Attributes

No additional attributes.

8.5.6.5 Associations

No additional associations.

8.5.6.6 Constraints

[1] Asserts that there are zero or more Capabilities that contribute to zero or more OperationalCapabilities and that these are the elements associated with this association

```
(self.endType->at(1).getAppliedStereotype('UPDM::Capability')->notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::OperationalCapability')-
>notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::Capability')->notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::OperationalCapability')-
>notEmpty())
```

8.5.6.7 Semantics

This association indicates that the strategic Capability is the specification of the resources necessary to realize this OperationalCapability. Therefore, when this association is created, the Capability must be added to the list of subjects of the OperationalCapability and OperationalCapability added to the list of useCases of the Capability.

8.5.6.8 Notation

No additional notational requirements.

8.5.7 CapabilityRequirement

8.5.7.1 Extension

8.5.7.2 Generalizations

- ConformingElement (from AllViews)

8.5.7.3 Description

A CapabilityRequirement defines an association between an OperationalCapability and an OperationalNode that provides the Capability and the TimePeriod and the Milestones that apply to that requirement.

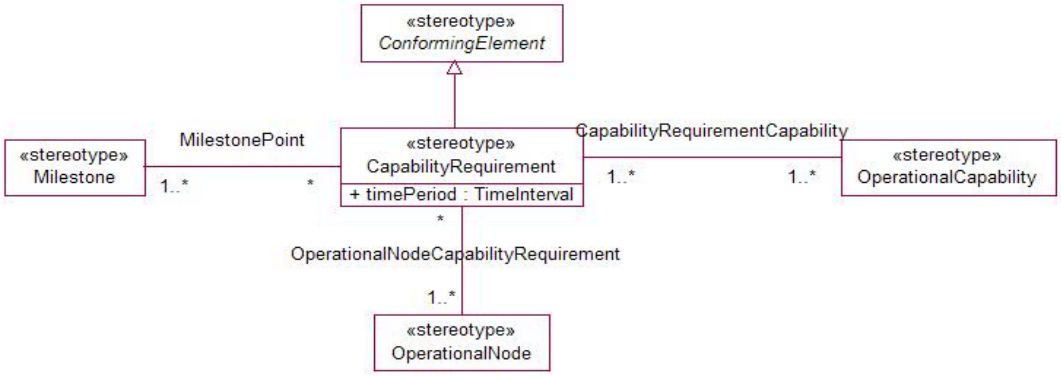


Figure 8.91 - CapabilityRequirement

8.5.7.4 Attributes

- timePeriod : TimeInterval [1]
  The TimePeriod in which the CapabilityRequirement is in effect.

8.5.7.5 Associations

- milestone : Milestone [1..*]
  Milestones that govern this requirement. See MilestonePoint for more details.
- operationalCapability : OperationalCapability [1..*]
  OperationalCapabilities associated with this strategic CapabilityRequirement. See CapabilityRequirementCapability for more details.
- operationalNode : OperationalNode [1..*]
  OperationalNodes that collaborate to meet this strategic CapabilityRequirement. See OperationalNodeCapabilityRequirement for more details.

8.5.7.6 Constraints

[1] Asserts that a Milestone exists that bounds the CapabilityRequirement

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()).association->any

  (getAppliedStereotype('UPDM::MilestonePoint')-> notEmpty())->notEmpty()
```

[2] Asserts that a TimePeriod exists that bounds the CapabilityRequirement

```
self.timePeriod->forAll(classifier->forAll(name='TimeInterval')) or

self.timePeriod->forAll(classifier->forAll(allParents()->notEmpty() and allParents()->exists(name='TimeInterval')))
```

[3] Asserts that there is a Capability Requirement for this OperationalNode

[4] Asserts that an association exists between the CapabilityRequirement and an OperationalCapability

```
self.getAllAttributes()->asOrderedSet()->select(association->
notEmpty()).association->
any(getAppliedStereotype('UPDM::OperationalNodeCapabilityRequirement')->
notEmpty())->notEmpty()
```

[5] Asserts that an OperationalCapability is realized by at least one CapabilityRequirement.

```
self.useCase->exists(getAppliedStereotype('UPDM::OperationalCapability')-
>notEmpty() )
```

8.5.7.7 Semantics

CapabilityRequirement associates an OperationalNode and OperationalCapability. The OperationalCapability is defined as a collaboration among OperationalNodes whose behavior is reflected in an EventTrace, an OperationalActivity or an OperationalStateTrace. The CapabilityRequirement shows the deliverable requirements - metrics, time period, and milestones.

8.5.7.8 Notation

No additional notational requirements.

8.5.8 CapabilityRequirementCapability

8.5.8.1 Extension

- Association (from UML2)

8.5.8.2 Generalizations

8.5.8.3 Description

Asserts that a CapabilityRequirement is related to an OperationalCapability.

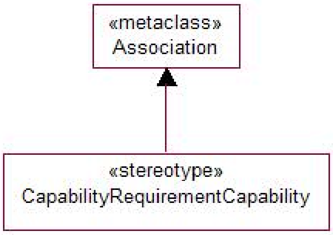


Figure 8.92 - CapabilityRequirementCapability

8.5.8.4 Attributes

No additional attributes.

8.5.8.5 Associations

No additional associations.

8.5.8.6 Constraints

[1] Asserts that a CapabilityRequirement is related to an OperationalCapability.

```
(self.endType->at(1).getAppliedStereotype('UPDM::CapabilityRequirement')-
>notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::OperationalCapability')-
>notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::CapabilityRequirement')-
>notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::OperationalCapability')-
>notEmpty())
```

8.5.8.7 Semantics

This association indicates that the strategic CapabilityRequirement is the specification of the metrics that measure the implementation of the OperationalCapability. Therefore, when this association is created, the CapabilityRequirement must be added to the list of subjects of the OperationalCapability and OperationalCapability added to the list of useCases of the CapabilityRequirement..

8.5.8.8 Notation

No additional notational requirements.

8.5.9 CausesEffect

8.5.9.1 Extension

- Transition (from UML2)

8.5.9.2 Generalizations

8.5.9.3 Description

A transition from one state to another Causes an Effect. The Effect is the call event that invokes an OperationalActivity on the receiving OperationalNode. The affected OperationalNode is the receiver of the InformationExchange while the causing OperationalNode is the originator of the InformationExchange. The change in state is the Effect of the InformationExchange. (See Effect).

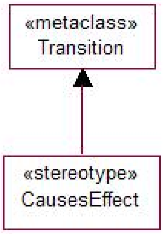


Figure 8.93 - CausesEffect

8.5.9.4 Attributes

No additional attributes.

8.5.9.5 Associations

No additional associations.

8.5.9.6 Constraints

[1] Asserts that the triggers of this transition are UPDM Triggers.

```
self.trigger.getAppliedStereotype('UPDM::Trigger')->notEmpty()
```

[2] Asserts that there is an Effect associated with the CausesEffect transition. The Effect is an OpaqueBehavior and can be elaborated as necessary by the modeler.

```
self.effect.getAppliedStereotype('UPDM::Effect')->notEmpty()
```

8.5.9.7 Semantics

No additional semantic requirements.

8.5.9.8 Notation

No additional notational requirements.

8.5.10 Effect

8.5.10.1 Extension

- OpaqueBehavior (from UML2)

8.5.10.2 Generalizations

8.5.10.3 Description

An Effect is an action that brings about a change in the state of something. The event that initiates the action is a Trigger. The action is the result of an OperationalTask. An Effect is caused by collaborating OperationalNodes that bring about the Effect as a result of their OperationalTasks. The collaboration, an OperationalCapabilityRealization, can be described in terms of the Effect that it has. This Effect can be observed in an EventTrace from the perspective of the sequence of

OperationalTasks. It can be seen in a StateTrace as a transition that CausesEffect showing the Trigger, the Effect, and the desired ResultOfEffect, the end state. The OperationalActivity can show the Events that are Triggered and the receiving Action that includes the OperationalTask in callOperationActions.

An Effect is the central player in all of these scenarios unifying the OperationalCapabilityRealization with the Effect, viewed from the perspective of the collaborating OperationalNodes, the Triggers, States, and Activities.

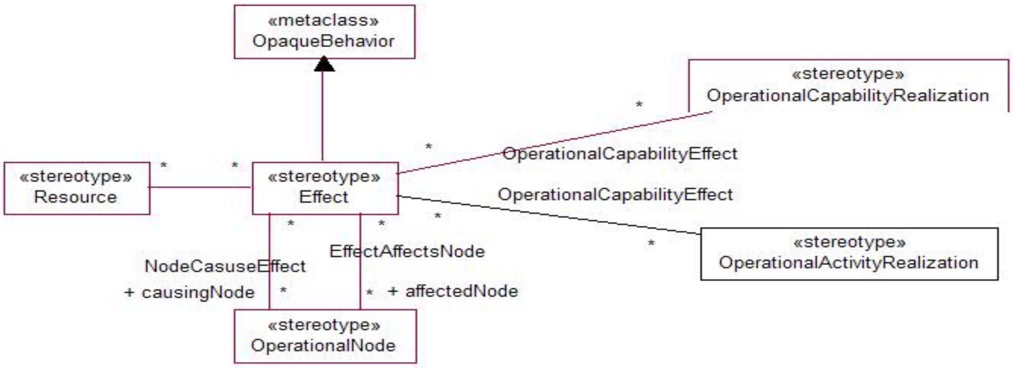


Figure 8.94 - Effect

8.5.10.4 Attributes

No additional attributes.

8.5.10.5 Associations

- affectedNode : OperationalNode [*]
  OperationalNodes affected by the Effect. See EffectAffectsNode for more details.
- causingNode : OperationalNode [*]
  OperationalNodes that cause this Effect. See NodeCausesEffect for more details.
- operationalActivityRealization : OperationalActivityRealization [*]
  OperationalActivityRealizations that realize the OperationalCapabilities that provide the Effect
- operationalCapabilityRealization : OperationalCapabilityRealization [*]
  OperationalCapabilityRealizations that provide this Effect
- resources : Resource [*]
  The Resource affected by this Effect

8.5.10.6 Constraints

[1] Asserts that the operation of the Trigger that is specified on the transition of this Effect is an OperationalTask or SystemTask

```
self.owner.oclAsType(uml::Transition).trigger.event.oclAsType(uml::CallEvent).oper
ation.getAppliedStereotype('UPDM::OperationalTask')->notEmpty() or
```

```
self.owner.oclAsType(uml::Transition).trigger.event.oclAsType(uml::CallEvent).oper
ation.getAppliedStereotype('UPDM::SystemTask')->notEmpty()
```

[2] Asserts that there are Resources affected by this Effect.

```
self.resources->forAll(getAppliedStereotype('UPDM::Resource')->notEmpty() or
getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::Resource') ->notEmpty())
```

[3] Asserts that this Effect is associated with the Capability that provides it.

```
self.getAllAttributes()->asOrderedSet()->select(association->
notEmpty()).association->
any(getAppliedStereotype('UPDM::OperationalCapabilityEffect')-> notEmpty())->
notEmpty()
```

8.5.10.7 Semantics

The causing OperationalNode that is the sender of the message (TaskInvocation) that invokes the OperationalTask and the affected OperationalNode can be derived by examining the OperationalCapabilityRealization. These are guaranteed to be OperationalNodes since the OperationalCapabilityRealization is a collaboration among OperationalNodes.

An Effect is an OpaqueBehavior, and therefore a behavior with implementation-specific semantics.

8.5.10.8 Notation

No additional notational requirements.

8.5.11 EffectAffectsNode

8.5.11.1 Extension

- Association (from UML2)

8.5.11.2 Generalizations

8.5.11.3 Description

EffectAffectsNode specifies that this Effect affects zero or more OperationalNodes and that this OperationalNode is affected by zero or more of this Effect.

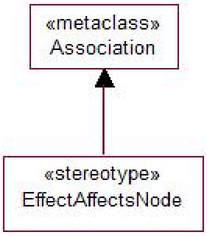


Figure 8.95 - EffectAffectsNode

8.5.11.4 Attributes

No additional attributes.

8.5.11.5 Associations

No additional associations.

8.5.11.6 Constraints

[1] Asserts that one end is an Effect and the other is an OperationalNode or one of its specializations.

```
(self.endType-> at(1).getAppliedStereotype('UPDM::Effect')->notEmpty()and

(self.endType->at(2).getAppliedStereotype('UPDM::OperationalNode')-> notEmpty()or

self.endType->at(2).getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty()))

 or

(self.endType->at(2).getAppliedStereotype('UPDM::Effect')->notEmpty() and

(self.endType->at(1).getAppliedStereotype('UPDM::OperationalNode')-> notEmpty()or

self.endType->at(1).getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty()))
```

[2] Asserts that the multiplicity of both ends is *

```
self.memberEnd.upper->forAll(a|a = -1) and self.memberEnd.lower->forAll(a|a = 0)
```

8.5.11.7 Semantics

No additional semantic requirements.

8.5.11.8 Notation

No additional notational requirements.

8.5.12 NodeCausesEffect

8.5.12.1 Extension

- Association (from UML2)

8.5.12.2 Generalizations

8.5.12.3 Description

NodeCausesEffect specifies that this Effect is caused by zero or more OperationalNodes and that this OperationalNode causes zero or more of this Effect.

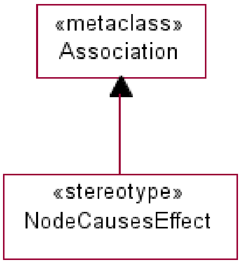


Figure 8.96 - NodeCausesEffect

8.5.12.4 Attributes

No additional attributes.

8.5.12.5 Associations

No additional associations.

8.5.12.6 Constraints

[1] Asserts that one end is an Effect and the other is an OperationalNode or one of its specializations

```
(self.endType-> at(1).getAppliedStereotype('UPDM::Effect')->notEmpty() and

(self.endType-> at(2).getAppliedStereotype('UPDM::OperationalNode')-> notEmpty()or

self.endType->at(2).getAppliedStereotypes()-> collect(allParents())->
any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty()))

 or

(self.endType-> at(2).getAppliedStereotype('UPDM::Effect')->notEmpty() and

(self.endType-> at(1).getAppliedStereotype('UPDM::OperationalNode')-> notEmpty()or

self.endType->at(1).getAppliedStereotypes()-> collect(allParents())->
any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty()))
```

[2] Asserts that the multiplicity of both ends is *

```
self.memberEnd.upper->forAll(a|a = -1) and self.memberEnd.lower->forAll(a|a = 0)
```

8.5.12.7 Semantics

No additional semantic requirements.

8.5.12.8 Notation

No additional notational requirements.

8.5.13 OperationalCapabilityEffect

8.5.13.1 Extension

- Association (from UML2)

8.5.13.2 Generalizations

8.5.13.3 Description

Specifies that an OperationalCapabilityRealization delivers the Effect, or that an OperationalActivityRealization realizes an Effect.

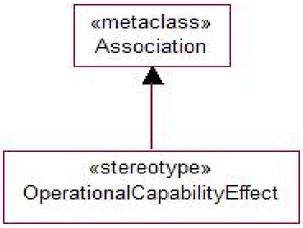


Figure 8.97 - OperationalCapabilityEffect

8.5.13.4 Attributes

No additional attributes.

8.5.13.5 Associations

No additional associations.

8.5.13.6 Constraints

[1] Asserts that the OperationalCapabilityRealization delivers the Effect, or that an OperationalActivityRealization realizes an Effect and that these are the elements associated with this association

```
((self.endType-
>at(1).getAppliedStereotype('UPDM::OperationalCapabilityRealization')->notEmpty()
or

self.endType->at(1).getAppliedStereotype('UPDM::OperationalActivityRealization')-
>notEmpty())and

self.endType->at(2).getAppliedStereotype('UPDM::Effect')->notEmpty()) or

((self.endType-
>at(2).getAppliedStereotype('UPDM::OperationalCapabilityRealization')->notEmpty()
or

self.endType->at(2).getAppliedStereotype('UPDM::OperationalActivityRealization')-
>notEmpty()) and
self.endType->at(1).getAppliedStereotype('UPDM::Effect')->notEmpty())
```

8.5.13.7 Semantics

No additional semantic requirements.

8.5.13.8 Notation

No additional notational requirements.

8.5.14 OperationalNodeCapabilityRequirement

8.5.14.1 Extension

- Association (from UML2)

8.5.14.2 Generalizations

8.5.14.3 Description

Asserts that a Capability is required by an OperationalNode.

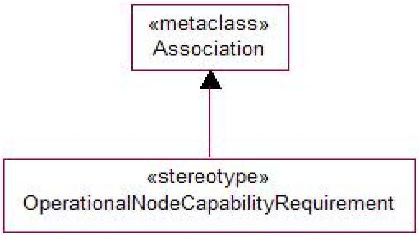


Figure 8.98 - OperationalNodeCapabilityRequirement

8.5.14.4 Attributes

No additional attributes.

8.5.14.5 Associations

No additional associations.

8.5.14.6 Constraints

[1] Asserts that a CapabilityRequirement is related to an OperationalNode or one of its specializations.

```
(self.endType-> at(1).getAppliedStereotype('UPDM::CapabilityRequirement')->notEmpty() and

(self.endType-> at(2).getAppliedStereotype('UPDM::OperationalNode')-> notEmpty()or

self.endType->at(2).getAppliedStereotypes()-> collect(allParents())->any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty()))
```

```
 or
(self.endType-> at(2).getAppliedStereotype('UPDM::CapabilityRequirement')-
>notEmpty() and
(self.endType-> at(1).getAppliedStereotype('UPDM::OperationalNode')-> notEmpty()or
self.endType->at(1).getAppliedStereotypes()-> collect(allParents())-
>any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty()))
```

8.5.14.7 Semantics

No additional semantic requirements.

8.5.14.8 Notation

No additional notational requirements.

8.5.15 OrganizationalResourceCapabilityConfiguration

8.5.15.1 Extension

- Association (from UML2)

8.5.15.2 Generalizations

8.5.15.3 Description

Defines the association between a CapabilityConfiguration and the OrganizationResources that are configured by it.

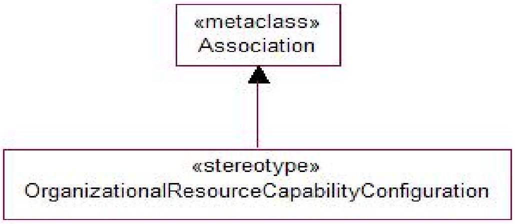


Figure 8.99 - OrganizationResourceCapabilityConfiguration

8.5.15.4 Attributes

No additional attributes.

8.5.15.5 Associations

No additional associations.

8.5.15.6 Constraints

[1] Asserts that there are zero or more OrganizationalResources configured by the CapabilityConfiguration and that these are the elements associated with this association

```
(self.endType->at(1).getAppliedStereotype('UPDM::CapabilityConfiguration')-
>notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::OrganizationalResource')-
>notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::CapabilityConfiguration')-
>notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::OrganizationalResource')-
>notEmpty())
```

8.5.15.7 Semantics

To model the inclusion of OrganizationalResources in CapabilityConfiguration, draw an association from a resource to a CapabiltyConfiguration and stereotype it as an OrganizationalResourceCapabilityConfiguration, giving the role name of the resource in the Configuration as appropriate, and the role name of the Configuration in the resource as “usedInCapabilityConfiguration.”

8.5.15.8 Notation

No additional notational requirements.

8.5.16 ResourceCapability

8.5.16.1 Extension

- Association (from UML2)

8.5.16.2 Generalizations

8.5.16.3 Description

Specifies an association exists between a Resource and the Capabilities that it provides, that is, the Resources required for a Capability.

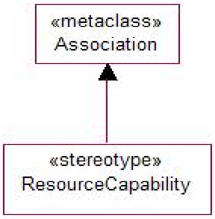


Figure 8.100 - ResourceCapability

8.5.16.4 Attributes

No additional attributes.

8.5.16.5 Associations

No additional associations.

8.5.16.6 Constraints

[1] Asserts that a ResourceCapability association exists between a Resource and the Capabilities that it provides and that these are the elements associated with this association

```
(self.endType-> at(1).owner.getAppliedStereotype('UPDM::Capability')-> notEmpty()
and

self.endType-> at(2).owner.getAppliedStereotype('UPDM::Resource')-> notEmpty()or

self.base_Association.endType-> at(2).getAppliedStereotypes()-
>collect(allParents())-> any(qualifiedName = 'UPDM::Resource') ->notEmpty() ) or

(self.endType-> at(2).owner.getAppliedStereotype('UPDM::Capability')-> notEmpty()
and

self.endType-> at(1).owner.getAppliedStereotype('UPDM::Resource')-> notEmpty()or

self.base_Association.endType-> at(1).getAppliedStereotypes()-
>collect(allParents())-> any(qualifiedName = 'UPDM::Resource') ->notEmpty())
```

8.5.16.7 Semantics

No additional semantic requirements.

8.5.16.8 Notation

No additional notational requirements.

8.5.17 ResourceCapabilityConfiguration

8.5.17.1 Extension

- Association (from UML2)

8.5.17.2 Generalizations

8.5.17.3 Description

Defines the association between a CapabilityConfiguration and the all of Resources that are included in it.

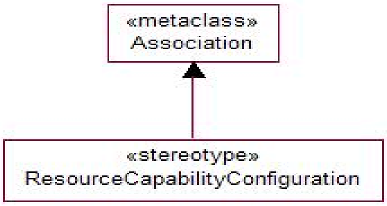


Figure 8.101 - ResourceCapabilityConfiguration

8.5.17.4 Attributes

No additional attributes.

8.5.17.5 Associations

No additional associations.

8.5.17.6 Constraints

[1] Asserts that there are zero or more Resources included in the CapabilityConfiguration and that these are the elements associated with this association

```
(self.endType->at(1).getAppliedStereotype('UPDM::CapabilityConfiguration')-
>notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::Resource')->notEmpty()or

self.base_Association.endType->at(2).getAppliedStereotypes()-
>collect(allParents())->any(qualifiedName = 'UPDM::Resource') ->notEmpty() ) or

(self.endType->at(2).getAppliedStereotype('UPDM::CapabilityConfiguration')-
>notEmpty() and

self.endType-> at(1).getAppliedStereotype('UPDM::Resource')-> notEmpty()or

self.base_Association.endType-> at(1).getAppliedStereotypes()-
>collect(allParents())-> any(qualifiedName = 'UPDM::Resource') ->notEmpty() )
```

8.5.17.7 Semantics

No additional semantic requirements.

8.5.17.8 Notation

No additional notational requirements.

8.5.18 StrategicView

8.5.18.1 Extension

8.5.18.2 Generalizations

- ArchitectureView (from AllViews)

8.5.18.3 Description

The StrategicView provides an overall enterprise architecture assessment of the strategic Capabilities and their relationships. While an Enterprise will have a number of UPDM ArchitectureDescriptions that have the Operational, System, Technical Standards, and All Views, only one StrategicView will exist across a number of ArchitectureDescripions. In the US, the function of the StrategicView is often provided through JCIDS, the Joint Capabilities Integration Development System.

The StrategicView is defined in MODAF and is not a part of the DoDAF. However, the elements in the StrategicView relate to elements in the Operational and Strategic Views in order to deliver the needed diagrams and matrices.

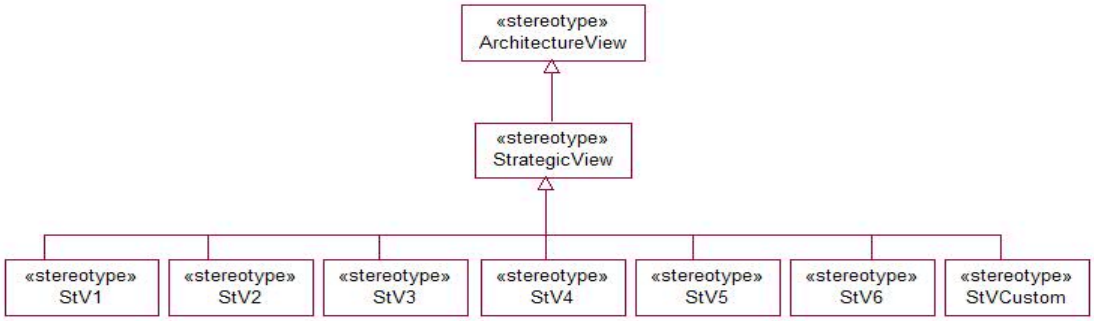


Figure 8.102 - StrategicView

8.5.18.4 Attributes

No additional attributes.

8.5.18.5 Associations

No additional associations.

8.5.18.6 Constraints

No additional constraints.

8.5.18.7 Semantics

Typically, the StrategicView will apply to a number of different architectural descriptions.

8.5.18.8 Notation

No additional notational requirements.

8.5.19 StV-1: Capability Vision

8.5.19.1 Extension

8.5.19.2 Generalizations

- StrategicView (from StrategicView)

8.5.19.3 Description

Capability Vision - Outlines the vision for a capability area over a particular time frame.

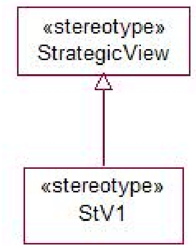


Figure 8.103 - StV-1: Capability Vision

8.5.19.4 Attributes

No additional attributes.

8.5.19.5 Associations

No additional associations.

8.5.19.6 Constraints

No additional constraints.

8.5.19.7 Semantics

No additional semantic requirements.

8.5.19.8 Notation

No additional notational requirements.

8.5.20 StV-2: Capability Taxonomy

8.5.20.1 Extension

8.5.20.2 Generalizations

- StrategicView (from StrategicView)

8.5.20.3 Description

Capability Taxonomy - Provides a structured list of capabilities and sub-capabilities (known as capability functions) that are required within a capability area during a certain period of time.

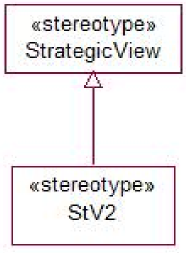


Figure 8.104 - StV-2: Capability Taxonomy

8.5.20.4 Attributes

No additional attributes.

8.5.20.5 Associations

No additional associations.

8.5.20.6 Constraints

No additional constraints.

8.5.20.7 Semantics

No additional semantic requirements.

8.5.20.8 Notation

No additional notational requirements.

8.5.21 StV-3: Capability Phasing

8.5.21.1 Extension

8.5.21.2 Generalizations

- StrategicView (from StrategicView)

8.5.21.3 Description

Capability phasing - Captures the planned availability of capability at different points in time.

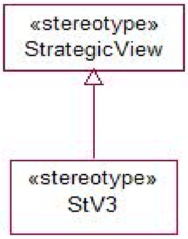


Figure 8.105 - StV-3: Capability Phasing

8.5.21.4 Attributes

No additional attributes.

8.5.21.5 Associations

No additional associations.

8.5.21.6 Constraints

No additional constraints.

8.5.21.7 Semantics

No additional semantic requirements.

8.5.21.8 Notation

No additional notational requirements.

8.5.22 StV-4: Capability Clusters

8.5.22.1 Extension

8.5.22.2 Generalizations

- StrategicView (from StrategicView)

8.5.22.3 Description

Capability clusters - Provides a means of analyzing the main dependencies between capabilities.

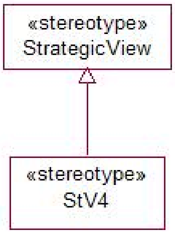


Figure 8.106 - StV-4: Capability Clusters

8.5.22.4 Attributes

No additional attributes.

8.5.22.5 Associations

No additional associations.

8.5.22.6 Constraints

No additional constraints.

8.5.22.7 Semantics

No additional semantic requirements.

8.5.22.8 Notation

No additional notational requirements.

8.5.23 StV-5: Capability to Systems Deployment Mapping

8.5.23.1 Extension

8.5.23.2 Generalizations

- StrategicView (from StrategicView)

8.5.23.3 Description

Capability to Systems Deployment Mapping - Shows the planned capability deployment as systems, equipment, training, etc. and their interconnection by organization / period of time.

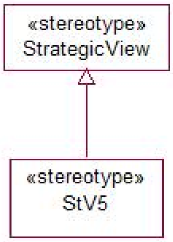


Figure 8.107 - StV-5: Capability to Systems Deployment Mapping

8.5.23.4 Attributes

No additional attributes.

8.5.23.5 Associations

No additional associations.

8.5.23.6 Constraints

No additional constraints.

8.5.23.7 Semantics

No additional semantic requirements.

8.5.23.8 Notation

No additional notational requirements.

8.5.24 StV-6: Capability Function to Operational Mapping

8.5.24.1 Extension

8.5.24.2 Generalizations

- StrategicView (from StrategicView)

8.5.24.3 Description

Capability Function to Operational Mapping - Describes the mapping between capability elements and operational activities that can be performed by using them and thereby provides a link between capability analysis and activity analysis.

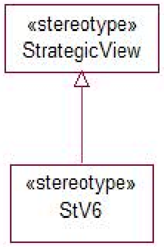


Figure 8.108 - StV-6: Capability Function to Operational Mapping

8.5.24.4 Attributes

No additional attributes.

8.5.24.5 Associations

No additional associations.

8.5.24.6 Constraints

No additional constraints.

8.5.24.7 Semantics

No additional semantic requirements.

8.5.24.8 Notation

No additional notational requirements.

8.5.25 StVCustom: Custom Strategic View

8.5.25.1 Extension

8.5.25.2 Generalizations

- StrategicView (from StrategicView)

8.5.25.3 Description

A user-defined view that applies to the StrategicView.

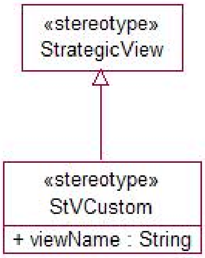


Figure 8.109 - StVCustom: Custom Strategic View

8.5.25.4 Attributes

- viewName : String [1]
  Provides a name for the custom view so that it can be easily distinguished from other custom StrategicViews.

8.5.25.5 Associations

No additional associations.

8.5.25.6 Constraints

No additional constraints.

8.5.25.7 Semantics

No additional semantic requirements.

8.5.25.8 Notation

No additional notational requirements.

8.6 SystemsView Package

8.6.1 Overview

The SystemsView Package contains UML stereotypes that assist the modeler in developing the views defined in the SystemsView viewpoint. These stereotypes support the description of systems and interconnections providing for, or supporting, DoD functions. The SystemsView Package also includes elements to identify the systems resources that support the operational activities and facilitate the exchange of information among operational nodes.

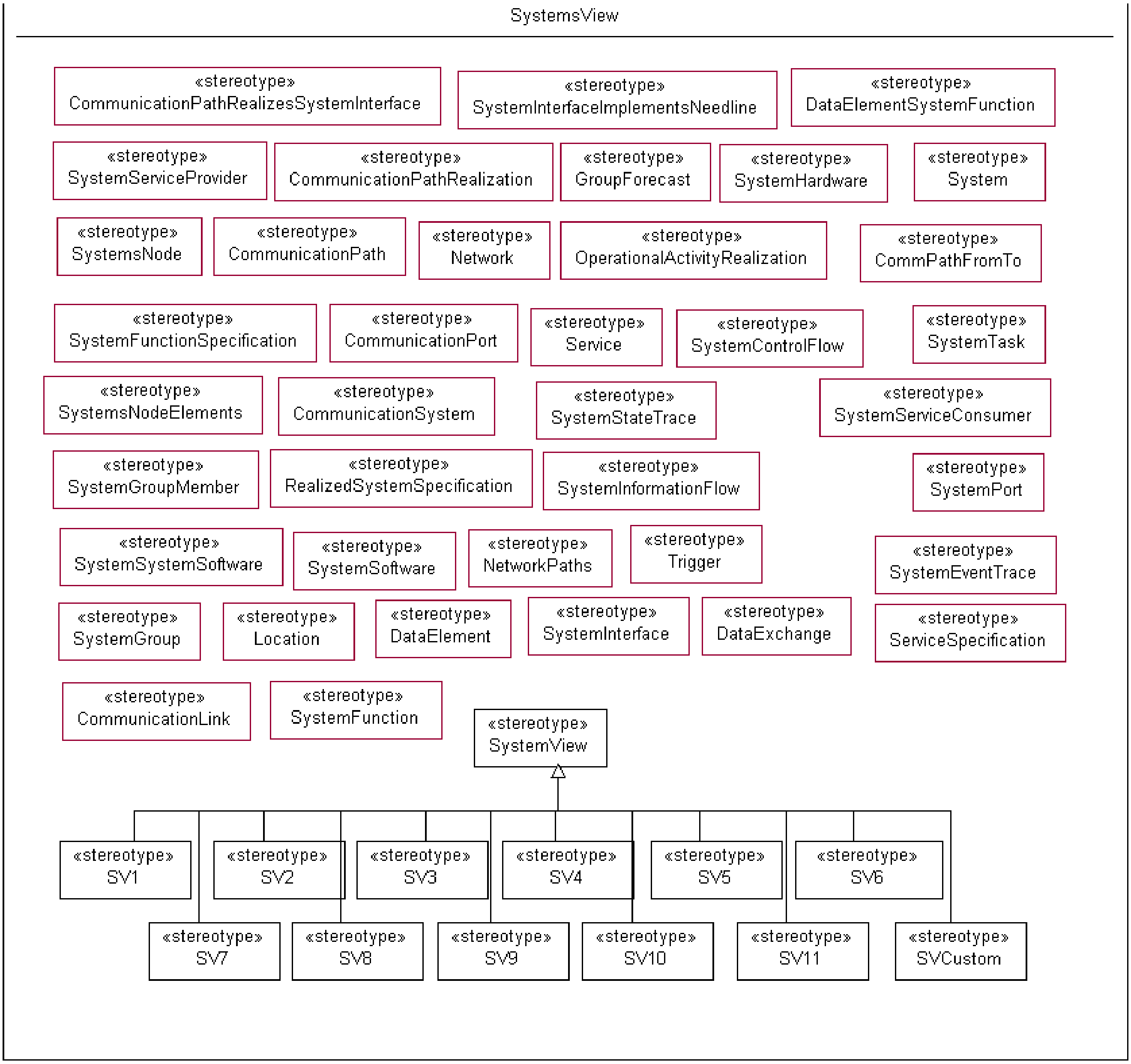

SystemsView
«stereotype» CommunicationPathRealizesSystemInterface
«stereotype» SystemInterfaceImplementsNeedline
«stereotype» DataElementSystemFunction
«stereotype» SystemServiceProvider
«stereotype» CommunicationPathRealization
«stereotype» GroupForecast
«stereotype» SystemHardware
«stereotype» System
«stereotype» SystemsNode
«stereotype» CommunicationPath
«stereotype» Network
«stereotype» OperationalActivityRealization
«stereotype» CommPathFromTo
«stereotype» SystemFunctionSpecification
«stereotype» CommunicationPort
«stereotype» Service
«stereotype» SystemControlFlow
«stereotype» SystemTask
«stereotype» SystemsNodeElements
«stereotype» CommunicationSystem
«stereotype» SystemStateTrace
«stereotype» SystemServiceConsumer
«stereotype» SystemGroupMember
«stereotype» RealizedSystemSpecification
«stereotype» SystemInformationFlow
«stereotype» SystemPort
«stereotype» SystemSystemSoftware
«stereotype» SystemSoftware
«stereotype» NetworkPaths
«stereotype» Trigger
«stereotype» SystemEventTrace
«stereotype» SystemGroup
«stereotype» Location
«stereotype» DataElement
«stereotype» SystemInterface
«stereotype» DataExchange
«stereotype» ServiceSpecification
«stereotype» CommunicationLink
«stereotype» SystemFunction
«stereotype» SystemView
«stereotype» SV1
«stereotype» SV2
«stereotype» SV3
«stereotype» SV4
«stereotype» SV5
«stereotype» SV6
«stereotype» SV7
«stereotype» SV8
«stereotype» SV9
«stereotype» SV10
«stereotype» SV11
«stereotype» SVCustom

Figure 8.110 - SystemsView Package

8.6.2 CommPathFromTo

8.6.2.1 Extension

- Association (from UML2)

8.6.2.2 Generalizations

8.6.2.3 Description

Asserts that there is an association between a CommunicationPath and a System.

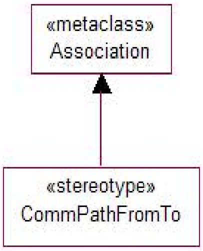


Figure 8.111 - CommPathFromTo

8.6.2.4 Attributes

No additional attributes.

8.6.2.5 Associations

No additional associations.

8.6.2.6 Constraints

[1] Asserts that this association is between a CommunicationPath and a System.

```
let e1:uml::Class = self.oclAsType(uml::Association).endType->asOrderedSet()-
>at(1).oclAsType(uml::Class) in

let e2:uml::Class = self.oclAsType(uml::Association).endType->asOrderedSet()-
>at(2).oclAsType(uml::Class) in

(e1.getAppliedStereotype('UPDM::CommunicationPath')->notEmpty() and

e2.getAppliedStereotype('UPDM::System')->notEmpty())

or

(e2.getAppliedStereotype('UPDM::CommunicationPath')->notEmpty() and

e1.getAppliedStereotype('UPDM::System')->notEmpty())
```

8.6.2.7 Semantics

No additional semantic requirements.

8.6.2.8 Notation

No additional notational requirements.

8.6.3 CommunicationLink

8.6.3.1 Extension

- Association (from UML2)
- Connector (from UML2)

8.6.3.2 Generalizations

8.6.3.3 Description

A communications link is a single physical connection from one system (or node) to another. A communications path is a (connected) sequence of communications systems and communications links originating from one system (or node) and terminating at another systems (or node).

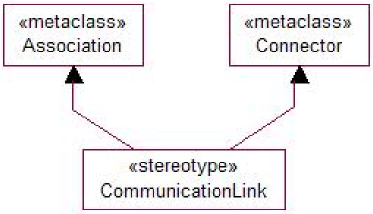


Figure 8.112 - CommunicationLink

8.6.3.4 Attributes

No additional attributes.

8.6.3.5 Associations

No additional associations.

8.6.3.6 Constraints

[1] Asserts that if this CommunicationLink is an association then it connects two Systems.

```
 if self.oclIsTypeOf(uml::Association) then

self.oclAsType(uml::Association) .endType-
>forAll(getAppliedStereotype('UPDM::System')->notEmpty())

 else

 true

 endif
```

[2] Asserts that when CommunicationLink is modeled with a Connector, then both ends must connect CommunicationPorts, or both ends must connect Systems.

```
 if self.oclIsTypeOf(uml::Connector) then
```

```
self.oclAsType(uml::Connector).end->forAll(
(role.oclIsTypeOf(uml::Port) and
role.getAppliedStereotype('UPDM::CommunicationPort')->notEmpty()) or
(role.oclIsTypeOf(uml::Property) and
role.type.getAppliedStereotype('UPDM::System')->notEmpty()))
else
true
endif
```

8.6.3.7 Semantics

A CommunicationLink can be drawn as an association between two Systems when no further detail is required, or when the description of CommunicationPaths and Networks is not required. This may be done early in a modeling effort when a Link is known but its relationship to other links is not.

A CommunicationLink can be modeled as a Connector linking systems or ports in a composite structure diagram. The composite structure is a CommunicationSystem.

Modeling the series of links that make up a CommunicationSystem shows their sequence of connections. It also allows the modeler to include individual ports and their details, such as protocols and interfaces. The connectors can be used to connect CommunicationPorts, individual Systems, or other CommunicationSystems.

The CommunicationLinks that are modeled as a connector can then be typed with the CommunicationLink that is modeled as an association, enabling linking the connector detail to the association.

To model the CommunicationLinks that comprise CommunicationPaths, associate composite CommunicationSystems that contain the sequence of systems linked with connectors with the CommunicationPath. This allows the modeler to specify the exact sequence of links that comprise the CommunicationPath. By associating more than one CommunicationSystem to a CommunicationPath, multiple possible routes for the path can be shown.

8.6.3.8 Notation

No additional notational requirements.

8.6.4 CommunicationPath

8.6.4.1 Extension

8.6.4.2 Generalizations

- ConformingElement (from AllViews)

8.6.4.3 Description

A CommunicationPath is a (connected) sequence of CommunicationSystems and CommunicationsLinks originating from one System and terminating at another System. A communications Network may contain multiple CommunicationPaths between the same pair of systems. (See also Network)

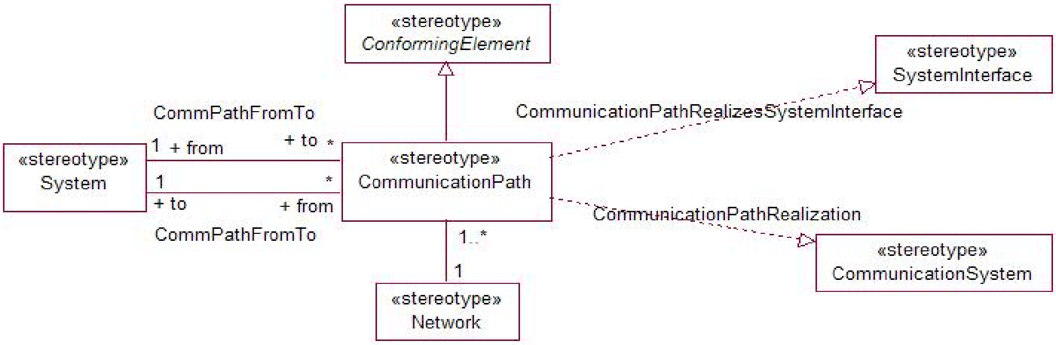


Figure 8.113 - CommunicationPath

8.6.4.4 Attributes

No additional attributes.

8.6.4.5 Associations

• from : System [1]	Start point of the CommunicationPath
• network : Network [1]	A collection of CommunicationPaths
• to : System [1]	End point of the CommunicationPath

8.6.4.6 Constraints

[1] Asserts that the CommuncationPath realizes one or more SystemInterfaces

```
self.oclAsType(uml::Class).clientDependency->

exists(getAppliedStereotype('UPDM::CommunicationPathRealizesSystemInterface')-
>notEmpty())
```

[2] Asserts that the CommunicationPath connects two Systems

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->select

  (getAppliedStereotype('UPDM::CommPathFromTo')-> notEmpty())->size()=2
```

[3] Asserts that the CommunicationPath is associated with a Network.

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->select

  (getAppliedStereotype('UPDM::NetworkPaths')-> notEmpty())->notEmpty()
```

[4] Asserts that there is an association that defines this CommunicationPath in terms of CommunicationsSystems and CommunicationLinks

```
self.oclAsType(uml::Class).clientDependency->

exists(getAppliedStereotype('UPDM::CommunicationPathRealization')->notEmpty())
```

8.6.4.7 Semantics

No additional semantic requirements.

8.6.4.8 Notation

No additional notational requirements.

8.6.5 CommunicationPathRealization

8.6.5.1 Extension

- Realization (from UML2)

8.6.5.2 Generalizations

8.6.5.3 Description

Asserts that the CommunicationPath is realized by a CommunicationsSystem described in terms of CommunicationsSystems and Systems connected by CommunicationLinks.

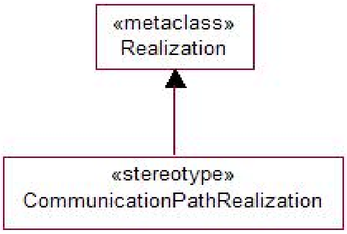


Figure 8.114 - CommunicationPathRealization

8.6.5.4 Attributes

No additional attributes.

8.6.5.5 Associations

No additional associations.

8.6.5.6 Constraints

[1] Asserts that the supplier is a CommunicationSystem and the client is a CommunicationPath

```
self.supplier.getAppliedStereotype('UPDM::CommunicationSystem')->notEmpty() and

self.client.getAppliedStereotype('UPDM::CommunicationPath')->notEmpty()
```

8.6.5.7 Semantics

No additional semantic requirements.

8.6.5.8 Notation

No additional notational requirements.

8.6.6 CommunicationPathRealizesSystemInterface

8.6.6.1 Extension

- Realization (from UML2)

8.6.6.2 Generalizations

8.6.6.3 Description

Asserts that a CommunicationPath realizes a SystemInterface.

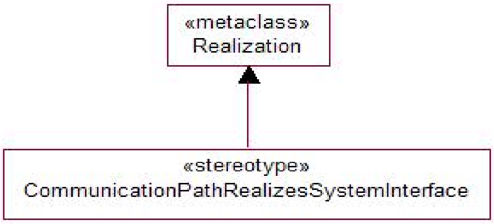


Figure 8.115 - CommunicationPathRealizesSystemInterface

8.6.6.4 Attributes

No additional attributes.

8.6.6.5 Associations

No additional associations.

8.6.6.6 Constraints

[1] Asserts that a CommunicationPath Realizes a SystemInterface

```
self.target.getAppliedStereotype('UPDM::SystemInterface')->notEmpty() and
self.source.getAppliedStereotype('UPDM::CommunicationPath')->notEmpty()
```

8.6.6.7 Semantics

No additional semantic requirements.

8.6.6.8 Notation

No additional notational requirements.

8.6.7 CommunicationPort

8.6.7.1 Extension

- Port (from UML2)

8.6.7.2 Generalizations

8.6.7.3 Description

A CommunicationPort occurs at the end of CommunicationLinks when it is necessary to provide details on the actual connection. CommunicationPort may implement a PortType, though there is no requirement to be typed.

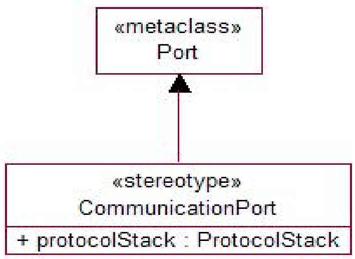


Figure 8.116 - CommunicationPort

8.6.7.4 Attributes

- protocolStack : ProtocolStack [1]
  A set of information that describes the protocol implemented by the CommunicationPort

8.6.7.5 Associations

No additional associations.

8.6.7.6 Constraints

[1] Asserts that the end of a CommunicationPort are CommunicationLinks - connectors.

```
self.oclAsType(uml::Port).end.getAppliedStereotype('UPDM::CommunicationLink')-
>notEmpty()
```

[2] Asserts that the owner or class of the SystemPort is a System or one of its specializations.

```
let x:uml::Class = self.owner.oclAsType(uml::Port).class in

x.getAppliedStereotype('UPDM::System') ->notEmpty() or

x.getAppliedStereotypes()->collect(allParents())-> any(qualifiedName =
'UPDM::System') ->notEmpty())
```

[3] Asserts that the protocolStack is an instance of a ProtocolStack

```
self.protocolStack->forAll(classifier->forAll(name='ProtocolStack')) or
```

```
self.protocolStack->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='ProtocolStack')))
```

8.6.7.7 Semantics

No additional semantic requirements.

8.6.7.8 Notation

No additional notational requirements.

8.6.8 CommunicationSystem

8.6.8.1 Extension

8.6.8.2 Generalizations

- System (from SystemsView)

8.6.8.3 Description

Communications systems (e.g., switches, routers, and communications satellites) are systems whose primary function is to control the transfer and movement of system data as opposed to performing mission application processing.

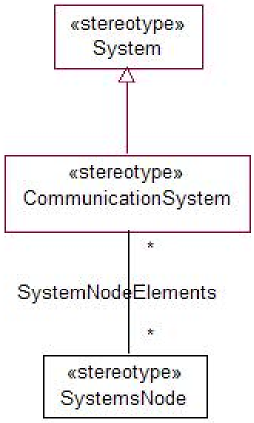


Figure 8.117 - CommunicationSystem

8.6.8.4 Attributes

No additional attributes.

8.6.8.5 Associations

- systemsnode : SystemsNode [*]

8.6.8.6 Constraints

[1] Asserts that this CommunicationSystem has parts that are Systems or CommunicationSystems and that there are CommunicationLinks connecting them.

```
self.attribute->exists(s|s.type.getAppliedStereotype('UPDM::System')->notEmpty()or

s.type.getAppliedStereotype('UPDM::CommunicationsSystem')->notEmpty())and

self.member->exists(c|c.oclIsTypeOf(uml::Connector) and
c.getAppliedStereotype('UPDM::CommunicationLink')->notEmpty())
```

8.6.8.7 Semantics

Use CommunicationSystem as a structure to create sequences of CommunicationLinks joined together to form a CommunicationPath. (See CommunicationPath)

8.6.8.8 Notation

No additional notational requirements.

8.6.9 DataElement

8.6.9.1 Extension

- Class (from UML2)

8.6.9.2 Generalizations

8.6.9.3 Description

A data element exchanged in between systems.

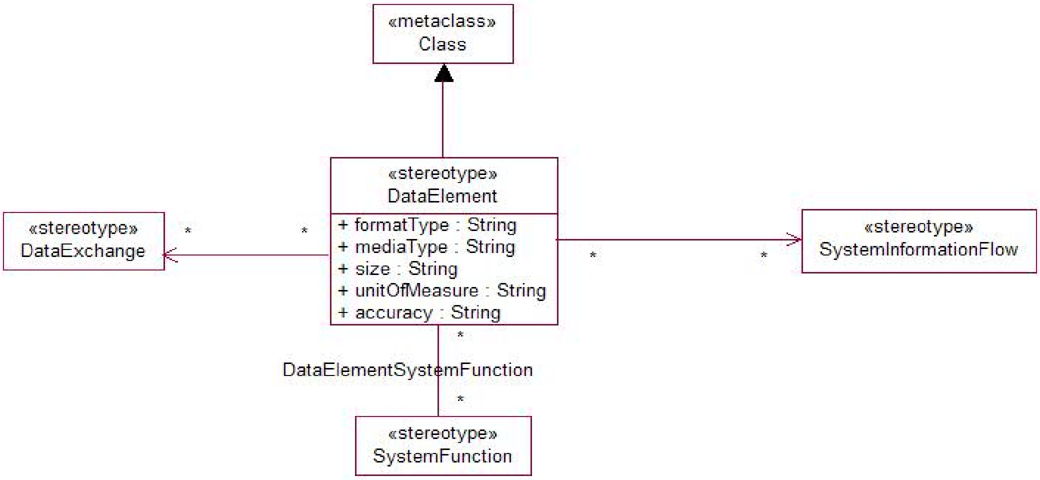


Figure 8.118 - DataElement

8.6.9.4 Attributes

- accuracy : String [1]
  Accuracy requirement for the data element

- formatType : String [1]
  The format for the data element.

- mediaType : String [1]
  The media for the data.

- size : String [1]
  Size of the data element.

- unitOfMeasure : String [1]
  Unit used to measure size.

8.6.9.5 Associations

- dataExchange : DataExchange [*]
  The DataExchanges that move these DataElements

- systemFunction : SystemFunction [*]
  The SystemFunctions that initiate the SystemInformationFlows along which the DataElement flows. See DataElementSystemFunction for more details.

- systemInformationFlow : SystemInformationFlow [*]
  The SystemInformationFlows along which the DataElement flows

8.6.9.6 Constraints

[1] Asserts that there are zero or more DataExchanges that utilize this DataElement

```
self.dataExchange->forAll(getAppliedStereotype('UPDM::DataExchange')->notEmpty())
```

[2] Asserts that a DataElementSystemFunction association exists between DataElement and SystemFunction.

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::DataElementSystemFunction')-> notEmpty())-
>notEmpty()
```

[3] Asserts that this DataElement is associated with zero or more SystemInformationFlows

```
self.systemInformationFlow.getAppliedStereotype('UPDM::SystemInformationFlow')-
>notEmpty()
```

8.6.9.7 Semantics

The DataElement indicates the type and structure of the information transferred between Systems represented by SystemInformationFlows.

8.6.9.8 Notation

No additional notational requirements.

8.6.10 DataElementSystemFunction

8.6.10.1 Extension

- Association (from UML2)

8.6.10.2 Generalizations

8.6.10.3 Description

Defines the association between a SystemFunction and the DataElements that are the result of the execution of its SystemTasks or the DataElements that are used by those SystemTasks.

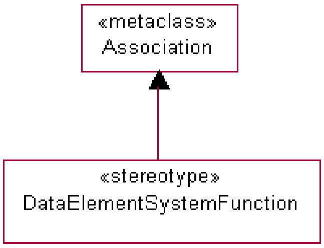


Figure 8.119 - DataElementSystemFunction

8.6.10.4 Attributes

No additional attributes.

8.6.10.5 Associations

No additional associations.

8.6.10.6 Constraints

[1] Asserts that there are zero or more DataElements that are involved in the SystemFunction and that these are the elements associated with this association

```
(self.endType->at(1).getAppliedStereotype('UPDM::SystemFunction')->notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::DataElement')->notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::SystemFunction')->notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::DataElement')->notEmpty())
```

8.6.10.7 Semantics

No additional semantic requirements.

8.6.10.8 Notation

No additional notational requirements.

8.6.11 DataExchange

8.6.11.1 Extension

- InformationFlow (from UML2)

8.6.11.2 Generalizations

8.6.11.3 Description

An exchange that can carry information and data. It must be between Systems or SystemFunctionSpecifications.

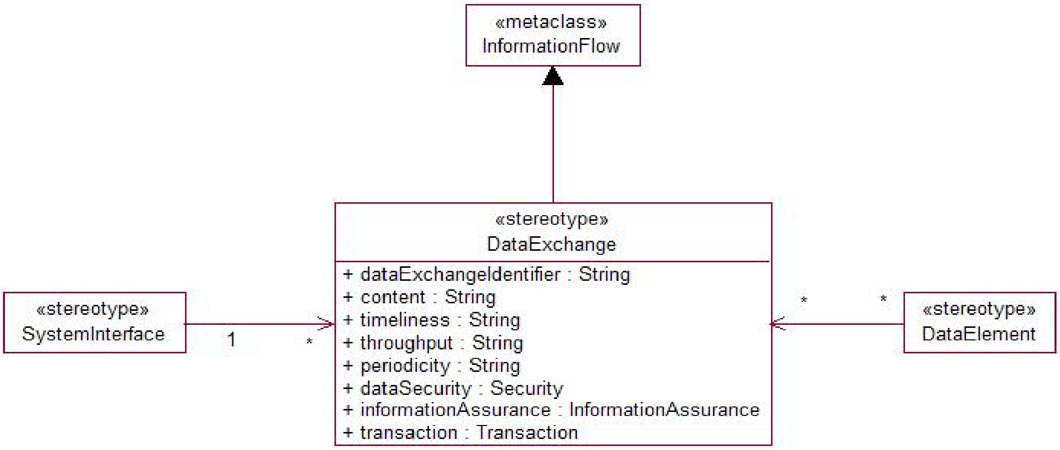


Figure 8.120 - DataExchange

8.6.11.4 Attributes

- content : String [1]
  The content of the flow.

- dataExchangeIdentifier : String [1]
  A user defined identifier - a primary key

- dataSecurity : Security [1]
  Security constraints that govern the DataExchange

- informationAssurance : InformationAssurance [1]
  InformationAssurance characteristics that govern the DataExchange

- periodicity : String [1]
  The update pace for the DataFlow.

- throughput : String [1]
  Amount of information that can travel on the flow.

- timeliness : String [1]
  Time since occurrence.

- transaction : Transaction [1]
  A Transaction that controls the DataExchange

8.6.11.5 Associations

No additional associations.

8.6.11.6 Constraints

[1] Asserts that the resulting value of the DataExchange is a DataElement

```
self.conveyed.oclAsType(uml::Classifier).getAppliedStereotype('UPDM::DataElement')
->notEmpty()
```

[2] Asserts that there is a SystemInterface along which the DataExchange occurs

```
self.realizingConnector->notEmpty() and

self.realizingConnector->forAll(getAppliedStereotype('UPDM::SystemInterface')-
>notEmpty() or

    type.getAppliedStereotype('UPDM::SystemInterface')->notEmpty()

)
```

[3] Asserts that there is InformationAssurance constraints on the exchange

```
self.informationAssurance->forAll(classifier->forAll(name='InformationAssurance'))
or

self.informationAssurance->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='InformationAssurance')))
```

[4] Asserts that there is a Security that governs the exchange

```
self.dataSecurity->forAll(classifier->forAll(name='Security')) or

self.dataSecurity->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Security')))
```

[5] Asserts that this exchange is under transaction control

```
self.transaction->forAll(classifier->forAll(name='Transaction')) or

self.transaction->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Transaction')))
```

[6] Asserts that this DataExchange occurs between two Systems or specializations of System.

```
(self.target.oclAsType(uml::Class).getAppliedStereotype('UPDM::System')->
notEmpty() or

self.target.oclAsType(uml::Class).getAppliedStereotypes()-> collect(allParents())-
>any(qualifiedName = 'UPDM::System') ->notEmpty())

and

(self.source.oclAsType(uml::Class).getAppliedStereotype('UPDM::System')->
notEmpty() or

self.source.oclAsType(uml::Class).getAppliedStereotypes()-> collect(allParents())-
>any(qualifiedName = 'UPDM::System') ->notEmpty())
```

[7] Asserts that all the systemActivityFlows are of type SystemControlFlow or SystemInformationFlow and that they are owned by a SystemFunction

```
self.realizingActivityEdge->
forAll((getAppliedStereotype('UPDM::SystemControlFlow')->notEmpty() or
getAppliedStereotype('UPDM::SystemInformationFlow')->notEmpty() ) and
owner.getAppliedStereotype('UPDM::SystemFunction')->notEmpty()
)
```

[8] Asserts that the Interaction that contains an InformationExchange must be an OperationalEventTrace

```
self.realizingMessage.getAppliedStereotype('UPDM::TaskInvocation')->notEmpty()
```

8.6.11.7 Semantics

No additional semantic requirements.

8.6.11.8 Notation

No additional notational requirements.

8.6.12 Forecast

8.6.12.1 Extension

· Class (from UML2)

8.6.12.2 Generalizations

8.6.12.3 Description

A Forecast describes the actual or predicted status of a System at a Project Milestone - i.e., a point in the lifecycle of the system. It can be a statement about the future state of one or more types of System or TechnicalStandardsForecast.

The Forecast is effective for a given timePeriod.

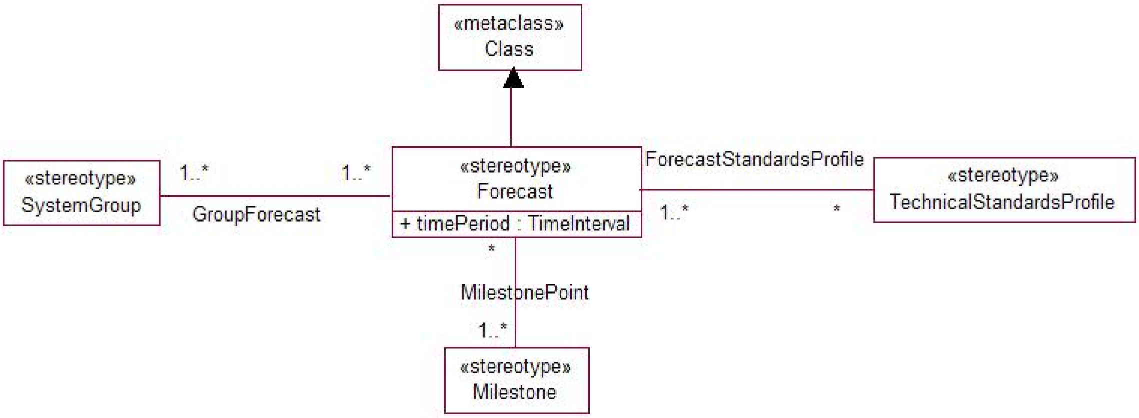


Figure 8.121 - Forecast

8.6.12.4 Attributes

- timePeriod : TimeInterval [1]

8.6.12.5 Associations

- milestone : Milestone [1..*]
  Milestones attributed to this Forecast
- systemGroup : SystemGroup [1..*]
  SystemGroup that is described by this Forecast.
- technicalStandardsProfile : TechnicalStandardsProfile [*]
  TechnicalStandardsProfile that is described by this Forecast

8.6.12.6 Constraints

[1] Asserts that the Forecast is bounded by a TimeInterval

```
self.timePeriod->forAll(classifier->forAll(name='TimeInterval')) or

self.timePeriod->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='TimeInterval')))
```

[2] Asserts that the Forecast applies to one or more SystemGroups

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::GroupForecast')-> notEmpty())->notEmpty()
```

[3] Asserts that the Forecast applies to one or more TechnicalStandardsProfiles

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

(getAppliedStereotype('UPDM::ForecastStandardsProfile')-> notEmpty())->notEmpty()
```

[4] Asserts that this Forecast is bounded by a Milestone

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::MilestonePoint')-> notEmpty())->notEmpty()
```

8.6.12.7 Semantics

No additional semantic requirements.

8.6.12.8 Notation

No additional notational requirements.

8.6.13 GroupForecast

8.6.13.1 Extension

- Association (from UML2)

8.6.13.2 Generalizations

8.6.13.3 Description

Asserts that a SystemGroup has an associated Forecast.

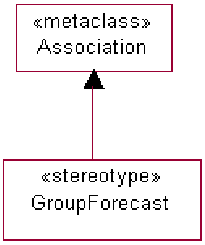


Figure 8.122 - GroupForecast

8.6.13.4 Attributes

No additional attributes.

8.6.13.5 Associations

No additional associations.

8.6.13.6 Constraints

[1] Asserts that the ends of the association are Forecast and SystemGroup

```
(self.endType->at(1).getAppliedStereotype('UPDM::Forecast')->notEmpty() and
self.endType->at(2).getAppliedStereotype('UPDM::SystemGroup')->notEmpty()) or
(self.endType->at(2).getAppliedStereotype('UPDM::Forecast')->notEmpty() and
self.endType->at(1).getAppliedStereotype('UPDM::SystemGroup')->notEmpty())
```

8.6.13.7 Semantics

No additional semantic requirements.

8.6.13.8 Notation

No additional notational requirements.

8.6.14 Location

8.6.14.1 Extension

- Class (from UML2)

8.6.14.2 Generalizations

8.6.14.3 Description

A Location is anywhere that can be specified. The means of describing the location is a string (locationDescription). The information contained in that string is governed by the locationType.

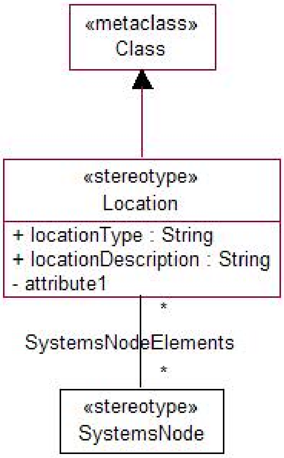


Figure 8.123 - Location

8.6.14.4 Attributes

- locationDescription : String [1]
  A description of the Location depending on the locationType.
- locationType : String [1]
  A description o f the means of location - e.g., GPS

8.6.14.5 Associations

- systemsnode : SystemsNode [*]
  SytemsNodes that are located by this Location

8.6.14.6 Constraints

[1] Asserts that a Location applies to at least one SystemsNode

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

8.6.14.7 Semantics

No additional semantic requirements.

8.6.14.8 Notation

No additional notational requirements.

8.6.15 Network

8.6.15.1 Extension

8.6.15.2 Generalizations

- System (from SystemsView)

8.6.15.3 Description

A Network is a collection of one or more CommunicationPaths. It may contain multiple CommunicationPaths between the same pair of Systems. This is typically realized by the Network owning a set of CommunicationPaths.

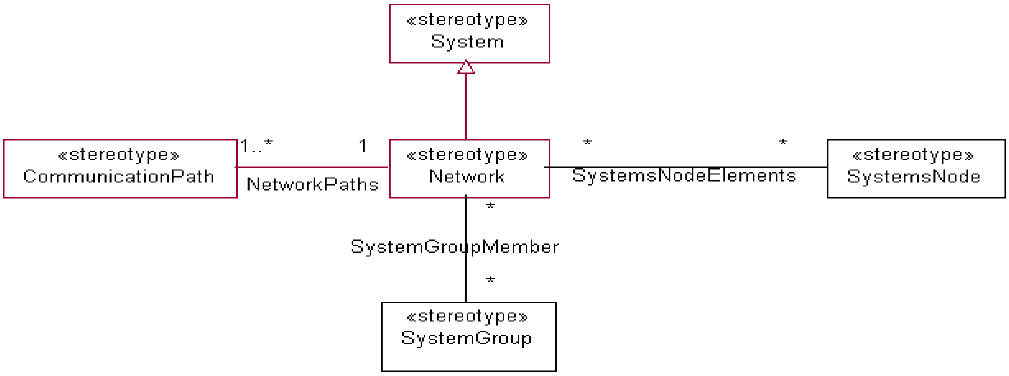


Figure 8.124 - Network

8.6.15.4 Attributes

No additional attributes.

8.6.15.5 Associations

- communicationPaths : CommunicationPath [1..*]
  The set of CommunicationPaths that make up this Network
- systemgroup : SystemGroup [*]
  The SystemGroup to which the Network belongs
- systemsnode : SystemsNode [*]
  The SystemsNode on which the Network is deployed.

8.6.15.6 Constraints

[1] Asserts that the Network consists of at least one CommunicationPath

```
self.communicationPaths.getAppliedStereotype('UPDM::CommunicationPath')-
>notEmpty()
```

[2] Asserts that a SystemGroup includes zero or more Networks

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemGroupMember')-> notEmpty())->notEmpty()
```

[3] Asserts that there is an association between the Network and a SystemsNode on which the Networks are deployed.

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

8.6.15.7 Semantics

No additional semantic requirements.

8.6.15.8 Notation

No additional notational requirements.

8.6.16 NetworkPaths

8.6.16.1 Extension

- Association (from UML2)

8.6.16.2 Generalizations

8.6.16.3 Description

Asserts that a Network is a collection of CommunicationPaths.

CommunicationPaths are routes from one system to another whose detail is specified by an ordered collection of CommunicationLinks or CommunicationSystems.

A Network is a collection of the CommunicationPaths taken as a whole.

If a Network exists, then it is comprised of at least one CommunicationPath, and one or more CommunicationPaths between two systems, constitutes a Network.

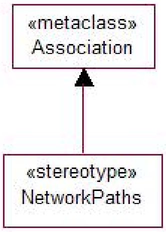


Figure 8.125 - NetworkPaths

8.6.16.4 Attributes

No additional attributes.

8.6.16.5 Associations

No additional associations.

8.6.16.6 Constraints

[1] Asserts that this association is between CommunicationPaths and a Network.

```
let e1:uml::Class = self.oclAsType(uml::Association).endType->asOrderedSet()-
>at(1).oclAsType(uml::Class) in

let e2:uml::Class = self.oclAsType(uml::Association).endType->asOrderedSet()-
>at(2).oclAsType(uml::Class) in

(e1.getAppliedStereotype('UPDM::CommunicationPath')->notEmpty() and

e2.getAppliedStereotype('UPDM::Network')->notEmpty())

or

(e2.getAppliedStereotype('UPDM::CommunicationPath')->notEmpty() and

e1.getAppliedStereotype('UPDM::Network')->notEmpty())
```

8.6.16.7 Semantics

No additional semantic requirements.

8.6.16.8 Notation

No additional notational requirements.

8.6.17 OperationalActivityRealization

8.6.17.1 Extension

8.6.17.2 Generalizations

- System (from SystemsView)

8.6.17.3 Description

OperationalActivityRealization is a collaboration among Systems or SystemFunctionSpecifications that realize an OperationalCapabilityRealization or other System capability.

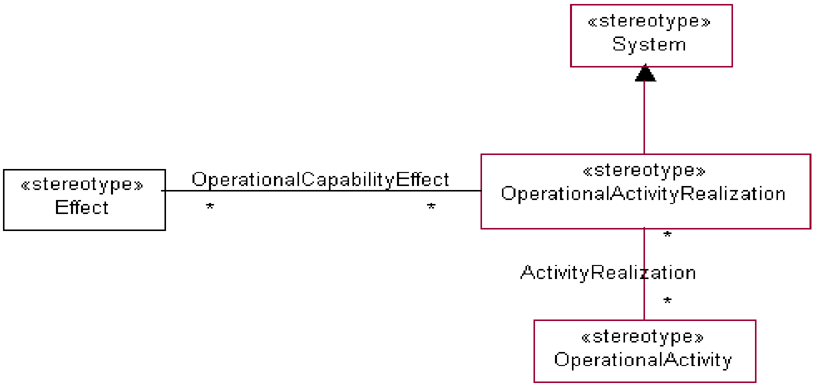


Figure 8.126 - OperationalActivityRealization

8.6.17.4 Attributes

No additional attributes.

8.6.17.5 Associations

- effect : Effect [*]
  Effect that is realized by the OperationalActivityrealization
- operationalActivity : OperationalActivity [*]
  The OperationalActivities that are realized by the Behavior of the OperationalActivityRealization

8.6.17.6 Constraints

[1] Asserts that this OperationalActivityRealization realizes an Effect

```
self.getAllAttributes()->asOrderedSet()->select(association->
notEmpty()).association->any
```

```
  (getAppliedStereotype('UPDM::OperationalCapabilityEffect')-> notEmpty())->
notEmpty()
```

8.6.17.7 Semantics

Insert: In a sense, an OperationalActivityRealization is a System of Systems. It performs one or more SystemFunctions to realize its desired capabilities.

OperationalActivityRealization behavior can be expressed as a SystemFunction, SystemEventTrace and/or a SystemStateTrace.

OperationalActivityRealization may also be associated with Effects to support Effects based planning traced through Effects that are also associated with OperationalCapabilityRealizations.

OperationalActivities can be associated with OperationalActivityRealizations to provide a mapping from OperationalActivities to the SystemFunctions defined on the OperationalActivityRealizations. By tracing the decomposition of those SystemFunctions to their constituent parts, the OperationalActivities can be mapped to the SystemFunctions that implement them.

Define an OperationalActivityRealization for every Effect and its OperationalActivity. Since Effects are also associated with OperationalCapabilities that deliver the Effect, this collaboration provides a clear picture of the SystemFunctions that enable an OperationalCapabilityRealization.

8.6.17.8 Notation

No additional notational requirements.

8.6.18 RealizedSystemSpecification

8.6.18.1 Extension

- Realization (from UML2)

8.6.18.2 Generalizations

8.6.18.3 Description

Specifies that a System or one of its sub stereotypes realizes an interface stereotyped SystemFunctionSpecification or one of its sub stereotypes.

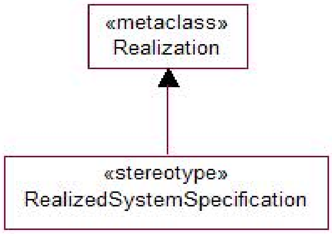


Figure 8.127 - RealizedSystemSpecification

8.6.18.4 Attributes

No additional attributes.

8.6.18.5 Associations

No additional associations.

8.6.18.6 Constraints

[1] Asserts that the source or client of the realization is a System or one of its sub stereotypes and the target or supplier is a SystemFunctionSpecification.

```
self.supplier.getAppliedStereotype('UPDM::SystemFunctionSpecification')-
>notEmpty() or
```

```
(self.client.getAppliedStereotype('UPDM::System')->notEmpty() or
```

```
self.client.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty())
```

8.6.18.7 Semantics

No additional semantic requirements.

8.6.18.8 Notation

No additional notational requirements.

8.6.19 Service

8.6.19.1 Extension

- Port (from UML2)

8.6.19.2 Generalizations

8.6.19.3 Description

The service model element provides the end-point for service interaction (in web service terminology), whereas the definition of these interactions is a part of the service specification stereotype. In your model, a service not only identifies the provided interface, but may also identify required interfaces (such as callback interfaces). A service has an additional property that denotes the binding to be used, such as SOAP-HTTP, SOAP-JMS, and so on.

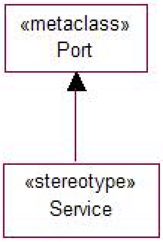


Figure 8.128 - Service

8.6.19.4 Attributes

No additional attributes.

8.6.19.5 Associations

No additional associations.

8.6.19.6 Constraints

[1] Asserts that a Service can only be owned by a ServiceProvider or a ServiceConsumer. A Service is really a port that connects service providers and consumers

```
self.owner.getAppliedStereotype('UPDM::SystemServiceProvider')->notEmpty() or

self.owner.getAppliedStereotype('UPDM::SystemServiceConsumer')->notEmpty() or

self.owner.getAppliedStereotype('UPDM::OperationalServiceProvider')->notEmpty() or

self.owner.getAppliedStereotype('UPDM::OperationalServiceConsumer')->notEmpty()
```

[2] Asserts that this Service must provide at least on ServiceSpecification

```
self.provided->notEmpty() and self.provided-
>forAll(getAppliedStereotype('UPDM::ServiceSpecification')->notEmpty()) or

self.required->notEmpty() and self.required-
>forAll(getAppliedStereotype('UPDM::ServiceSpecification')->notEmpty())
```

8.6.19.7 Semantics

No additional semantic requirements.

8.6.19.8 Notation

No additional notational requirements.

8.6.20 ServiceSpecification

8.6.20.1 Extension

- Interface (from UML2)

8.6.20.2 Generalizations

8.6.20.3 Description

The use of an interface denotes a set of operations provided by a service.

Note that a service may implement more than one interface. By convention, it is possible to attach a protocol state machine or UML 2.0 collaboration or class to such a specification to denote the order of invocation of operations on a service specification. With such a behavioral specification, any implementing service can be validated against not only a static but dynamic specification of its structure and behavior.

Note that the service specification may only provide public operations, and each operation should only consume at most one message and produce at most one message.

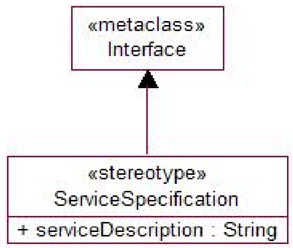


Figure 8.129 - ServiceSpecification

8.6.20.4 Attributes

- serviceDescription : String [1]
  The description of the service or a link to a formal service description.

8.6.20.5 Associations

No additional associations.

8.6.20.6 Constraints

No additional constraints.

8.6.20.7 Semantics

No additional semantic requirements.

8.6.20.8 Notation

No additional notational requirements.

8.6.21 SV-1: Systems Interface Description

8.6.21.1 Extension

8.6.21.2 Generalizations

- SystemView (from SystemsView)

8.6.21.3 Description

Systems Interface Description - Identification of systems nodes, systems, and system items and their interconnections, within and between systems nodes.

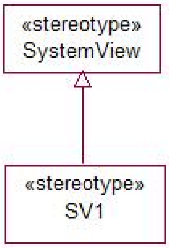


Figure 8.130 - SV-1: Systems Interfact Description

8.6.21.4 Attributes

No additional attributes.

8.6.21.5 Associations

No additional associations.

8.6.21.6 Constraints

No additional constraints.

8.6.21.7 Semantics

No additional semantic requirements.

8.6.21.8 Notation

No additional notational requirements.

8.6.22 SV-2: Systems Communications Description

8.6.22.1 Extension

8.6.22.2 Generalizations

- SystemView (from SystemsView)

8.6.22.3 Description

Systems Communication Description - Systems nodes, systems, and system items, and their related communications laydowns.

Variation a) System Port Specification - System ports and protocols used by those ports when communicating with other systems.

Variation b) System to System Port Connectivity - Protocol stack used by a connection between two ports. The ports may be on different systems.

Variation c) System Connectivity Clusters - Individual connections between system ports, and grouping into logical connections between nodes.

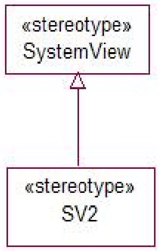


Figure 8.131 - SV-2: Systems Communications Description

8.6.22.4 Attributes

No additional attributes.

8.6.22.5 Associations

No additional associations.

8.6.22.6 Constraints

No additional constraints.

8.6.22.7 Semantics

No additional semantic requirements.

8.6.22.8 Notation

No additional notational requirements.

8.6.23 SV-3: System-Systems Matrix

8.6.23.1 Extension

8.6.23.2 Generalizations

- SystemView (from SystemsView)

8.6.23.3 Description

Systems-Systems Matrix - Relationships among systems in a given architecture; can be designed to show relationships of interest, e.g., system-type interfaces, planned vs. existing interfaces, etc.

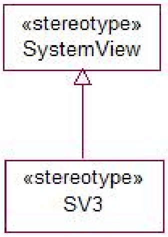


Figure 8.132 - SV-3: Systems-Systems Matrix

8.6.23.4 Attributes

No additional attributes.

8.6.23.5 Associations

No additional associations.

8.6.23.6 Constraints

No additional constraints.

8.6.23.7 Semantics

No additional semantic requirements.

8.6.23.8 Notation

No additional notational requirements.

8.6.24 SV-4: Systems Functionality Description

8.6.24.1 Extension

8.6.24.2 Generalizations

- SystemView (from SystemsView)

8.6.24.3 Description

- Systems Functionality Description - Functions performed by systems and the system data flows among system functions.

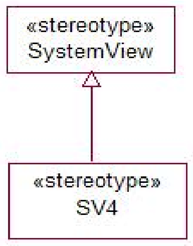


Figure 8.133 - SV-4: Systems Functionality Description

8.6.24.4 Attributes

No additional attributes.

8.6.24.5 Associations

No additional associations.

8.6.24.6 Constraints

No additional constraints.

8.6.24.7 Semantics

No additional semantic requirements.

8.6.24.8 Notation

No additional notational requirements.

8.6.25 SV-5: Operational Activity to Systems Function Traceability Matrix

8.6.25.1 Extension

8.6.25.2 Generalizations

- SystemView (from SystemsView)

8.6.25.3 Description

Operational Activity to Systems Function Traceability Matrix - Mapping of systems back to capabilities or of system functions back to operational activities.

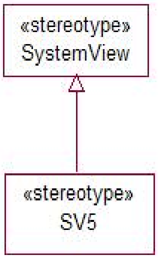


Figure 8.134 - SV-5: Operational Activity to Systems Function Traceability Matrix

8.6.25.4 Attributes

No additional attributes.

8.6.25.5 Associations

No additional associations.

8.6.25.6 Constraints

No additional constraints.

8.6.25.7 Semantics

No additional semantic requirements.

8.6.25.8 Notation

No additional notational requirements.

8.6.26 SV-6: Systems Data Exchange Matrix

8.6.26.1 Extension

8.6.26.2 Generalizations

- SystemView (from SystemsView)

8.6.26.3 Description

Operational Activity to Systems Function Traceability Matrix - Mapping of systems back to capabilities or of system functions back to operational activities.

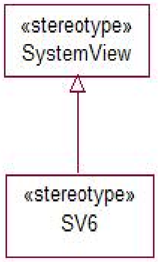


Figure 8.135 - SV-6: Systems Data Exchange Matrix

8.6.26.4 Attributes

No additional attributes.

8.6.26.5 Associations

No additional associations.

8.6.26.6 Constraints

No additional constraints.

8.6.26.7 Semantics

No additional semantic requirements.

8.6.26.8 Notation

No additional notational requirements.

8.6.27 SV-7: Systems Performance Parameters Matrix

8.6.27.1 Extension

8.6.27.2 Generalizations

- SystemView (from SystemsView)

8.6.27.3 Description

System Performance Parameters Matrix - Performance characteristics of Systems View elements for the appropriate time frame(s).

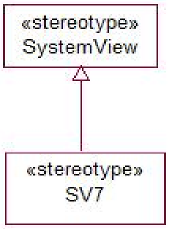


Figure 8.136 - SV-7: Systems Performance Parameters Matrix

8.6.27.4 Attributes

No additional attributes.

8.6.27.5 Associations

No additional associations.

8.6.27.6 Constraints

No additional constraints.

8.6.27.7 Semantics

No additional semantic requirements.

8.6.27.8 Notation

No additional notational requirements.

8.6.28 SV-8: Systems Evolution Description

8.6.28.1 Extension

8.6.28.2 Generalizations

- SystemView (from SystemsView)

8.6.28.3 Description

Systems Evolution Description - Planned incremental steps for migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation.

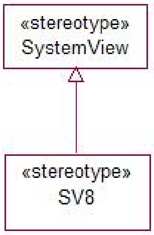


Figure 8.137 - SV-8: Systems Evolution Description

8.6.28.4 Attributes

No additional attributes.

8.6.28.5 Associations

No additional associations.

8.6.28.6 Constraints

No additional constraints.

8.6.28.7 Semantics

No additional semantic requirements.

8.6.28.8 Notation

No additional notational requirements.

8.6.29 SV-9: Systems Technology Forecast

8.6.29.1 Extension

8.6.29.2 Generalizations

- SystemView (from SystemsView)

8.6.29.3 Description

Systems Technology Forecast- Emerging technologies and software/hardware products that are expected to be available in a given set of time frames and that will affect future development of the architecture.

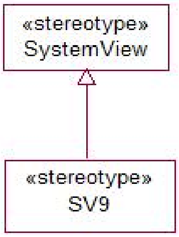


Figure 8.138 - SV-9: Systems Technology Forecast

8.6.29.4 Attributes

No additional attributes.

8.6.29.5 Associations

No additional associations.

8.6.29.6 Constraints

No additional constraints.

8.6.29.7 Semantics

No additional semantic requirements.

8.6.29.8 Notation

No additional notational requirements.

8.6.30 SV-10: Systems Functionality Sequence and Timing Descriptions (SV-10A, 10B and 10C)

8.6.30.1 Extension

8.6.30.2 Generalizations

- SystemView (from SystemsView)

8.6.30.3 Description

Variation a) Systems Rule Model - Identifies constraints that are imposed on systems functionality due to some aspect of systems design or implementation.

Variation b) Systems State Transition Description - Identifies responses of a system to events.

Variation c) Systems Event-Trace Description - Identifies system-specific refinements of critical sequences of events described in the OperationalView

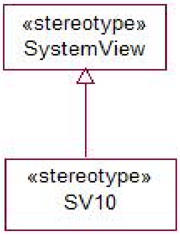


Figure 8.139 - SV-10: Systems Functionality Sequence and Timing Descriptions

8.6.30.4 Attributes

No additional attributes.

8.6.30.5 Associations

No additional associations.

8.6.30.6 Constraints

No additional constraints.

8.6.30.7 Semantics

No additional semantic requirements.

8.6.30.8 Notation

No additional notational requirements.

8.6.31 SV-11: Physical Schema

8.6.31.1 Extension

8.6.31.2 Generalizations

- SystemView (from SystemsView)

8.6.31.3 Description

Physical Schema - Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema.

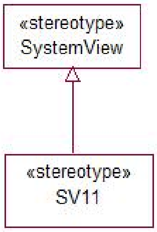


Figure 8.140 - SV-11: Physical Schema

8.6.31.4 Attributes

No additional attributes.

8.6.31.5 Associations

No additional associations.

8.6.31.6 Constraints

No additional constraints.

8.6.31.7 Semantics

No additional semantic requirements.

8.6.31.8 Notation

No additional notational requirements.

8.6.32 SVCustom: Custom Systems View

8.6.32.1 Extension

8.6.32.2 Generalizations

- SystemView (from SystemsView)

8.6.32.3 Description

A user-defined view that applies to the SystemsView.

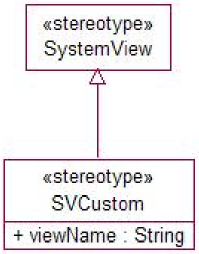


Figure 8.141 - SVCustom: Custom Systems View

8.6.32.4 Attributes

- viewName : String [1]
  Provides a name for the custom view so that it can be easily distinguished from other custom SystemsView.

8.6.32.5 Associations

No additional associations.

8.6.32.6 Constraints

No additional constraints.

8.6.32.7 Semantics

No additional semantic requirements.

8.6.32.8 Notation

No additional notational requirements.

8.6.33 System

8.6.33.1 Extension

8.6.33.2 Generalizations

- ConformingElement (from AllViews)

8.6.33.3 Description

A System is any organized assembly of resources and procedures united and regulated by interaction or interdependence to accomplish a set of specific functions.

Systems consist of family of systems (FoS), System of systems (SoS), Networks of systems, individual Systems, and items (e.g., equipment SystemHardware and SystemSoftware).

A System provides the core construct of the SystemView. A System can be expressed in UML using a variety of constructs, for example, a component or a class. Systems will have a number of operations and functions that they will deliver. The subset of information that relates to SystemFunctions will typically reflect only those functions of interest to link to OperationalActivities.

Two examples of approaches to system modeling indicate the need for extending such a basic UML construct as Class. A Systems View that is reviewing sets of components may model those systems as Components and align potential 'to be' architectures to the SystemFunction definitions. The component can have required and provided SystemFunctionSpecifications, modeled as UML interfaces.

In contrast, other approaches will need to model the detailed physical structure of the Systems and components, requiring detailed drill down into the composite structure of a System.

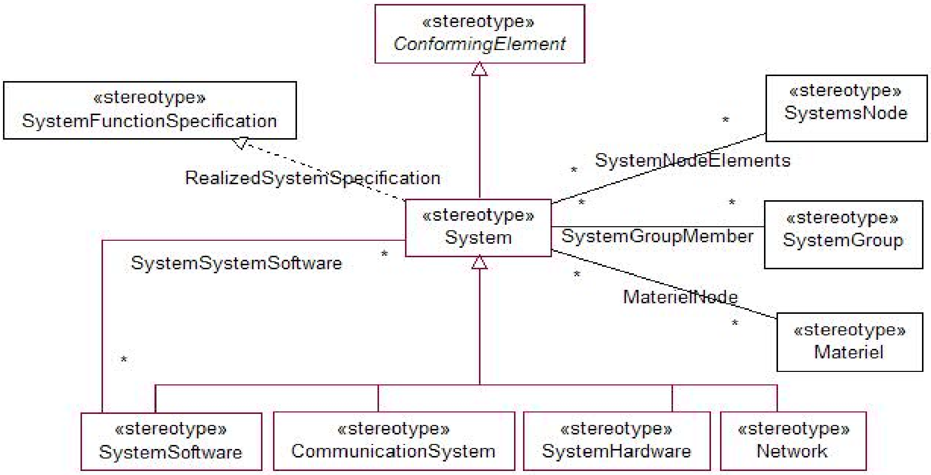


Figure 8.142 - System

8.6.33.4 Attributes

No additional attributes.

8.6.33.5 Associations

- from : CommunicationPath [*]
  The ‘from’ System in a CommunicationPath

- materiel : Materiel [*]
  Materiel needed by the System

- systemgroup : SystemGroup [*]
  The SystemsGroups to which this system belongs

- systemsnode : SystemsNode [*]
  The SystemsNode on which this system is deployed.

- systemSoftware : SystemSoftware [*]
  Software used by this System. See SystemSystemSoftware for more details.

- to : CommunicationPath [*]
  The ‘to’ System in a CommunicationPath

8.6.33.6 Constraints

[1] If a Needline exists between two OperationalNodes, then there must exist a SystemsNode that hosts each of the OperationalNodes and a SystemInterface must exist between them.

```
self.getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any
 (getAppliedStereotype('UPDM::SystemInterface')-> notEmpty())->notEmpty()


self.getAllAttributes().association ->
any (getAppliedStereotype('UPDM::SystemSystemSoftware')->notEmpty())->notEmpty()
```

[2] For all ports defined for a System, at least one of its stereotypes must be 'SystemPort'

```
self.oclAsType(uml::Class).ownedPort-> asBag()->
 forAll(p|p.getAppliedStereotype('UPDM::SystemPort')->notEmpty())
```

[3] Asserts that a System realizes interfaces that are SystemFunctionSpecifications, i.e., interfaces

```
self.clientDependency->forAll(spec|
if spec.oclIsKindOf(uml::InterfaceRealization)then
spec.getAppliedStereotype('UPDM::RealizedSystemSpecification')->notEmpty()
else
true
endif)
```

[4] Asserts that a System is part of at least one SystemGroup

```
self.getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any
 (getAppliedStereotype('UPDM::SystemGroupMember')-> notEmpty())->notEmpty()
```

[5] Asserts that there is an association between the System and a SystemsNode on which the Systems are deployed.

```
self.getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any
 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

[6] Asserts that there is at least oneSystemTask defined for the System

```
self.oclAsType(uml::Class).ownedOperation->asOrderedSet() ->
```

```
exists(getAppliedStereotype('UPDM::SystemTask')->notEmpty())
```

[7] Asserts that all behavior of a System must be one of SystemFunction, SystemEventTrace or SystemStateTrace

```
self.oclAsType(uml::Class).ownedBehavior->asOrderedSet()->exists(a|
a.getAppliedStereotype('UPDM::SystemFunction')->notEmpty() or
a.getAppliedStereotype('UPDM::SystemlEventTrace')->notEmpty()or
a.getAppliedStereotype('UPDM::SystemStateTrace')->notEmpty())
```

8.6.33.7 Semantics

No additional semantic requirements.

8.6.33.8 Notation

No additional notational requirements.

8.6.34 SystemControlFlow

8.6.34.1 Extension

- ControlFlow (from UML2)

8.6.34.2 Generalizations

8.6.34.3 Description

A flow of control, energy from one activity node to another.

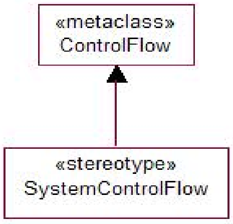


Figure 8.143 - SystemControlFlow

8.6.34.4 Attributes

No additional attributes.

8.6.34.5 Associations

No additional associations.

8.6.34.6 Constraints

[1] Asserts that the source of the SystemControlFlow is one of

a CallBehaviorAction whose behavior is a SystemFunction

a CallOperationAction whose operation is a SystemTask

an InitialNode

```
if self.source.oclIsKindOf(uml::CallOperationAction) then

self.source.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UP
DM::SystemTask')->notEmpty()

else

if self.source.oclIsKindOf(uml::CallBehaviorAction) then

self.source.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM
::SystemFunction')->notEmpty()

else

   self.source.oclIsKindOf(uml::InitialNode)

endif

endif
```

[2] Asserts that the target of the SystemControlFlow is one of:

```
   a CallBehaviorAction whose behavior is an SystemFunction

   a CallOperationAction whose operation is a SystemTask

   anFinalNode

if self.target.oclIsKindOf(uml::CallOperationAction) then


self.target.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UP
DM::SystemTask')->notEmpty()

else

   if self.target.oclIsKindOf(uml::CallBehaviorAction) then


self.target.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM
::SystemFunction')->notEmpty()

   else

      self.target.oclIsKindOf(uml::FinalNode)

   endif
```

```
endif
```

8.6.34.7 Semantics

SystemControlFlows are the flow of control among activity nodes, usually represented on Activity Diagrams.

The flow of control from one action to another specifies the actions within a SystemFunction. The actions invoke either specific SystemTasks or other SystemFunctions. A SystemControlFlow has a source action (FROM), the source of the SystemControlFlow that is one of a CallBehaviorAction whose behavior is a SystemFunction, a CallOperationAction whose operation is a SystemTask, or an InitialNode. It has a target action (TO), the target of the SystemControlFlow, that is one of a CallBehaviorAction whose behavior is an SystemFunction, a CallOperationAction whose operation is an SystemTask, or a FinalNode

A SystemControlFlow specifies partially or fully one InformationExchange.

8.6.34.8 Notation

No additional notational requirements.

8.6.35 SystemEventTrace

8.6.35.1 Extension

- Interaction (from UML2)

8.6.35.2 Generalizations

8.6.35.3 Description

Specifies the behavior of a System or OperationalActivityRealization as an Interaction - by depicting the sequence of events between Systems and the invocation (TaskInvocation) of SystemTasks. The behavior of an interaction is depicted in a sequence diagram that shows the ordered exchange of information between Systems through the activation of their SystemTasks that participate in the interaction..

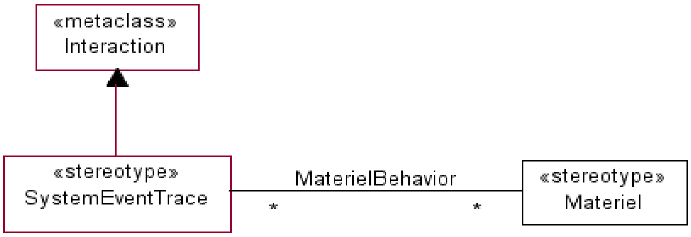


Figure 8.144 - SystemEventTrace

8.6.35.4 Attributes

No additional attributes.

8.6.35.5 Associations

- materiel : Materiel [*]
  Materiel needed by this behavior.

8.6.35.6 Constraints

```
let x:uml::Class = self.owner.oclAsType(uml::Class) in

x.getAppliedStereotype('UPDM::OperationalActivityRealization') ->notEmpty() or

x.getAppliedStereotype('UPDM::System')->notEmpty() or

x.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()
```

8.6.35.7 Semantics

No additional semantic requirements.

8.6.35.8 Notation

No additional notational requirements.

8.6.36 SystemFunction

8.6.36.1 Extension

- Activity (from UML2)

8.6.36.2 Generalizations

8.6.36.3 Description

Defines the behavior of a System or OperationalActivityRealization as an activity in terms of invocations of SystemFunctions connected by SystemControlFlows and the flow of objects through the System with SystemInformationFlows.

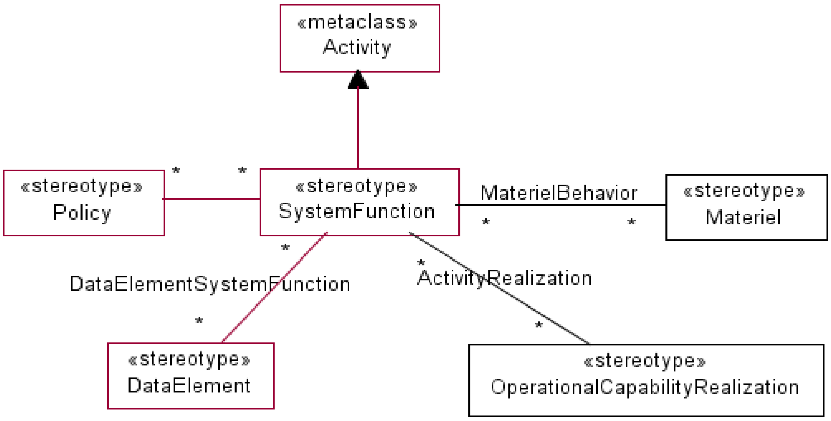


Figure 8.145 - SystemFunction

8.6.36.4 Attributes

No additional attributes.

8.6.36.5 Associations

- dataElement : DataElement [*]
  The DataElements acted upon by this SystemFunction See DataElementSystemFunction for more details.
- materiel : Materiel [*]
  Materiel needed by this SystemFunction
- operationalCapabilityRealization : OperationalCapabilityRealization [*]
  OperationalCapabilityRealizations realized by this SystemFunction
- policy : Policy [*]
  Policies that govern this activity

8.6.36.6 Constraints

[1] Asserts that a DataElementSystemFunction association exists between DataElement and SystemFunction

```
(self.endType->at(1).getAppliedStereotype('UPDM::SystemFunction')->notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::DataElement')->notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::SystemFunction')->notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::DataElement')->notEmpty())
```

[2] Asserts that this SystemFunction specifies the behavior of a System, one of its specializations or an OperationalActivityRealization

```
let x:uml::Class = self.owner.oclAsType(uml::Class) in

x.getAppliedStereotype('UPDM::OperationalActivityRealization') ->notEmpty() or

x.getAppliedStereotype('UPDM::System')->notEmpty() or

x.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()
```

[3] Asserts that the entries in policy are type Policy

```
self.policy->forAll(getAppliedStereotype('UPDM::Policy')->notEmpty())
```

[4] Asserts that for every callOperationAction in a SystemFunction that refers to an Operation that is stereotyped SystemTask then:

- if the SystemTask that is the operation of a callOperationAction that resides in a partition that represents a System or a Property that is typed by a System or one of its specializations then
- the SystemTask must be owned by that System and the SystemTask must have at least one method that specifies a corresponding SystemFunction owned by the System, (and SystemTask could also specify Interactions or StateTraces)

OR

- if the SystemTask that is the operation of a callOperationAction that resides in a partition that represents a SystemFunctionSpecification or a Property that is typed by a SystemFunctionSpecification ,
- then the SystemFunctionSpecification must own the SystemTask.

```
self.node->select(

      nde|nde.oclIsKindOf(uml::CallOperationAction) and

nde.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UPDM::Syst
emTask') ->notEmpty()

)->forAll(op|op.inPartition.represents->forAll(x|not x.oclIsUndefined()) and

(op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::System')->notEmpty() or
reps.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()). oclAsType(uml::Class).ownedOperation->any(ownedop|
ownedop=op.oclAsType(uml::CallOperationAction).operation and ownedop.method->
any(meth|meth.oclIsTypeOf(uml::Activity) and
meth.getAppliedStereotype('UPDM::SystemFunction')->notEmpty() and meth.owner =
ownedop.owner)-> notEmpty() )->notEmpty()

          or

op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::SystemFunctionSpecification')-
>notEmpty()).oclAsType(uml::Interface).ownedOperation-
>any(ownedop|ownedop=op.oclAsType(uml::CallOperationAction).operation)->notEmpty()

      or op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::System')-
>notEmpty() or reps.oclAsType(uml::Property).type.getAppliedStereotypes()-
>collect(allParents())->any(qualifiedName = 'UPDM::System') ->
notEmpty()).oclAsType(uml::Property).type.oclAsType(uml::Class).ownedOperation-
>any(ownedop| ownedop=op.oclAsType(uml::CallOperationAction).operation and
ownedop.method->any(meth|meth.oclIsTypeOf(uml::Activity) and
meth.getAppliedStereotype('UPDM::SystemFunction')->notEmpty() and  meth.owner =
ownedop.owner

 )->notEmpty())->notEmpty()

or

op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::SystemFunctionSpeci
fication')->notEmpty()

).oclAsType(uml::Property).type.oclAsType(uml::Interface).ownedOperation->

any(ownedop|ownedop=op.oclAsType(uml::CallOperationAction).operation)->notEmpty()

     ))
```

[5] Asserts that a SystemFunction that is invoked by a callBehaviorAction in a Partition of a SystemFunction that represents an element, then that element must be stereotyped SystemFunctionSpecification or System or one of its specializations or a Property that is typed by a System or a SystemFunctionSpecification. The SystemFunction specification must be a SystemTask that is owned by the owner of the SystemFunction.

If the partition represents an interface, then that interface must be implemented by the owner of the SystemFunction and own a SystemlTask that has the same name as the SystemFunction's specification.

Usage: This constraint assures the navigability from a System to all of the SystemFunctions that it performs.

```
self.node->select(nde| nde.oclIsKindOf(uml::CallBehaviorAction) and

nde.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM::System
Function') ->notEmpty()   )->forAll(op| op.inPartition.represents->

forAll(x|not x.oclIsUndefined() ) and

(op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::System')->notEmpty() or
reps.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()). oclAsType(uml::Class).ownedBehavior->any(

ownedop| ownedop=op.oclAsType(uml::CallBehaviorAction).behavior and
ownedop.specification->any(spec| spec.oclIsTypeOf(uml::Operation) and

   spec.getAppliedStereotype('UPDM::SystemTask')->notEmpty() and spec.owner =
ownedop.owner)->notEmpty()

      )->notEmpty()

or

op.inPartition.represents->select(reps|
reps.getAppliedStereotype('UPDM::SystemFunctionSpecification')->notEmpty() and

op.oclAsType(uml::CallBehaviorAction).behavior.owner.oclAsType(uml::Class).clientD
ependency.supplier->any(x|x = reps)->notEmpty()).
oclAsType(uml::Interface).ownedOperation->any(ownedop|

 ownedop.name = op.oclAsType(uml::CallBehaviorAction).behavior.specification.name

      )->notEmpty()

or

op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::System')-
>notEmpty() or

reps.oclAsType(uml::Property).type.getAppliedStereotypes()->collect(allParents())-
>any(qualifiedName = 'UPDM::System') ->notEmpty()

). oclAsType(uml::Property).type.oclAsType(uml::Class).ownedBehavior->any(

      ownedop| ownedop=op.oclAsType(uml::CallBehaviorAction).behavior and
ownedop.specification->any(spec|
spec.oclIsTypeOf(uml::Operation) and
spec.getAppliedStereotype('UPDM::SystemTask')->notEmpty() and spec.owner =
ownedop.owner

      )->notEmpty()

       )->notEmpty()

or

op.inPartition.represents->select(reps|
reps.oclAsType(uml::Property).type.getAppliedStereotype('UPDM::SystemFunctionSpeci
fication')->notEmpty() and

op.oclAsType(uml::CallBehaviorAction).behavior.owner.oclAsType(uml::Class).clientD
ependency.supplier->any(x|x = reps)->notEmpty()

).oclAsType(uml::Property).type.oclAsType(uml::Interface).ownedOperation->

 any(ownedop|ownedop.name
=op.oclAsType(uml::CallBehaviorAction).behavior.specification.name)->notEmpty()

      )

)
```

8.6.36.7 Semantics

No additional semantic requirements.

8.6.36.8 Notation

No additional notational requirements.

8.6.37 SystemFunctionSpecification

8.6.37.1 Extension

- Interface (from UML2)

8.6.37.2 Generalizations

8.6.37.3 Description

SystemFunctionSpecifications are abstract Systems modeled as Interfaces. They are not limited to internal system functions and can include Human Computer Interface (HCI) and Graphical User Interface (GUI) functions or functions that consume or produce DataElements from/to SystemFunctionSpecifications that belong to external systems. The external system data sources and/or sinks can be used to represent the humans that interact with the system or external systems. The SystemInformationFlows between the external system data source/sink (representing the human or system) and the HCI, GUI, or SystemFunctionSpecification can be used to represent human-system interactions, or system-system interfaces. Standards that apply to system functions, such as HCI and GUI standards, are also specified in this product through the ConformingElement relationship.

The relationship between OperationalActivities and SystemFunctions can also be expected to be many-to- many.

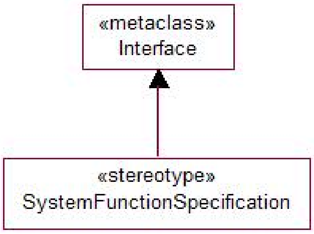


Figure 8.146 - SystemFunctionSpecification

8.6.37.4 Attributes

No additional attributes.

8.6.37.5 Associations

No additional associations.

8.6.37.6 Constraints

[1] Asserts that there is at least one SystemTask defined for the SystemFunctionSpecification

```
self.oclAsType(uml::Interface).ownedOperation->asOrderedSet() ->

exists(getAppliedStereotype('UPDM::SystemTask')->notEmpty())
```

8.6.37.7 Semantics

In many implementations of DoDAF, SystemFunctionSpecifications have an abstract or logical expression in order to demonstrate a “to be” architecture to support the desired business functions outlined in the OperationalView. In addition, the SystemFunctionSpecification's importance relates to the core system functions that are shown on the SV-5. These are expressed as UML interfaces with Systems that implement these SystemFunctionSpecifications realizing that interface.

A System that implements that SystemFunctionSpecification interface provides that SystemFunctionSpecification. UPDM allows for multiple systems to implement the same SystemFunctionSpecification. In addition, the SystemFunctionSpecification allows for a more coarse grained assessment of System Functionality of the type likely to interest viewers of an SV-5, depending on the nature, scope, and goals of the architectural descriptions.

The SystemFunctionSpecification can have any number of operations, SystemTasks. For high level architectures, the interface will define one Operation. On the SV-4, the SystemFunctionSpecification description, the activity diagram describing the choreography of the SystemFunctionSpecifications will indicate usage of that SystemFunctionSpecification via callBehaviorActions relating to SystemTasks on the activity diagram.

If the SystemFunctionSpecification is a service, then delivery will be via a ServiceSpecification. There may be more than one ServiceSpecification to this SystemFunctionSpecification, depending on the consumers. For example, there may be systems based on different software languages or interface protocols such as CORBA and J2EE-RMI.

8.6.37.8 Notation

No additional notational requirements.

8.6.38 SystemGroup

8.6.38.1 Extension

- Class (from UML2)

8.6.38.2 Generalizations

8.6.38.3 Description

The SystemGroup is a collection that includes instances of System, SystemHardware and SystemSoftware that forms a unit for tracking and assessment purposes.

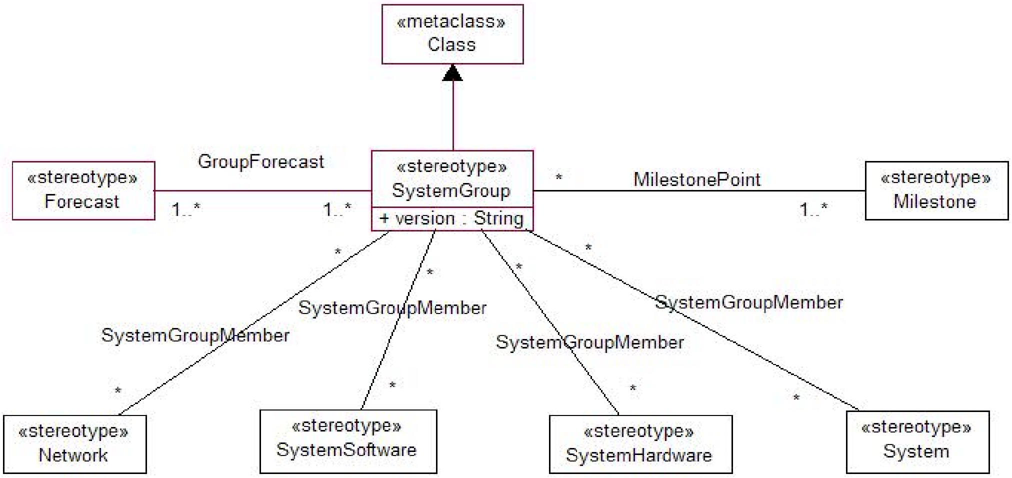


Figure 8.147 - SystemGroup

8.6.38.4 Attributes

- version : String [1]
  Version identifier for the system group.

8.6.38.5 Associations

- forecast : Forecast [1..*]
  Forecast that applies to this SystemGroup

- milestone : Milestone [1..*]
  Milestones that apply to this SystemGroup

- network : Network [*]
  Networks belonging to the SystemGroup

- system : System [*]
  Systems belonging to the SystemGroup

- systemhardware : SystemHardware [*]

Hardwarebelonging to the SystemGroup

- systemsoftware : SystemSoftware [*]
  Software belonging to the SystemGroup

8.6.38.6 Constraints

[1] Asserts that a GroupForecast association exists

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::GroupForecast')-> notEmpty())->notEmpty()
```

[2] Asserts that the SystemGroup is related to a Milestone

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::MilestonePoint')-> notEmpty())->notEmpty()
```

[3] Asserts that there is at least one association between SystemGroup and one of: Systems, Networks, HardwareItems or SoftwareItems associated with the SystemGroup

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemGroupMember')-> notEmpty())->notEmpty()
```

8.6.38.7 Semantics

The SystemGroup allows the aggregations of the Systems and supporting software and hardware into major units for tracking. The SystemGroups at a particular period of time allow the attainment of goals associated with Milestones. The period of time related to the SystemGroup will usually be a SystemEvolutionTimePeriod.

8.6.38.8 Notation

No additional notational requirements.

8.6.39 SystemGroupMember

8.6.39.1 Extension

- Association (from UML2)

8.6.39.2 Generalizations

8.6.39.3 Description

Specifies the System elements configured with a System.

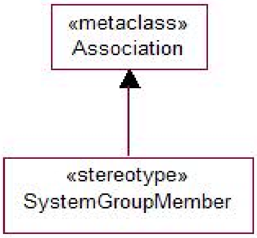


Figure 8.148 - SystemGroupMember

8.6.39.4 Attributes

No additional attributes.

8.6.39.5 Associations

No additional associations.

8.6.39.6 Constraints

[1] Asserts that there are zero or more System elements configured on the System and that these are the elements associated with this association

```
let e1:uml::Class = self.oclAsType(uml::Association).endType-
>at(1).oclAsType(uml::Class) in

let e2:uml::Class = self.oclAsType(uml::Association).endType-
>at(2).oclAsType(uml::Class) in

(e1.getAppliedStereotype('UPDM::SystemGroup')->notEmpty() and

(e2.getAppliedStereotype('UPDM::SystemSoftware')->notEmpty() or

e2.getAppliedStereotype('UPDM::SystemHardware')->notEmpty() or

e2.getAppliedStereotype('UPDM::System')->notEmpty()or

e2.getAppliedStereotype('UPDM::Network')->notEmpty()))

or

(e2.getAppliedStereotype('UPDM::SystemGroup')->notEmpty() and

(e1.getAppliedStereotype('UPDM::SystemSoftware')->notEmpty()or

e1.getAppliedStereotype('UPDM::SystemHardware')->notEmpty()or

e1.getAppliedStereotype('UPDM::System')->notEmpty()or

e1.getAppliedStereotype('UPDM::Network')->notEmpty()))
```

8.6.39.7 Semantics

No additional semantic requirements.

8.6.39.8 Notation

No additional notational requirements.

8.6.40 SystemHardware

8.6.40.1 Extension

8.6.40.2 Generalizations

- System (from SystemsView)

8.6.40.3 Description

Hardware items associated with a System.

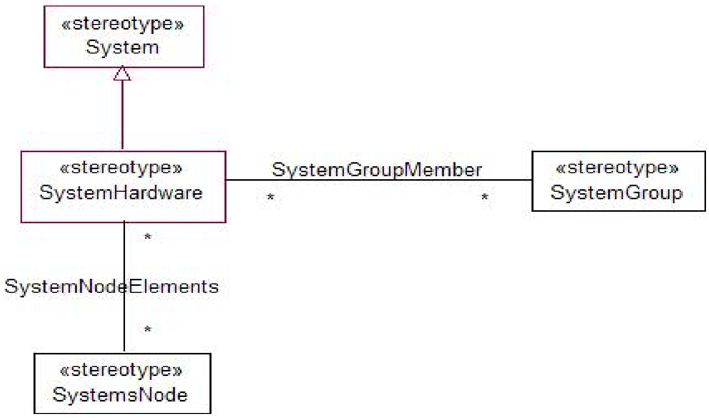


Figure 8.149 - SystemHardware

8.6.40.4 Attributes

No additional attributes.

8.6.40.5 Associations

- systemgroup : SystemGroup [*]
  SystemGroups to which this hardware belongs
- systemsnode : SystemsNode [*]
  SystemsNodes on which this hardware is deployed

8.6.40.6 Constraints

[1] Asserts that there is an association between the Hardware and a SystemGroup

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->
```

```
select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemGroupMember')-> notEmpty())->notEmpty()
```

[2] Asserts that there is an association between the Hardware and a SystemsNode on which the Hardware items are deployed.

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

8.6.40.7 Semantics

No additional semantic requirements.

8.6.40.8 Notation

No additional notational requirements.

8.6.41 SystemInformationFlow

8.6.41.1 Extension

- ObjectFlow (from UML2)

8.6.41.2 Generalizations

8.6.41.3 Description

SystemInformationFlow is the flow of objects, InformationElements, within SystemFunctions. A flow of information, energy or materiel from one activity node to another.

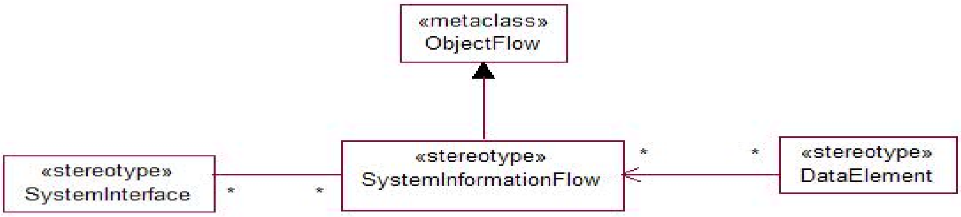


Figure 8.150 - SystemInformationFlow

8.6.41.4 Attributes

No additional attributes.

8.6.41.5 Associations

- systemInterface : SystemInterface [*]
  The SystemInterfaces that carry this SystemInformationFlow

8.6.41.6 Constraints

[1] Asserts that the systemInterface element is of type SystemInterface

```
self.systemInterface.getAppliedStereotype('UPDM::SystemInterface')->notEmpty()

The type of the input or output pin must be an InformationElement.

if self.target.oclIsKindOf(uml::InputPin) then


self.target.oclAsType(uml::InputPin).type.getAppliedStereotype('UPDM::DataElement'
)->notEmpty()

else

if self.source.oclIsKindOf(uml::OutputPin) then


self.source.oclAsType(uml::OutputPin).type.getAppliedStereotype('UPDM::DataElement
')->notEmpty()

else

   false

endif

endif
```

[2] Asserts that if the source of the object flow is an outputpin, then the owner of the output pin must be a SystemAction - a CallBehaviorAction whose behavior is a SystemFunction, or a CallOperationAction whose operation is a SystemTask.

```
if self.oclAsType(uml::ObjectFlow).source.oclIsKindOf(uml::OutputPin) then

   (let src:uml::OutputPin = self.source.oclAsType(uml::OutputPin) in

      (src.owner.oclIsKindOf(uml::CallOperationAction) and


src.owner.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UPDM
::SystemTask')->notEmpty())

   or

      (src.owner.oclIsKindOf(uml::CallBehaviorAction) and


src.owner.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM::
SystemFunction')->notEmpty())

   )
```

```
else
   true
endif
```

[3] Asserts that if the target of the object flow is an input pin, then the owner of the input pin must be a SystemAction - a CallBehaviorAction whose behavior is a SystemFunction, or a CallOperationAction whose operation is a SystemTask.

```
if self.oclAsType(uml::ObjectFlow).target.oclIsKindOf(uml::InputPin) then
   (let trg:uml::InputPin = self.target.oclAsType(uml::InputPin) in
      (trg.owner.oclIsKindOf(uml::CallOperationAction) and

trg.owner.oclAsType(uml::CallOperationAction).operation.getAppliedStereotype('UPDM
::SystemTask')->notEmpty())
   or
      (trg.owner.oclIsKindOf(uml::CallBehaviorAction) and

trg.owner.oclAsType(uml::CallBehaviorAction).behavior.getAppliedStereotype('UPDM::
SystemFunction')->notEmpty())
   )
else
   true
endif
```

8.6.41.7 Semantics

SystemInformationFlows are the flow of information among activity nodes, usually represented on Activity Diagrams. In the case of flow of physical entities, they may include parameters or constraints representing their physical characteristics.

In an IT environment information flow objects' size is not a factor and transmission times are usually considered instantaneous unless the model is, for example, modeling real communications systems that must include factors such as latency, bandwidth and packet size considerations.

In physical flows, the volume or size of the object is a consideration, and the time to traverse the SystemInterface is not instantaneous. In this case, constraints can be applied that indicate the size of the objects (e.g., volume) and the rate of flow.

8.6.41.8 Notation

No additional notational requirements.

8.6.42 SystemInterface

8.6.42.1 Extension

- Association (from UML2)
- Connector (from UML2)

8.6.42.2 Generalizations

8.6.42.3 Description

A SystemInterface is a simplified, abstract representation of one or more communications paths between systems nodes or between systems (including communications systems) and is usually depicted graphically as a straight line. A SystemInterface does not represent an interface typically described in UML but rather a conduit for passing information between two systems

The actual implementation of an interface may take more than one form (e.g., multiple physical links). .

Key interfaces represent those system connections that are especially important for the functioning of the system. By policy, an architecture can demand additional protocols and management for any interface designated as a key interface.

For a system model scoped at a logical level between classifiers that obtain regardless of the context, use an association. For a system model that uses composite structures and looks at the system interface in terms of parts, roles, and connectors, use the connector.

AI SystemInterface is the systems representation of a Needline. A single Needline may translate into multiple SystemInterfaces. Unlike a Needline, the SystemInterface is bi-directional, so it is assumed that the CommunicationPaths allow communication in both directions.

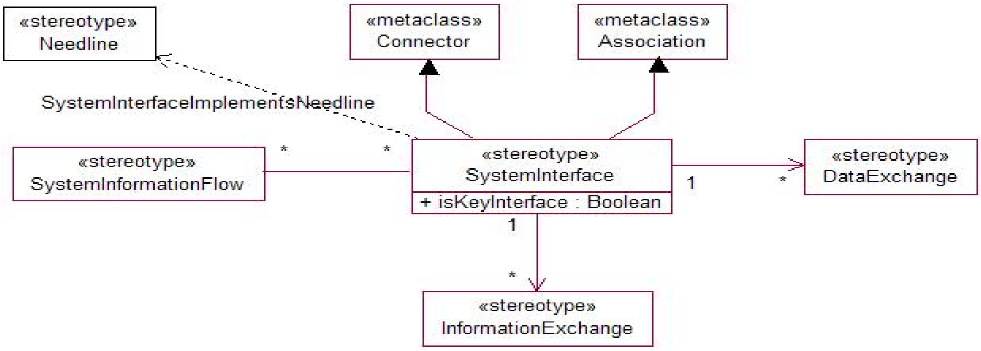


Figure 8.151 - SystemInterface

8.6.42.4 Attributes

- isKeyInterface : Boolean [1]
  Is this System Interface key for system functioning?

8.6.42.5 Associations

- dataExchange : DataExchange [*]
  The dataExchanges that are realized on this SystemInterface

- informationExchange : InformationExchange [*]
  The dataExchanges that are realized on this SystemInterface
- materiel : Materiel [*]
  Materiel used by this SystemInterface
- systemInformationFlow : SystemInformationFlow [*]
  The informationFlows that are carried on this SystemInterface

8.6.42.6 Constraints

[1] Asserts that the elements in the dataExchange are stereotyped DataExchange

```
self.dataExchange-> forAll(getAppliedStereotype('UPDM::DataExchange')->notEmpty())
```

[2] Asserts that the elements in the systemInformationFlow are stereotyped SystemInformationFlow

```
self.systemInformationFlow->
forAll(getAppliedStereotype('UPDM::SystemInformationFlow')->notEmpty())
```

[3] Asserts that if the SystemInterface is modeled as an association then both ends of the SystemInterface are Systems or one of its specializations, or both ends are SystemFunctionsSpecifications.

```
 if self.oclIsTypeOf(uml::Association) then

let nl:uml::Association = self.oclAsType(uml::Association) in

(nl.endType->forAll(getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::System') ->notEmpty() or

getAppliedStereotype('UPDM::System')->notEmpty())) or

nl.endType->forAll(getAppliedStereotype('UPDM::SystemFunctionSpecification')->
notEmpty())

 else

 true

 endif
```

[4] Asserts that the elements in the informationExchange are stereotyped InformationExchange

```
self.informationExchange-
>forAll(getAppliedStereotype('UPDM::InformationExchange')->notEmpty())
```

[5] Asserts that there is at least one dependency between the SystemInterface and a Needline that it implements.

```
self.clientDependency->notEmpty() and

self.clientDependency-
>any(getAppliedStereotype('UPDM::SystemInterfaceImplementsNeedline')->
notEmpty())->notEmpty()
```

[6] Asserts that when SystemInterface is modeled with a Connector, then both ends must connect SystemPorts, both ends must connect Systems or its specializations, or both ends must be SystemFunctionSpecifications.

```
if self.oclIsTypeOf(uml::Connector) then

let con:uml::Connector = self.oclAsType(uml::Connector) in

con.end->forAll(role.oclIsTypeOf(uml::Port) and

role.getAppliedStereotype('UPDM::SystemPort')-> notEmpty()) or

con.end->forAll

(role.oclIsTypeOf(uml::Property) and

(role.type.getAppliedStereotype('UPDM::System')-> notEmpty() or

role.type.getAppliedStereotypes()-> collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty() )) or

con.end->forAll

(role.oclIsTypeOf(uml::Property) and

role.type.getAppliedStereotype('UPDM::SystemFunctionSpecification')-> notEmpty())

 else

 true

 endif
```

8.6.42.7 Semantics

For SystemView products at a high level, such as one that is using components that provide and realize interfaces, the SystemInterfaces will be modeled using an association. For those system descriptions that require drill down into internal structure, a "connector" should be used to in a CompositeStructureDiagram to show a key interface.

8.6.42.8 Notation

No additional notational requirements.

8.6.43 SystemInterfaceImplementsNeedline

8.6.43.1 Extension

- Dependency (from UML2)

8.6.43.2 Generalizations

8.6.43.3 Description

A SystemInterface is the systems representation of a Needline. A single Needline may translate into multiple system interfaces. This dependency specifies that semantics of SystemInterface model elements are dependent on the structure and semantics of the Needlines that they implement.

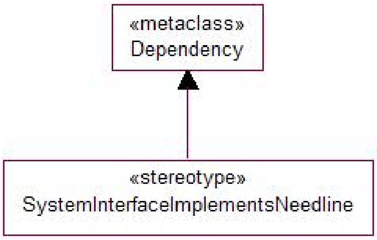


Figure 8.152 - SystemInterfaceImplementsNeedline

8.6.43.4 Attributes

No additional attributes.

8.6.43.5 Associations

No additional associations.

8.6.43.6 Constraints

[1] Asserts that the provider is a Needline and the client is a SystemInterface.

```
self.client.getAppliedStereotype('UPDM::SystemInterface')-> notEmpty()and
self.supplier.getAppliedStereotype('UPDM::Needline')-> notEmpty()
```

8.6.43.7 Semantics

No additional semantic requirements.

8.6.43.8 Notation

No additional notational requirements.

8.6.44 SystemPort

8.6.44.1 Extension

- Port (from UML2)

8.6.44.2 Generalizations

8.6.44.3 Description

A SystemPort represents the system hardware that implements the connection of two systems. An interface (logical or physical) provided by a System. SystemPort may implement a PortType, though there is no requirement for SystemPorts to be typed. The SystemPort is a mechanism to show connecting two systems.

A SystemInterface when modeled as a Connector can have SystemPorts on either end.

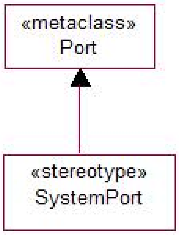


Figure 8.153 - SystemPort

8.6.44.4 Attributes

No additional attributes.

8.6.44.5 Associations

No additional associations.

8.6.44.6 Constraints

[1] Asserts that the end of a SystemPort is a SystemInterface

```
self.oclAsType(uml::Port).end.owner-
>forAll(getAppliedStereotype('UPDM::SystemInterface')-> notEmpty())
```

[2] Asserts that the owner or class of the SystemPort is a System or one of its specializations

```
let x:uml::Class = self.oclAsType(uml::Port).owner.oclAsType(uml::Class) in

x.getAppliedStereotype('UPDM::System')->notEmpty() or

x.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()
```

8.6.44.7 Semantics

No additional semantic requirements.

8.6.44.8 Notation

No additional notational requirements.

8.6.45 SystemServiceConsumer

8.6.45.1 Extension

8.6.45.2 Generalizations

- System (from SystemsView)

8.6.45.3 Description

Any classifier (class, component, and so on) may act as the consumer of a service, and that includes another service. While this stereotype is most definitely optional, it may help you identify elements of a model -- that are not services themselves -- as clients of services.

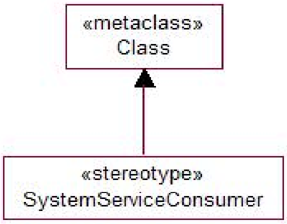


Figure 8.154 - SystemServiceConsumer

8.6.45.4 Attributes

No additional attributes.

8.6.45.5 Associations

No additional associations.

8.6.45.6 Constraints

[1] Asserts that a SystemServiceConsumer must have at least one port that is stereotyped Service and that it requires an interface that is stereotyped ServiceSpecification.

```
self.ownedPort->notEmpty() and

self.ownedPort->exists(p|

p.getAppliedStereotype('UPDM::Service')->notEmpty() and

p.required->notEmpty() and

p.required-> exists(inf|inf.getAppliedStereotype('UPDM::ServiceSpecification')->
notEmpty()))
```

8.6.45.7 Semantics

No additional semantic requirements.

8.6.45.8 Notation

No additional notational requirements.

8.6.46 SystemServiceProvider

8.6.46.1 Extension

8.6.46.2 Generalizations

- System (from SystemsView)

8.6.46.3 Description

The service provider is a software element that provides one or more services. A service provider has a property that captures information about its location, although the meaning of this is implementation-dependent. The class acting as the service provider may not expose any attributes or operations directly: only public ports may be provided (stereotyped as service), and these are typed by service specifications.

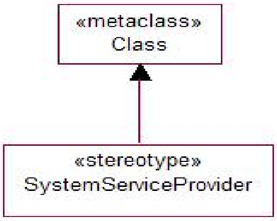


Figure 8.155 - SystemServiceProvider

8.6.46.4 Attributes

No additional attributes.

8.6.46.5 Associations

No additional associations.

8.6.46.6 Constraints

[1] Asserts that a SystemServiceProvider must have at least one port that is stereotyped Service and that it provides an interface that is stereotyped ServiceSpecification.

```
self.ownedPort->notEmpty() and

self.ownedPort->exists(p|

getAppliedStereotype('UPDM::Service')->notEmpty() and

p.provided->notEmpty() and

p.provided-> exists(inf|inf.getAppliedStereotype('UPDM::ServiceSpecification')->
notEmpty()))
```

8.6.46.7 Semantics

No additional semantic requirements.

8.6.46.8 Notation

No additional notational requirements.

8.6.47 SystemsNode

8.6.47.1 Extension

8.6.47.2 Generalizations

- Resource (from AllViews)

8.6.47.3 Description

A SystemsNode is a configuration with identification and allocation of resources (e.g., platforms, units, facilities, and locations) required to implement specific roles and missions. Also known as a Physical Asset in the MODAF.

A class of physical object that can host systems and/or people.

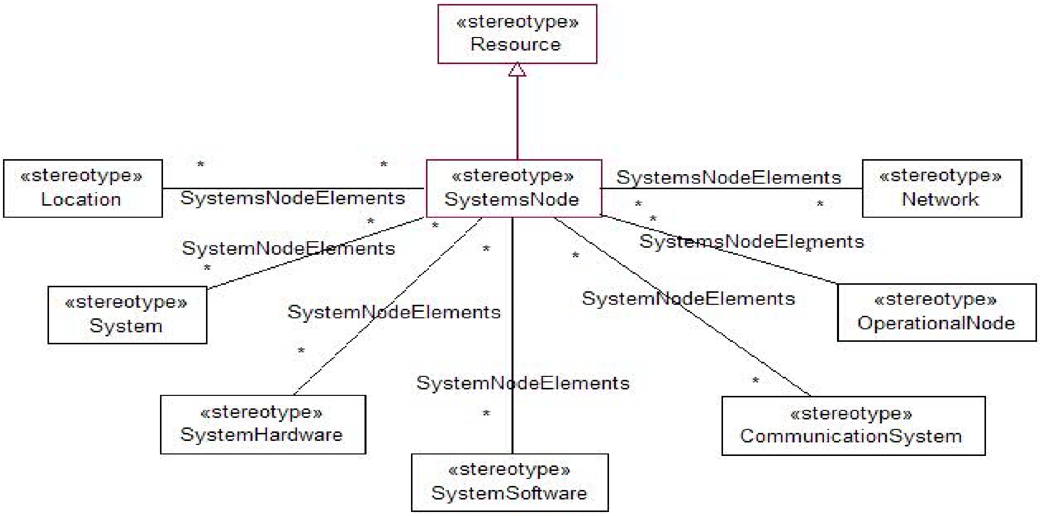


Figure 8.156 - SystemsNode

8.6.47.4 Attributes

No additional attributes.

8.6.47.5 Associations

- communicationsystem : CommunicationSystem [*]
  CommunicationSystems deployed on this SystemsNode
- location : Location [*]
  Location of this SystemsNode
- network : Network [*]
- operationalnode : OperationalNode [*]
  OperationalNodes housed in this SystemsNode

- organizationalresource : OrganizationalResource [1]
  OrganizationalResources deployed on this SystemsNode
- system : System [*]
  Systems deployed on this SystemsNode
- systemhardware : SystemHardware [*]
  Hardware deployed on this SystemsNode
- systemsoftware : SystemSoftware [*]
  Software deployed on this SystemsNode

8.6.47.6 Constraints

[1] Asserts that there is an association between a SystemsNode and its elements

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

8.6.47.7 Semantics

No additional semantic requirements.

8.6.47.8 Notation

No additional notational requirements.

8.6.48 SystemsNodeElements

8.6.48.1 Extension

- Association (from UML2)

8.6.48.2 Generalizations

8.6.48.3 Description

Specifies these are the elements associated with a SystemsNode.

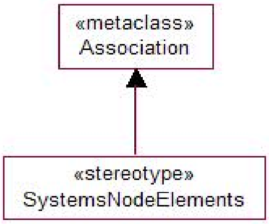


Figure 8.157 - SystemNodeElements

8.6.48.4 Attributes

No additional attributes.

8.6.48.5 Associations

No additional associations.

8.6.48.6 Constraints

[1] Asserts that the SystemsNodes are required to be deployed to zero or more Locations, have OrganizationalResources, HardwareItems, SoftwareItems, Networks, or Systems deployed on them, and house OperationalNodes; and that these are the elements associated with this SystemsNode.

```
let e1:uml::Class = self.oclAsType(uml::Association).endType->
at(1).oclAsType(uml::Class) in

let e2:uml::Class = self.oclAsType(uml::Association).endType->
at(2).oclAsType(uml::Class) in

(e1.getAppliedStereotype('UPDM::SystemsNode')->notEmpty() and

(e2.getAppliedStereotype('UPDM::System')->notEmpty() or

e2.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty() or

e2.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

e2.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty() or

e2.getAppliedStereotype('UPDM::Location')->notEmpty() or

e2.getAppliedStereotype('UPDM::OrganizationalResource')->notEmpty()))

or

(e2.getAppliedStereotype('UPDM::SystemsNode')->notEmpty() and

(e1.getAppliedStereotype('UPDM::System')->notEmpty() or

e1.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty() or

e1.getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or

e1.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::OperationalNode') ->notEmpty() or

e1.getAppliedStereotype('UPDM::Location')->notEmpty() or

e1.getAppliedStereotype('UPDM::OrganizationalResource')->notEmpty())
```

8.6.48.7 Semantics

No additional semantic requirements.

8.6.48.8 Notation

No additional notational requirements.

8.6.49 SystemSoftware

8.6.49.1 Extension

8.6.49.2 Generalizations

- System (from SystemsView)

8.6.49.3 Description

Software needed for the functioning of the system that is typically included as expected background functionality, such as an operating system. The SystemTasks and SystemFunctions of SystemSoftware do not appear in the links to operational activities, typically, and represent functionality that is not the focal point of the architecture description.

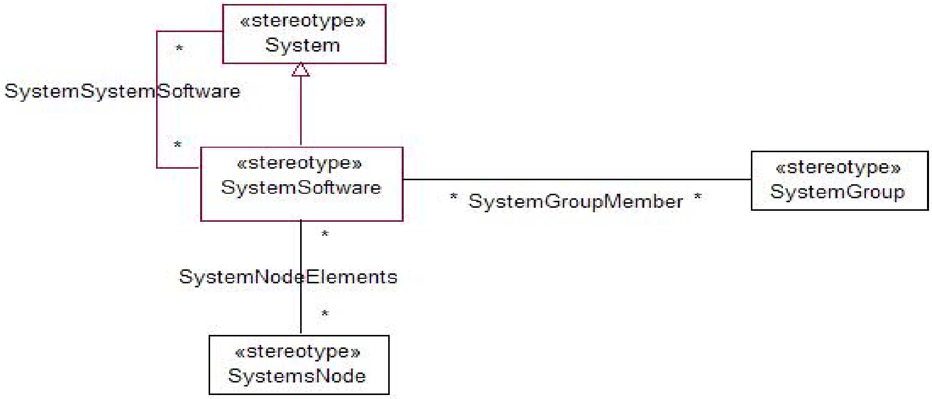


Figure 8.158 - SystemSoftware

8.6.49.4Attributes

No additional attributes.

8.6.49.4 Associations

- system : System [*]
  See SystemSystemSoftware for more details.

- systemgroup : SystemGroup [*]
  SystemGroups to which this Software belong

- systemsnode : SystemsNode [*]
  SystemsNodes on which this Software is deployed

8.6.49.5 Constraints

[1] Asserts that the SystemSoftware is associated with a SystemGroup

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemGroupMember')-> notEmpty())->notEmpty()
```

[2] Asserts that the SystemSoftware is associated with a System

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemSystemSoftware')-> notEmpty())->notEmpty()
```

[3] Asserts that there is an association between the Software and a SystemsNode on which the Software items are deployed.

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

8.6.49.6 Semantics

Depending on the nature of the architectural description, the boundaries for SystemSoftware will differ. In some cases, the concerns of the stakeholder will indicate that monitoring operating system functionality represents too low a level of detail. There are other projects where the project requires a precise understanding of what would be the SystemSoftware for another effort.

A type of system constrained to have operations that are not typically shown, as they are background operations, such as operating system functions.

8.6.49.7 Notation

No additional notational requirements.

8.6.50 SystemStateTrace

8.6.50.1 Extension

- StateMachine (from UML2)

8.6.50.2 Generalizations

8.6.50.3 Description

Specifies the behavior of a System or OperationalActivityRealization as a StateMachine - by depicting the transitions (CausesEffects) among Systems and/or OperationalActivityRealizations.

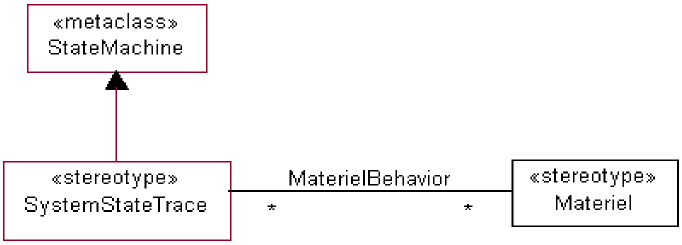


Figure 8.159 - SystemStateTrace

8.6.50.4 Attributes

No additional attributes.

8.6.50.5 Associations

- materiel : Materiel [*]
  Materiel required by this behavior

8.6.50.6 Constraints

[1] Asserts that this SystemState models the behavior of a System, one of its specializations or an OperationalActivityRealization

```
self.owner.getAppliedStereotype('UPDM::OperationalActivityRealization')-
>notEmpty() or
```

```
self.owner.getAppliedStereotype('UPDM::System')->notEmpty() or
```

```
self.owner.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty())
```

8.6.50.7 Semantics

Use the CausesEffect transition stereotype when modeling Effects based systems.

Use the ResultsOfEffect to model the end state of an Effect in the SystemState.

8.6.50.8 Notation

No additional notational requirements.

8.6.51 SystemSystemSoftware

8.6.51.1 Extension

- Association (from UML2)

8.6.51.2 Generalizations

8.6.51.3 Description

Specifies that zero or more SystemSoftware components are configured on one or more Systems.

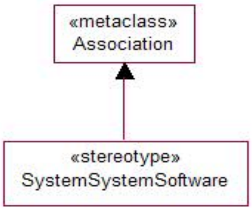


Figure 8.160 - SystemSystemSoftware

8.6.51.4 Attributes

No additional attributes.

8.6.51.5 Associations

No additional associations.

8.6.51.6 Constraints

[1] Asserts that zero or more SystemSoftware components are configured in this System or one of its specializations and that these are the elements associated with this association

```
let e1:uml::Class= self.endType.oclAsType(uml::Class)->at(1) in

let e2:uml::Class= self.endType.oclAsType(uml::Class)->at(2) in

(e1.getAppliedStereotype('UPDM::SystemSoftware')->notEmpty() and

e2.getAppliedStereotype('UPDM::System')->notEmpty() or

e2.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()) or

(e2.getAppliedStereotype('UPDM::SystemSoftware')->notEmpty() and

e1.getAppliedStereotype('UPDM::System')->notEmpty() or

e1.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty())
```

8.6.51.7 Semantics

No additional semantic requirements.

8.6.51.8 Notation

No additional notational requirements.

8.6.52 SystemTask

8.6.52.1 Extension

- Operation (from UML2)

8.6.52.2 Generalizations

8.6.52.3 Description

SystemTasks are the operations defined on Systems and SystemFunctionSpecifications.

In the SystemEventTrace, the messages, TaskInvocations invoke these operations in the receiving Systems or SystemFunctionSpecifications..

SystemTask is used as a CallOperationAction in SystemFunctions.

An action is a named element that is the fundamental unit of executable functionality. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise. An action represents a single step within an activity, that is, one that is not further decomposed within the activity.

A CallOperationAction is an action that transmits an operation call request to the target object, where it may cause the invocation of an associated behavior. The argument values of the action, the DataElements, are available to the execution of the invoked behavior. Any values, also DataElements, returned as part of the reply transmission are put on the result output pins of the CallOperationAction. Upon receipt of the reply transmission, execution of the CallOperationAction is complete.

Individual SystemTasks are depicted in an Activity Diagram by inserting the CallOperationActions that invoke SystemTasks defined on Systems or SystemFunctionSpecifications. See also OperationalTask

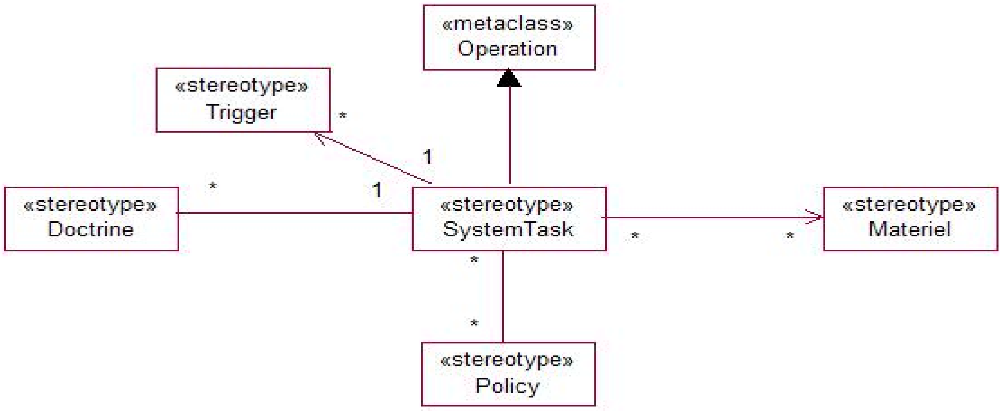


Figure 8.161 - SystemTask

8.6.52.4 Attributes

No additional attributes.

8.6.52.5 Associations

- adheresToPolicy : Policy [*]
  Policies to which this SystemTask adheres

- governedByDoctrine : Doctrine [*]
  Doctrine that governs the SystemTask

- triggeredByEvents : Trigger [*]
  Triggers, callEvents, that trigger this SystemTask

- utilizesMateriel : Materiel [*]
  Materiel utilized by this SystemTask

8.6.52.6 Constraints

[1] Asserts that this SystemTask adheres to zero or more Policies

```
self.adheresToPolicy-> forAll(getAppliedStereotype('UPDM::Policy')->notEmpty())
```

[2] Asserts that there are zero or more Doctrines that govern this SystemTask

```
self.governedByDoctrine->forAll(getAppliedStereotype('UPDM::Doctrine')-
>notEmpty())
```

[3] Asserts that Materiel is utilized by this SystemTask

```
self.utilizesMateriel-> forAll(getAppliedStereotype('UPDM::Materiel')->notEmpty())
```

[4] Asserts that the Triggers of this SystemTask can be stereotyped Trigger, a callEventAction

```
self.triggeredByEvents->forAll(getAppliedStereotype('UPDM::Trigger') ->
notEmpty())
```

[5] Asserts that the Method property of this SystemTask is not null and that its stereotype is SystemFunction or SystemStateTrace or SystemEventTrace

```
self.method->notEmpty() and self.method->forAll(x|
(x.oclIsKindOf(uml::Activity)and
x.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty())

or (x.oclIsKindOf(uml::StateMachine)and
x.getAppliedStereotype('UPDM::OperationalStateTrace')->notEmpty())

or (x.oclIsKindOf(uml::StateMachine)and
x.getAppliedStereotype('UPDM::OperationalEventTrace')->notEmpty())

)
```

[6] Asserts that this SystemTask is owned by a System one of its specializations or a SystemFunctionSpecification.

```
self.owner.getAppliedStereotype('UPDM::SystemFunctionSpecification')-> notEmpty()
or

self.owner.getAppliedStereotype('UPDM::System')->notEmpty() or

self.owner.getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::System') ->notEmpty()
```

8.6.52.7 Semantics

No additional semantic requirements.

8.6.52.8 Notation

No additional notational requirements.

8.6.53 SystemView

8.6.53.1 Extension

8.6.53.2 Generalizations

- ArchitectureView (from AllViews)

8.6.53.3 Description

SystemView is a package that contains information for one or more of the SystemView deliverables or information used for the System View. These packages do not have to be within the same parent package.

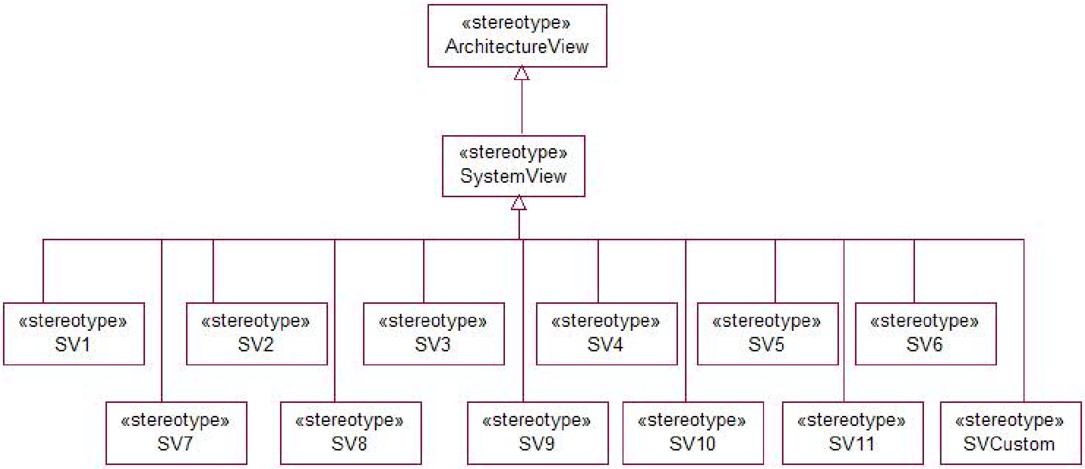


Figure 8.162 - SystemView

8.6.53.4 Attributes

No additional attributes.

8.6.53.5 Associations

No additional associations.

8.6.53.6 Constraints

No additional constraints.

8.6.53.7 Semantics

The stereotyped package indicates to the user that the contents inside that package satisfy the indicated framework product or user defined product. The package here is viewed as an organizing mechanism for views into the architecture, not as a package to hold DoDAF elements. Individual DoDAF elements will typically reside outside of these packages with the diagrams that show the views for these elements inside the package.

8.6.53.8 Notation

No additional notational requirements.

8.6.54 Trigger

8.6.54.1 Extension

- Trigger (from UML2)

8.6.54.2 Generalizations

8.6.54.3 Description

A Trigger initiates an action either in an OperationalActivity or SystemFunction or OperationalStateTrace or a SystemStateTrace.

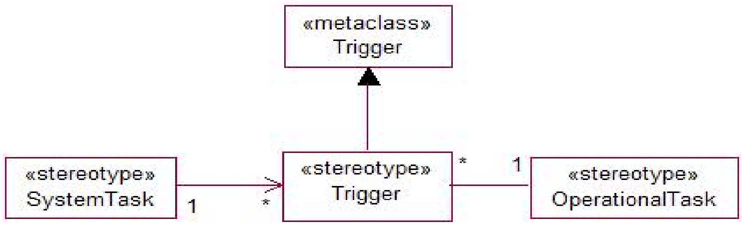


Figure 8.163 - Trigger

8.6.54.4 Attributes

No additional attributes.

8.6.54.5 Associations

- operationaltask : OperationalTask [1]
  Task triggered by the Trigger

8.6.54.6 Constraints

[1] Asserts that the event defined for this trigger is a CallEvent and has an OperationalTask operation.

```
self.event.oclAsType(uml::CallEvent).operation.

getAppliedStereotype('UPDM::OperationalTask')->notEmpty()  or

self.event.oclAsType(uml::CallEvent).operation.

getAppliedStereotype('UPDM::SystemTask')->notEmpty()
```

[2] Asserts that this Trigger's transition has an Effect (OpaqueBehavior) defined.

```
self.owner.ownedElement->asOrderedSet()->

exists(oclAsType(uml::OpaqueBehavior).getAppliedStereotype('UPDM::Effect')-
>notEmpty())
```

8.6.54.7 Semantics

No additional semantic requirements.

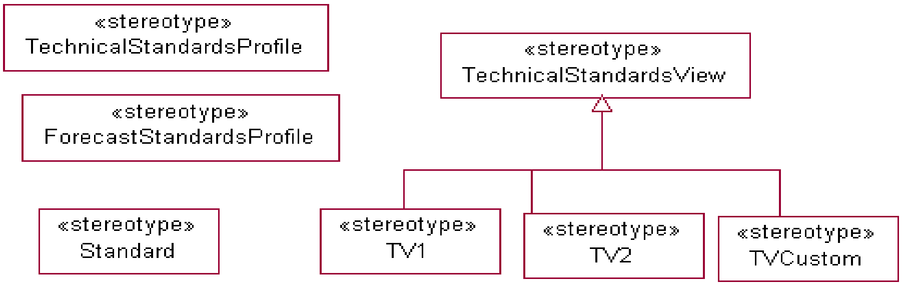
8.6.54.8 Notation

No additional notational requirements.

8.7 TechnicalStandardsView Package

8.7.1 Overview

The TechnicalStandardsView Package contains UML stereotypes that assist the modeler in developing the views defined in the TechnicalStandardsView viewpoint. These stereotypes support the description of the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. The TechnicalStandardsView Package includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture.



8.7.2 ForecastStandardsProfile

8.7.2.1 Extension

- Association (from UML2)

8.7.2.2 Generalizations

8.7.2.3 Description

Asserts that there is an association between a Forecast and a TechnicalStandardsProfile.

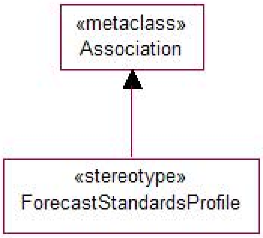


Figure 8.164 - ForecastStandardsProfile

8.7.2.4 Attributes

No additional attributes.

8.7.2.5 Associations

No additional associations.

8.7.2.6 Constraints

[1] Asserts that one end of the association is a Forecast and the other is a TechnicalStandardsProfile

```
(self.endType->at(1).getAppliedStereotype('UPDM::Forecast')->notEmpty() and

self.endType->at(2).getAppliedStereotype('UPDM::TechnicalStandardsProfile')-
>notEmpty()) or

(self.endType->at(2).getAppliedStereotype('UPDM::Forecast')->notEmpty() and

self.endType->at(1).getAppliedStereotype('UPDM::TechnicalStandardsProfile')-
>notEmpty())
```

8.7.2.7 Semantics

No additional semantic requirements.

8.7.2.8 Notation

No additional notational requirements.

8.7.3 Standard

8.7.3.1 Extension

- Class (from UML2)

8.7.3.2 Generalizations

8.7.3.3 Description

A Standard represents an external rule or constraint applied to elements in the System indicating compliance with rules regarding the type of deployments that can be accepted. Standards will typically stem from an external source or reference model.

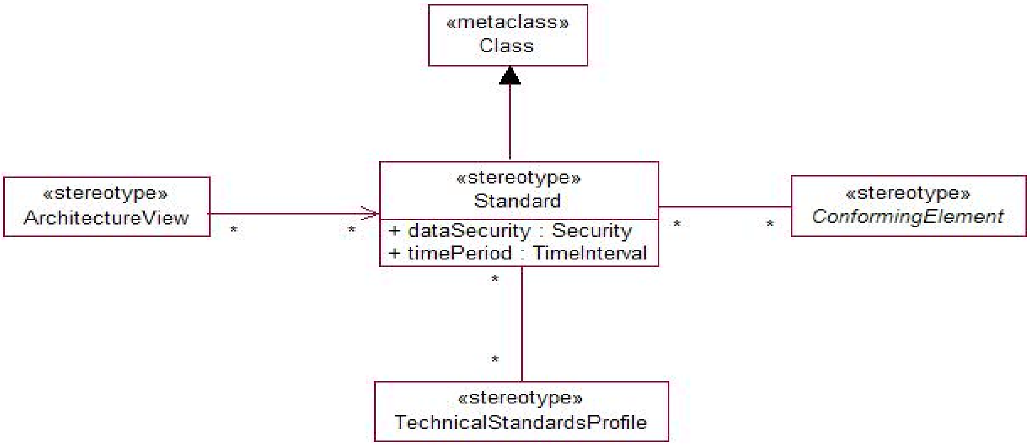


Figure 8.165 - Standard

8.7.3.4 Attributes

- dataSecurity : Security [1]
  Security constraints that govern the Standard

- timePeriod : TimeInterval [1]
  The time period for which the StandardApplies

8.7.3.5 Associations

- conformingElements : ConformingElement [*]
  The collection of Elements that conform to this standard.

- technicalStandardsProfile : TechnicalStandardsProfile [*]
  The assembly of elements reference models and standards that comprise the TV-1

8.7.3.6 Constraints

[1] Asserts that the contents of technicalStandardsProfile are typed TechnicalStandardsProfile.

```
self.technicalStandardsProfile->
forAll(getAppliedStereotype('UPDM::TechnicalStandardsProfile')->notEmpty())
```

[2] Asserts that there is a TimePeriod associated with the Standard.

```
self.timePeriod->forAll(classifier->forAll(name='TimeInterval')) or
```

```
self.timePeriod->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='TimeInterval')))
```

[3] Asserts that the items in the conformingElements are typed ConformingElement or one of its specializations

```
self.conformingElements-> forAll(getAppliedStereotype('UPDM::ConformingElement')-
>notEmpty() or
```

```
getAppliedStereotypes()->collect(allParents())->any(qualifiedName =
'UPDM::ConformingElement') ->notEmpty())
```

[4] Asserts that there is a Security element associated with the Standard.

```
self.dataSecurity->forAll(classifier->forAll(name='Security')) or
```

```
self.dataSecurity->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Security')))
```

8.7.3.7 Semantics

The technical standards for a military architecture typically involve rigorous accreditation and planning. Typically, new technology cannot be implemented without detailed testing and assessment. In a System of Systems architecture, such an approach creates the management issue of making sure all the systems relied on for delivery of a capability are available for use on high performance classified systems. A standard will make it clear if a System has any issues with deployment based on organizational standards.

Any reference models can be associated with the Standard through an ExternalReference of type ‘ReferenceModel.’ The ExternalReference of type ‘Standard’ provides access to the actual document of the Standard.

8.7.3.8 Notation

No additional notational requirements.

8.7.4 TechnicalStandardsProfile

8.7.4.1 Extension

- Class (from UML2)

8.7.4.2 Generalizations

8.7.4.3 Description

The TechnicalStandardsProfile is a collection of Standards, the Service Areas and Services of a ReferenceModel to which they apply.

The TechnicalStandardsProfile works together with Standards and ConformingElement to provide a mapping from ConformingElements through their Standards to the TechnicalStandardsProfile and thus to the Reference Model to which they conform.

A Forecast refers to the standards and reference model through the TechnicalStandardsProfile

Reference models can be applied using UPDMAttributes to attach external references to the Reference Models.

The TechnicalStandardsProfile is the collection of objects that comprise the TV-1

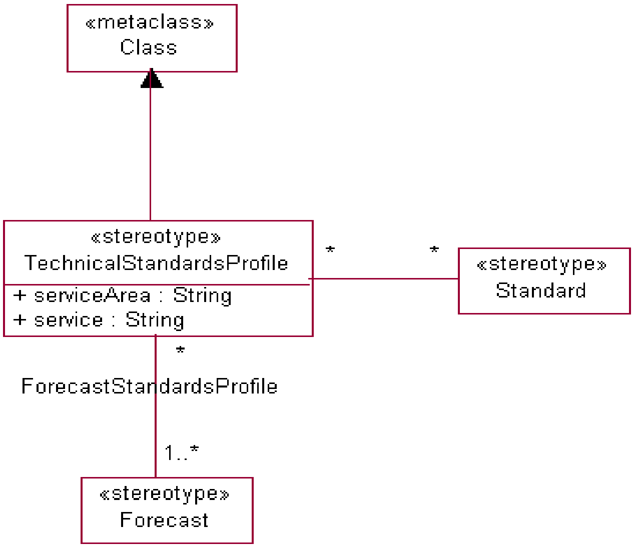


Figure 8.166 - TechnicalStandardsProfile

8.7.4.4 Attributes

- service : String [1]
  The Service designation of a ReferenceModel in which the Conforming element is classified
- serviceArea : String [1]
  The ServiceArea designation of a ReferenceModel in which the Conforming element is classified

8.7.4.5 Associations

- forecast : Forecast [1..*]
  Forecast that refer to this profile. See ForecastStandardsProfile for more details.
- standards : Standard [*]
  The Standards in this profile

8.7.4.6 Constraints

[1] Asserts that the standards in this profile are of type Standard

```
self.standards->forAll(getAppliedStereotype('UPDM::Standard')->notEmpty())
```

[2] Asserts that the association. ForecastStandardsProfile exists between Forecast and TechnicalStandardsProfile

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any
```

```
(getAppliedStereotype('UPDM::ForecastStandardsProfile')-> notEmpty())->notEmpty()
```

8.7.4.7 Semantics

No additional semantic requirements.

8.7.4.8 Notation

No additional notational requirements.

8.7.5 TechnicalStandardsView

8.7.5.1 Extension

8.7.5.2 Generalizations

- ArchitectureView (from AllViews)

8.7.5.3 Description

The TechnicalStandardsView manages the application of all applicable enterprise rules that apply to the technical systems. The TechnicalStandardsProfile provides an overview of the relevant technical standards as applied to the architecture and the Forecast reviews likely changes in those Standards. For the U.S. government, these technical standards often reflect requirements and constraints from models in the Federal Enterprise Architecture. The TechnicalStandardsView provides the mechanisms for using these external reference models inside an architecture to apply technical standards.

The Technical Standards View (TV) provides the technical systems-implementation standards upon which engineering specifications are based, common building blocks are established, and product lines are developed.

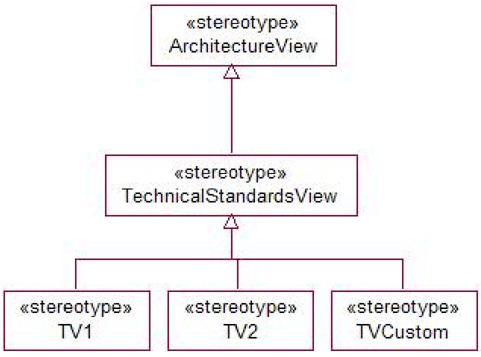


Figure 8.167- TechnicalStandardsView

8.7.5.4 Attributes

No additional attributes.

8.7.5.5 Associations

No additional associations.

8.7.5.6 Constraints

No additional constraints.

8.7.5.7 Semantics

The contents of the TechnicalStandardsView provides a description of emerging standards and the potential impact to all the Views in a given architecture, within a set of time frames. As the technical standards typically arrive as an external constraint, the architect will not always be able to control the format. While the technical standards will be in a document, it is suggested that the entities in the TechnicalStandardsView will help show how the UPDMElements comply with standards.

The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed. The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture.

8.7.5.8 Notation

No additional notational requirements.

8.7.6 TV-1: Technical Standards Profile

8.7.6.1 Extension

8.7.6.2 Generalizations

- TechnicalStandardsView (from TechnicalStandardsView)

8.7.6.3 Description

TechnicalStandardsProfile - Listing of standards that apply to all the Views in a given architecture. The TechnicalStandardsProfile 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.

Primarily, this product is concerned with delineating systems standards rules and conventions that apply to architecture implementations. When the standards profile is tied to the system elements to which they apply, TV-1 serves as the bridge between the SV and TV.

TV-1 consists of the set of systems standards rules that govern systems implementation and operation of that architecture. The technical standards generally govern what hardware and software may be implemented and what system data formats may be used (i.e., the profile delineates which standards may be used to implement the systems, system hardware/ software items, communications protocols, and system data formats).

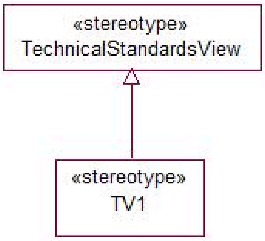


Figure 8.168- TV-1: Technical Standards Profile

8.7.6.4 Attributes

No additional attributes.

8.7.6.5 Associations

No additional associations.

8.7.6.6 Constraints

No additional constraints.

8.7.6.7 Semantics

No additional semantic requirements.

8.7.6.8 Notation

No additional notational requirements.

8.7.7 TV-2: Technical Standards Forecast

8.7.7.1 Extension

8.7.7.2 Generalizations

- TechnicalStandardsView (from TechnicalStandardsView)

8.7.7.3 Description

Technical Standards Forecast - Description of emerging standards and potential impact to all the Views in a given architecture, within a set of time frames.

The Technical Standards Forecast 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.

One of the prime purposes of this product 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.

TV-2 lists emerging or evolving technology standards relevant to the systems covered by the architecture. It contains predictions about the availability of emerging standards, and relates these predictions to the Systems View elements and the time periods that are listed in the SV-8 and SV-9.

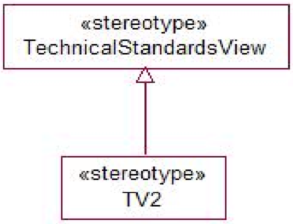


Figure 8.169- TV-2: Technical Standards Forecast

8.7.7.4 Attributes

No additional attributes.

8.7.7.5 Associations

No additional associations.

8.7.7.6 Constraints

No additional constraints.

8.7.7.7 Semantics

No additional semantic requirements.

8.7.7.8 Notation

No additional notational requirements.

8.7.8 TVCustom: Custom Technical View

8.7.8.1 Extension

8.7.8.2 Generalizations

- TechnicalStandardsView (from TechnicalStandardsView)

8.7.8.3 Description

A user-defined view that applies to the TechnicalStandardsView.

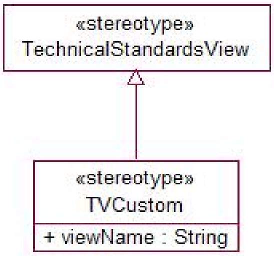


Figure 8.170- TVCustom: Custom Technical View

8.7.8.4 Attributes

- viewName : String [1]
  Provides a name for the custom view so that it can be easily distinguished from other custom TechnicalStandardsViews.

8.7.8.5 Associations

No additional associations.

8.7.8.6 Constraints

No additional constraints.

8.7.8.7 Semantics

No additional semantic requirements.

8.7.8.8 Notation

No additional notational requirements.

9 UPDM Class Library Compliance Level 0

9.1 Overview

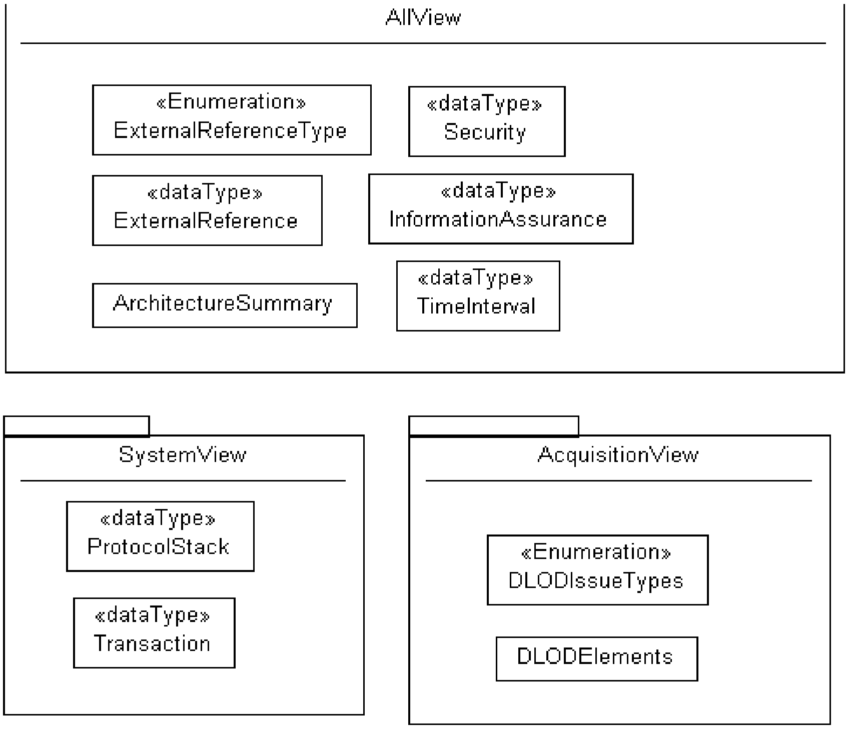


Figure 9.1 - Class Library Main Diagram

9.1.1 Description

A Model Library also called a Class Library is a package that contains model elements that are intended to be reused by other packages. Model libraries are frequently used in conjunction with applied profiles. This is expressed by defining a dependency between a profile and a model library package, or by defining a model library as contained in a profile package. The classes in a model library are not stereotypes and tagged definitions extending the metamodel. A model library is analogous to a class library in some programming languages. When a model library is defined as a part of a profile, it is imported or deleted with the application or removal of the profile. The profile is implicitly applied to its model library. In the other case, when the model library is defined as an external package imported by a profile, the profile requires that the model library be there in the model at the stage of the profile application. The application or the removal of the profile does not affect the presence of the model library elements.

9.2 AcquisitionView Package

9.2.1 Overview

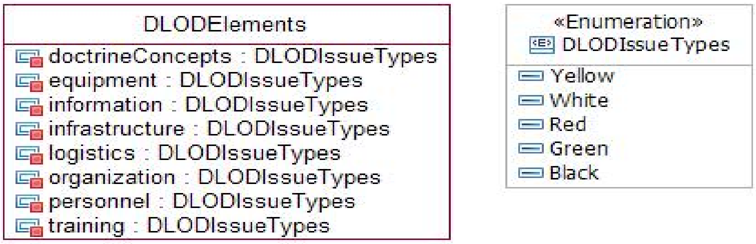

DLODElements
doctrineConcepts : DLODIssueTypes
equipment : DLODIssueTypes
information : DLODIssueTypes
infrastructure : DLODIssueTypes
logistics : DLODIssueTypes
organization : DLODIssueTypes
personnel : DLODIssueTypes
training : DLODIssueTypes
«Enumeration»
DLODIssueTypes
Yellow
White
Red
Green
Black

Figure 9.2 - AcquisitionView Library Elements

9.2.2 DLODElements Class

9.2.2.1 Extension

9.2.2.2 Generalizations

9.2.2.3 Description

Defense Logistics Operation Centre has the attributes to show the status of a particular Capability in terms of whether it is available. The attributes used in the pie charts to illustrate project maturity at lifelines.

9.2.2.4 Attributes

- doctrineConcepts : DLODIssueTypes [1]
- equipment : DLODIssueTypes [1]
- information : DLODIssueTypes [1]
- infrastructure : DLODIssueTypes [1]
- logistics : DLODIssueTypes [1]
- organization : DLODIssueTypes [1]
- personnel : DLODIssueTypes [1]
- training : DLODIssueTypes [1]

9.2.2.5 Associations

No additional associations.

9.2.2.6 Constraints

No additional constraints.

9.2.2.7 Semantics

No additional semantic requirements.

9.2.2.8 Notation

No additional notational requirements.

9.2.3 DLODIssueTypes Enumerated Type

9.2.3.1 Description

Indicators of the status of DLODElements as they apply to a Project within the context of a specific Milestone.

9.2.3.2 Literals

Table 9.1 - DLODIssueTypes Enumeration Literals

Literal	Definition
Yellow	there are outstanding areas of concern, but there are plans in place to address these concerns
White	the DLOD status is not known
Red	an urgent issue that requires changes in project plan
Green	no areas of concern
Black	the DLOD status is not required

9.3 AllView Package

9.3.1 Overview

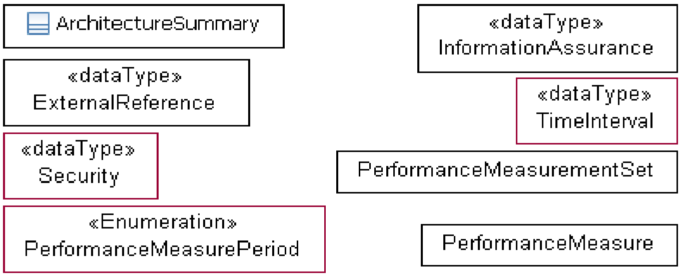


Figure 9.3 - AllView Package Library Elements

9.3.2 ArchitectureSummary Class

9.3.2.1 Extension

9.3.2.2 Generalizations

9.3.2.3 Description

An ArchitectureSunmmary is specification of a system of systems at a technical level that also provides the business context for the system of systems.

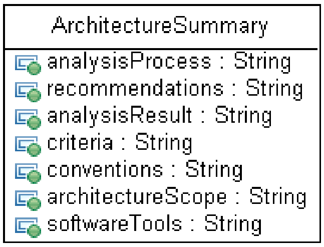


Figure 9.4 - ArchitectureSummary

9.3.2.4 Attributes

- analysisProcess : String [1]
  A description of any analytical processes or procedures used to generate analysis results.
- analysisResult : String [1]
  A description of any results from the analytical processes used for the architectural description project.
- architectureScope : String [1]
  A description of the core mission for the enterprise addressed by the architecture.
- conventions : String [1]
  A description of the goals for the architecture project.
- criteria : String [1]
  The vision that drives the goals for the architecture project.
- recommendations : String [1]
  Recommendations generated from the architecture descriptions.
- softwareTools : String [1]
  A description of the software tools and applications to be utilized by the project developing the systems.

9.3.2.5 Associations

No additional associations.

9.3.2.6 Constraints

No additional constraints.

9.3.2.7 Semantics

No additional semantic requirements.

9.3.2.8 Notation

No additional notational requirements.

9.3.3 ExternalReference DataType

9.3.3.1 Extension

9.3.3.2 Generalizations

9.3.3.3 Description

A triple consisting of name, type, and URI that identifies and refers to a reference that may be external to the model. (See UPDMAttributes)

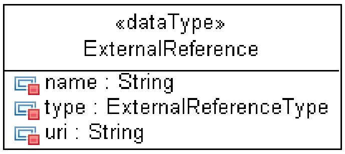


Figure 9.5 - ExternalReference

9.3.3.4 Attributes

- name : String [1]
  The name of the external reference

- type : ExternalReferenceType [1]
  A type designation for the reference

- uri : String [1]
  A unique reference identifier

9.3.3.5 Associations

No additional associations.

9.3.3.6 Constraints

No additional constraints.

9.3.3.7 Semantics

No additional semantic requirements.

9.3.3.8 Notation

No additional notational requirements.

9.3.4 ExternalReferenceType Enumerated Type

9.3.4.1 Description

These are commonly used types of ExternalReferences.

9.3.4.2 Literals

Table 9.2 - ExternalReference Type Enumeration Literals

Literal	**Definition**
CDD	Capability Development Document
CPD	Capability Production Document
CRD	Capstone Requirements Document
Custom	Custom
DOC	Doctrine
EXE	External Executable Process
ICD	Initial Capabilities Document
KPP	Key Performance Parameters
ORG	Org Chart
RM	Reference Model
STD	Standard
SRD	System Requirements Document
VIEWSPEC	Viewpoint Specification Constraints
Image	A URI for a specific Image
CDD	Capability Development Document
CPD	Capability Production Document
CRD	Capstone Requirements Document
Custom	A Custom descriptor
Doctrine	Doctrine
EXE	External Executable Process
ICD	Initial Capabilities Document
KPP	Key Performance Parameters
ORG	Org Chart
RM	Reference Model
STD	Standard
SRD	System Requirements Document
VIEWSPEC	Viewpoint Specification

Table 9.2 - ExternalReference Type Enumeration Literals

COI	Critical Operational Issue
Security	Reference to Security Documents, Standards and Procedures
URI	Universal Resource Identifier

9.3.5 InformationAssurance DataType

9.3.5.1 Extension

9.3.5.2 Generalizations

9.3.5.3 Description

A message is related to an InformationExchange. InformationAssurances are applied to the dataExchanges and informationExchanges.

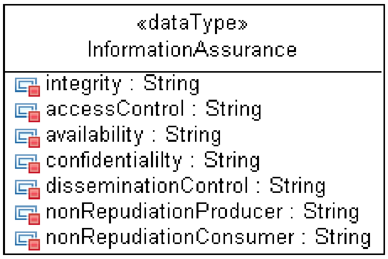

«dataType»
InformationAssurance
integrity : String
accessControl : String
availability : String
confidentialilty : String
disseminationControl : String
nonRepudiationProducer : String
nonRepudiationConsumer : String

Figure 9.6 - InformationAssurance

9.3.5.4 Attributes

- accessControl : String [1]
  A string used to control access to data

- availability : String [1]
  A ratio of the expected value of the uptime of a system to the aggregate of the expected values of up and down time

- confidentiality : String [1]
  A string indicating the nature of the information in terms of who needs to know

- disseminationControl : String [1]
  A string indicating dissemination policy and restrictions

- integrity : String [1]
  Indicates whether the information is in a fragment

- nonRepudiationConsumer : String [1]
  The actual consumer of the information, not an intermediate broadcaster or other consumer

- nonRepudiationProducer : String [1]
  The actual producer of the information, not an intermediate broadcaster or other producer

9.3.5.5 Associations

No additional associations.

9.3.5.6 Constraints

No additional constraints.

9.3.5.7 Semantics

No additional semantic requirements.

9.3.5.8 Notation

No additional notational requirements.

9.3.6 Security DataType

9.3.6.1 Extension

9.3.6.2 Generalizations

9.3.6.3 Description

A security domain has one or more resources. Each resource may have zero or more vulnerabilities. The risk of vulnerability is the effect it will cause to a resource if a loss occurs because of a threat that is manifested. The threats exploit one or more vulnerability. Safeguards mitigate vulnerabilities.

Mitigation responds proactively to protect assets from known vulnerabilities and threats.

A Security is a set of attributes related to a security profile that can be applied to any number of InformationExchanges, or DataElements.

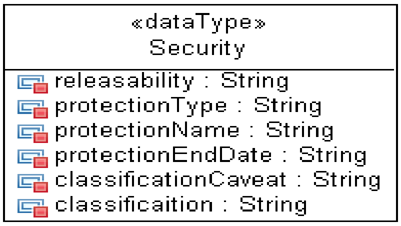


Figure 9.7- Security

9.3.6.4 Attributes

- classification : String [1]
  The classification for this item. An implementation tuned to a specific organization might make this an enumeration to reflect a specific security taxonomy.

- classificationCaveat: String [1]
  Special circumstances for release of information.

- protectionEndDate : String [1]
  Date when the protection on this information lapsed. If blank, the information must still comply by the security standards.

- protectionName : String [1]
  A name for the security protection.

- protectionType : String [1]
  The nature of the protection on the information or data.

- releasability: String [1]
  A description of how the information can be released.

9.3.6.5 Associations

No additional associations.

9.3.6.6 Constraints

No additional constraints.

9.3.6.7 Semantics

The defense domain requires sophisticated security models and this is a part of the domain that is evolving rapidly with publish/subscribe and service oriented architectures. The structure here for security can apply to any number of UPDM elements and is not constrained to the associations emphasized here. The associations emphasized here enable the delivery of existing DoDAF and MODAF work products.

A security domain has one or more resources. Each resource may have zero or more vulnerabilities. The risk of a vulnerability is the effect it will cause to a resource if a loss occurs because of a threat that is manifested. The threats exploit one or more vulnerability. Safeguards mitigate vulnerabilities.

9.3.6.8 Mitigation responds proactively to protect assets from known vulnerabilities and threatsNotation

No additional notational requirements.

9.3.7 TimeInterval DataType

9.3.7.1 Extension

9.3.7.2 Generalizations

9.3.7.3 Description

TimeInterval is a DataType that has a begin and end date in String form. A period of time, defined by start and end dates - sometimes termed an “Epoch” in the MOD. TimePeriods may overlap. A single TimePeriod may be assigned to many different elements in the Model.

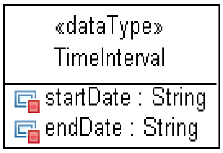


Figure 9.8 - TimeInterval

9.3.7.4 Attributes

- startDate : String [1]
  Starting date of the interval.

- endDate : String [1]
  Ending date of the interval

9.3.7.5 Associations

No additional associations.

9.3.7.6 Constraints

No additional constraints.

9.3.7.7 Semantics

No additional semantic requirements.

9.3.7.8 Notation

No additional notational requirements.

9.4 SystemView Package

9.4.1 Overview

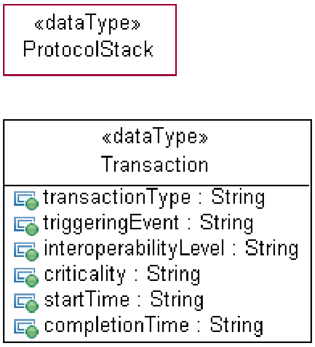


Figure 9.9 - SystemView Package Library Elements

9.4.2 ProtocolStack DataType

9.4.2.1 Extension

9.4.2.2 Generalizations

9.4.2.3 Description

ProtocolStack is a DataType that can be customized to describe capabilities and standards for CommunicationPorts. It is a holder that will be extended to fit requirements.

9.4.2.4 Attributes

No additional attributes.

9.4.2.5 Associations

No additional associations.

9.4.2.6 Constraints

No additional constraints.

9.4.2.7 Semantics

Users will define subtypes with details of specific protocols to be applied to CommunicationPorts..

9.4.2.8 Notation

No additional notational requirements.

9.4.3 Transaction DataType

9.4.3.1 Extension

9.4.3.2 Generalizations

9.4.3.3 Description

A construct to link the information exchange and data flow used by the architecture. If the architecture extends to deal with Transactions that involve items other than information, this description of Transaction would also apply to those exchanges. Typically, a Transaction will have an association with both an InformationExchange or a DataFlow, but that is not made a constraint to allow users to allow Transactions to be used for other flows.

9.4.3.4 Attributes

- completionTime : String [1]
  Time transaction completed
- criticality : String [1]
  Criticality of the transaction
- interoperabilityLevel : String [1]
  Interoperability evaluation measurement - e.g., NCOIC's NCAT
- startTime : String [1]
  Time transaction starts
- transactionType : String [1]
  User defined type of transaction
- triggeringEvent : String [1]
  Event that triggers the transaction

9.4.3.5 Associations

No additional associations.

9.4.3.6 Constraints

No additional constraints.

9.4.3.7 Semantics

The Transaction connects the InformationExchange and DataFlow with a number of attributes describing the exchange.

When an InformationExchange stereotype is created, it should include a static variable of type Transaction. When a DataExchange stereotype is created, it should include a static variable of type Transaction. When a Transaction instance specification is created, and when the user assigns the InformationExchange and DataExchange to the stereotype properties, the default value of the respective statics should be set to the Transaction Instance Specification.

9.4.3.8 Notation

Transaction applies to DataExchange or InformationExchange.

10 UPDM Profile Specification Compliance Level 1

10.1 Overview

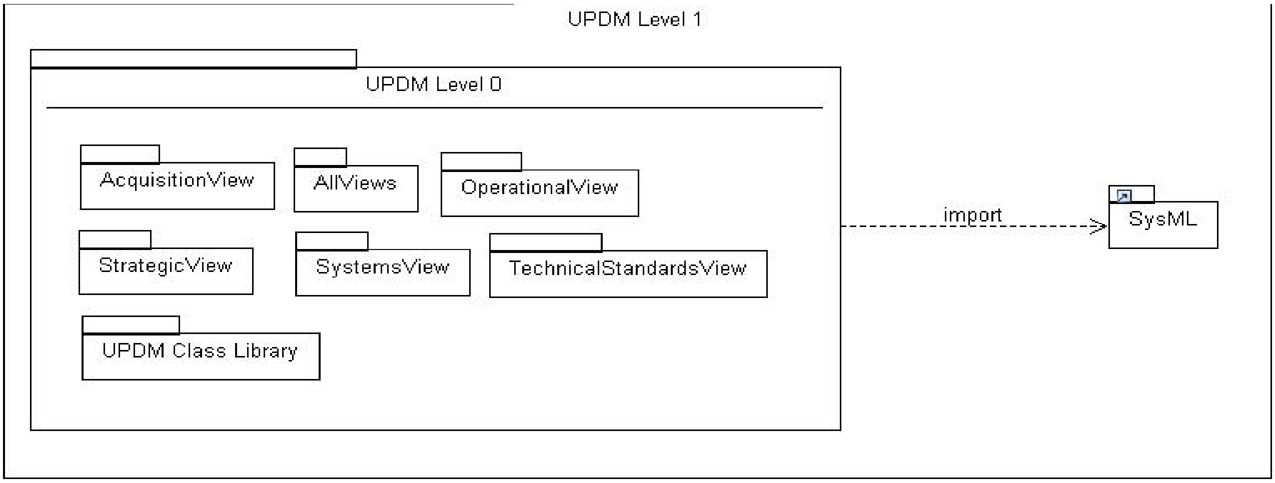


Figure 10.1 - UPDM Level 1 Compliance Packages

UPDM Level 1 compliance mapping alternatives to the Level 0 specification are shown in Table 10.1. UPDM Level 0 stereotypes that extend the meta-classes defined in the UML subset, UML4SysML are shown in column 1. These same stereotypes specialize the respective the stereotypes defined in SysML shown in column 2. Thus, these stereotypes exhibit characteristics of UPDM Level-0 and their SysML generalizations..

The purpose of this section is to define the UPDM extensions associated with the Level 1 Compliance. Level 1 Compliance further extends the element definitions defined in Section 4 using the OMG SysML™ stereotype definitions and extensions. The use of the Level 1 Compliance enables UPDM to leverage the SysML features highlighted below. It is also intended to facilitate the integration of DoDAF and MODAF architecture and engineering models. The Level 1 Compliance provides for use of all SysML (non-restricted mode) features that includes support for structural, behavioral, parametric and requirements aspects of DoDAF and MODAF models. The extensions enable the following:

- Use of SysML blocks as the modular unit of structure to represent operational nodes, system nodes, systems, and other structural elements. This in turn enables the use of other SysML features such as flow ports, item flows, and value properties with units and distributions.
- Use of SysML activities which includes extensions that support continuous flow modeling, activity hierarchies, and additional semantics to support enhanced functional flow block diagrams.
- Use of SysML requirements that enable text based requirements to be captured and traced to other model elements using the satisfy, derive, verify and refine relationships.
- Use of SysML parametrics that enable the integration of engineering analysis with the architecture models. In particular, the performance parameters in an SV-7 can be captured in parametric equations.
- Use of SysML allocations to support various types of mappings. These are useful for modeling a variety of DoDAF and MODAF products such as an SV-5 that maps systems to capabilities or system functions to operational activities.

- In addition, other features of SysML, such as the more formalized use of diagram frames and headers, labeling of compartments, and other features can be leveraged.

The following figures indicate the way in which SysML features can be leveraged with UPDM Compliance Level 1.

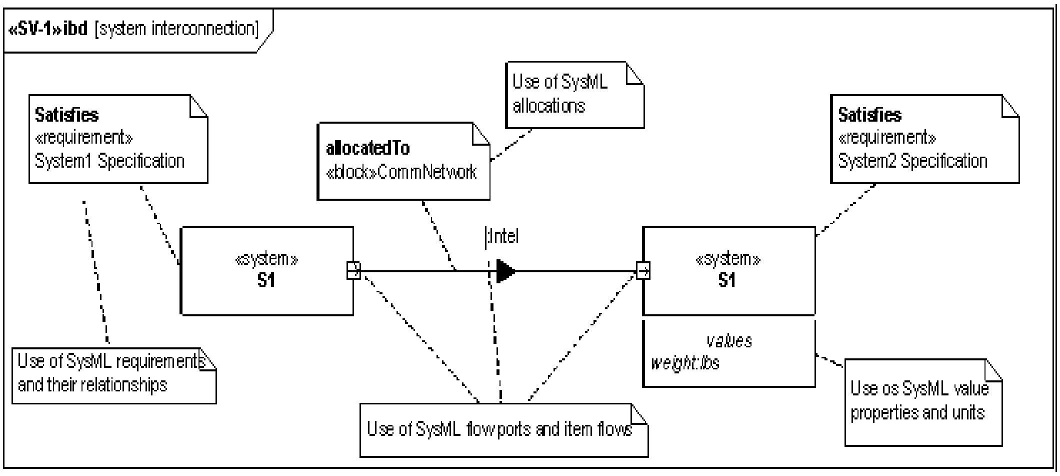


Figure 10.2 - Leveraging SysML Features with Compliance Level 1

10.2 Level 1 Mapping Table

The Level 1 compliance mapping alternatives to the level 0 specification are defined the Table 6-1 below. The Level 0 Stereotypes extend the meta-classes defined in Level 0. These same stereotypes will realize the meta-classes defined in SysML. Thus, these eleven stereotypes have multiple inheritance.

Table 10.1 - Level 1 Specification Element Mapping Table

Stereotype	**L1 Abstract Syntax**
UPDM::AllViews::Requirement	SysML::Requirements::Requirement
UPDM::AllViews::Resource	SysML::Blocks::Block
UPDM::OperationalView::InformationElement	SysML::Blocks::Block
UPDM::OperationalView::InformationExchange	SysML::PortsAndFlows::ItemFlow
UPDM::OperationalView::Materiel	SysML::Blocks::Block
UPDM::OperationalView::OperationalNode	SysML::Blocks::Block
UPDM::OperationalView::OrganizationalRole	SysML::Blocks::BlockProperty
UPDM::SystemsView::DataElement	SysML::Blocks::Block
UPDM::SystemsView::DataExchange	SysML::PortsAndFlows::ItemFlow
UPDM::SystemsView::System	SysML::Blocks::Block

Level 1 compliance allows full use of the SysML features applied to all Level 0 model elements, limited by the UPDM Level-0 constraints and SysML specific feature constraints with the UPDM Level-0 constraints taking precedence in the case of conflict.

10.3 AllViews Package

10.3.1 Overview

The SysML profile is imported into the UPDM Level 0 to form the basis for UPDM Level 1. The AllViews Package contains UML stereotypes that assist the modeler in developing the views defined in the AllViews viewpoint. These stereotypes support the description of some overarching aspects of an architecture that relate to all three views. The AllViews Package includes the subject area and time frame for the architecture. The setting in which the architecture exists comprises the interrelated conditions that compose the context for the architecture. These conditions include doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations (CONOPS); scenarios; and environmental conditions.

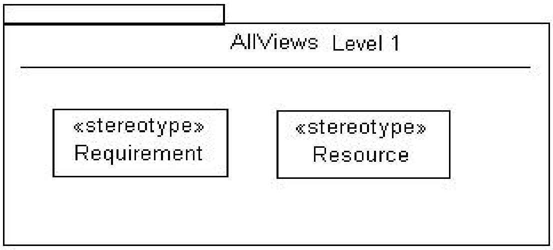


Figure 10.3 - AllViews

10.3.2 Requirement

10.3.2.1 Extension

- Class (from UML2)

10.3.2.2 Generalizations

- Requirement (from SysML::Requirements)

10.3.2.3 Description

A Requirement associates a textual statement of a requirement with one or more elements in the model and zero or more external references.

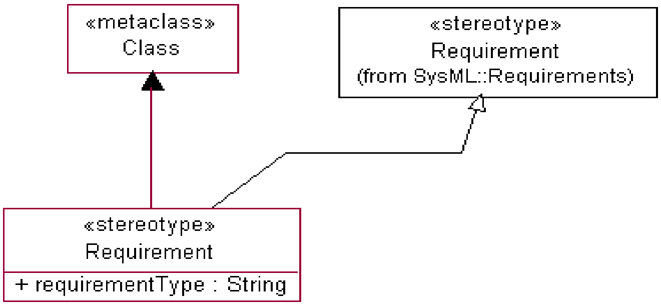


Figure 10.4 - Requirement

10.3.2.4 Attributes

- requirementType : String [1]

10.3.2.5 Associations

No additional associations.

10.3.2.6 Constraints

No additional constraints.

10.3.2.7 Semantics

A requirement traces a requirement to one or more elements that satisfy the requirement or specify the requirement in detail such as a Use Case or OperationalActivity.

10.3.2.8 Notation

No additional notational requirements.

10.3.3 Resource

10.3.3.1 Extension

- Class (from UML2)

10.3.3.2 Generalizations

- Block (from SysML::Blocks)

10.3.3.3 Description

A Resource is something that is able to supply functionality, information or material. It can be a SystemsNode that represent facilities, platforms, units, and locations or OrganizationalResource that can contribute towards fulfilling a Capability.

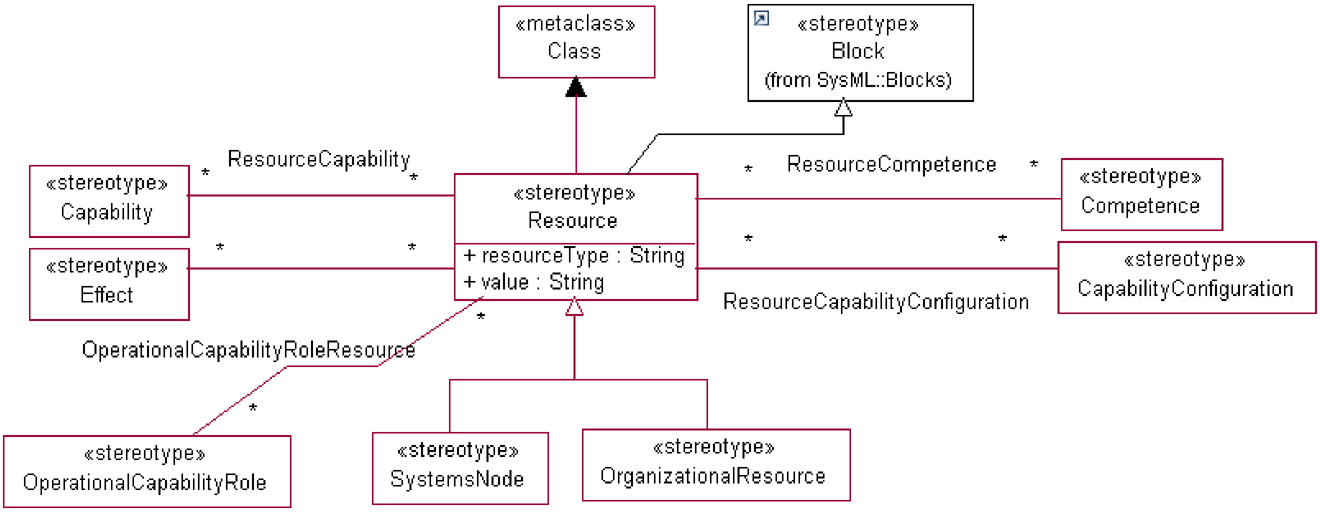


Figure 10.5 - Resource

10.3.3.4 Attributes

- resourceType : String [1]
  The type of resource - organization, organization type, human, etc.

- value : String [1]
  The value of the resource to the enterprise. This value is used in risk assessment and mitigation calculations. It could be a monetary amount, a critical supply scale, etc.

10.3.3.5 Associations

- capability : Capability [*]
  See ResourceCapability for more details.

- capabilityConfiguration : CapabilityConfiguration [*]
  See ResourceCapabilityConfiguration for more details.

- competence : Competence [*]
  See ResourceCompetence for more details.

- effect : Effect [*]
  The effects that affect the resource.

- operationalCapabilityRole : OperationalCapabilityRole [*]
  The OperationalCapabilityRoles that are filled by this Resource

10.3.3.6 Constraints

[1] Asserts that zero or more effects affect this resource.

```
self.effect-> forAll(getAppliedStereotype('UPDM::Effect')->notEmpty())
```

[2] Asserts that a ResourceCapability association exists between a Resource and the Capabilities that it provides

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::ResourceCapability')-> notEmpty())->notEmpty()
```

[3] Asserts that a ResourceCapabilityConfiguration association exists between Resource and CapabilityConfiguration

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::ResourceCapabilityConfiguration')-> notEmpty())->notEmpty()
```

[4] Asserts that a ResourceCompetence association exists between Resource and the Competence that it provides

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::ResourceCompetence')-> notEmpty())->notEmpty()
```

[5] Asserts that there is an association between a Competence and the OperationalCapabilityRole that requires it.

```
self.getAllAttributes()->asOrderedSet()->select(association->notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::OperationalCapabilityRoleResource')-> notEmpty())->notEmpty()
```

10.3.3.7 Semantics

No additional semantic requirements.

10.3.3.8 Notation

No additional notational requirements.

10.4 OperationalView Package

10.4.1 Overview

The OperationalView Package contains UML stereotypes that assist the modeler in developing the views defined in the Operational View viewpoint. These stereotypes support the description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. The OperationalView Package also includes elements for the identification of the operational nodes, assigned tasks and activities, and information flows required between nodes.

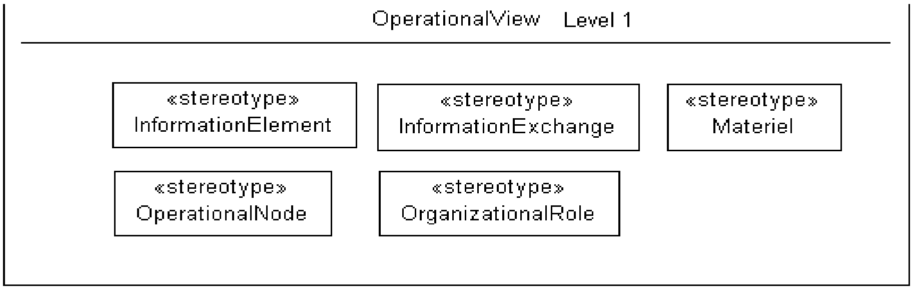


Figure 10.6 - OperationalView Package

10.4.2 InformationElement

10.4.2.1 Extension

10.4.2.2 Generalizations

- ConformingElement (from AllViews)
- Block (from SysML::Blocks)

10.4.2.3 Description

The collection of InformationElements and their performance attributes such as timeliness, quality, and quantity values. DoDAF defines InformationElement as "... a formalized representation of information subject to an operational process (e.g., the information content that is required to be exchanged between nodes). In contrast an information exchange is comprised of the Needline, the information element, and other attributes such as Information Assurance attributes. The specific attributes of the information elements included are dependent on the objectives of the specific architecture effort and include the information scope, accuracy, and language. An information element may be used in one or more information exchanges

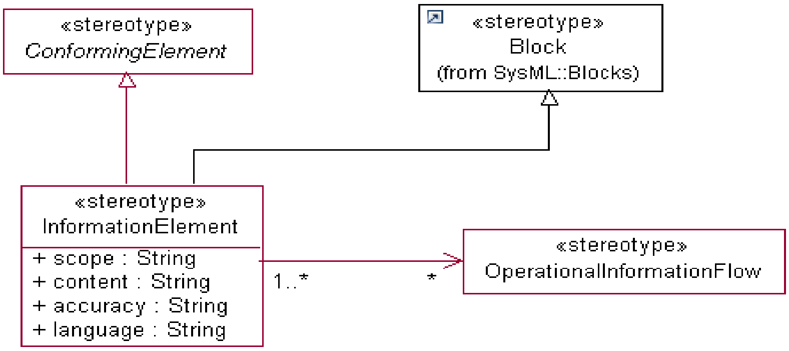


Figure 10.7 - InformationElement

10.4.2.4 Attributes

- accuracy : String [1]
  The required accuracy of the InformationElement.
- content : String [1]
  The content of the InformationElement.
- language : String [1]
  The language in which the information is expressed. For example, this could be a natural language, a computer language, mathematics, scientific, engineering or other discipline's nomenclature.
- scope : String [1]
  A description of the scope of the InformationElement.

10.4.2.5 Associations

- operationalInformationFlow : OperationalInformationFlow [*]
  The OperationalInformationFlows associated with this InformationElement.

10.4.2.6 Constraints

[1] Asserts that this InformationElement is associated with an OperationalActivityFlow.

```
self.operationalInformationFlow.getAppliedStereotype('UPDM::OperationalInformation
Flow')->notEmpty()
```

10.4.2.7 Semantics

The identity of the individual InformationElements and their attributes must be documented in detail for many military missions. The InformationElements must be combined with InformationExchanges along with the InformationAssurance and Security definitions. InformationElements can also have relationships and constraints with other InformationElements outlined in a class diagram.

10.4.2.8 Notation

No additional notational requirements.

10.4.3 InformationExchange

10.4.3.1 Extension

- InformationFlow (from UML2)

10.4.3.2 Generalizations

- ItemFlow (from SysML:PortsAndFlows)

10.4.3.3 Description

InformationExchange is an act of exchanging InformationElements between two distinct operational Nodes and the characteristics of the act, including the InformationElements that need to be exchanged and the attributes associated with the informationElement (e.g., Scope), as well as attributes associated with the exchange (e.g., Transaction Type). A Needline represents one or more InformationExchanges between the same two Nodes as supplier and consumer of the InformationExchange.

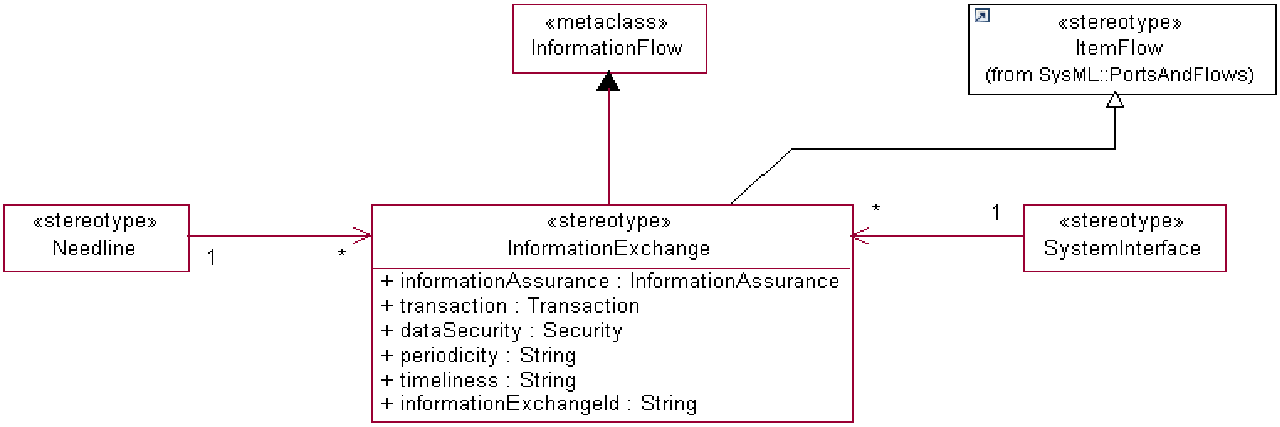


Figure 10.8 - InformationExchange

10.4.3.4 Attributes

- dataSecurity : Security [1]
  The Security elements that constrain this InformationExchange
- informationAssurance : InformationAssurance [1]
  The InformationAssurance elements that apply to this InformationExchange
- informationExchangeId : String [1]
  An identifier such as a unique id or serial number
- periodicity : String [1]
  A statement of the operational period of the InformationExchange
- timeliness : String [1]
  The schedule constraints for this InformationExchange
- transaction : Transaction [1]
  Specifies the Transaction if this InformationExchange is governed by transaction semantics

10.4.3.5 Associations

No additional associations.

10.4.3.6 Constraints

[1] Asserts that all the activity edges that realize this InformationFlow are either OperationalControlFlows or OperationalInformationFlows are owned by an OperationalActivity

```
self.realizingActivityEdge->
forAll((getAppliedStereotype('UPDM::OperationalControlFlow')->notEmpty() or

getAppliedStereotype('UPDM::OperationalInformationFlow')->notEmpty()) and

owner.getAppliedStereotype('UPDM::OperationalActivity')->notEmpty())
```

[2] Asserts that this InformationAssurance applies to the InformationExchange.

```
self.informationAssurance->forAll(classifier->forAll(name='InformationAssurance'))
or

self.informationAssurance->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='InformationAssurance')))
```

[3] Asserts that the stereotype of transaction is Transaction

```
self.transaction->forAll(classifier->forAll(name='Transaction')) or

self.transaction->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Transaction')))
```

[4] Asserts that the stereotype of realizingConnector or its type's stereotype is Needline

```
self.realizingConnector->notEmpty() and

self.realizingConnector->forAll(getAppliedStereotype('UPDM::Needline')->notEmpty()
or

    type.getAppliedStereotype('UPDM::Needline')->notEmpty())
```

[5] Asserts that the stereotype of dataSecurity is Security

```
self.dataSecurity->forAll(classifier->forAll(name='Security')) or

self.dataSecurity->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Security')))
```

[6] Asserts that the Interaction that contains an InformationExchange must be an OperationalEventTrace

```
self.realizingMessage.getAppliedStereotype('UPDM::TaskInvocation')->notEmpty()
```

[7] Asserts that this InformationExchange occurs between two OperationalNodes..

```
(self.target->any(getAppliedStereotype( 'UPDM::OperationalNode') ->notEmpty() ) -
>notEmpty() or

self.target->forAll(getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty() ))

and

(self.source->any(getAppliedStereotype( 'UPDM::OperationalNode') ->notEmpty() ) -
>notEmpty() or

self.source->forAll(getAppliedStereotypes()->collect(allParents())->
any(qualifiedName = 'UPDM::OperationalNode') ->notEmpty() ))
```

[8] Asserts that the value provided by an OperationalActivity invoked by an InformationExchange is an InformationElement.

```
self.conveyed.oclAsType(uml::Classifier).getAppliedStereotype('UPDM::InformationEl
ement')->notEmpty()
```

10.4.3.7 Semantics

A set of InformationExchanges flowing from a providing OperationalNode to a consuming OperationalNode implies that a Needline exists between those two nodes with a directionality implied by the direction of the flow.

In the case where the InformationExchange is realized by a message, the signature of the message is an OperationalTask and the arguments of the message must correspond to the conveyed information element of the InformationExchange. The Behavior of the OperationalTask is the receiving OperationalActivity.

When a CompositeStructureDiagram is used to describe the structure of the encompassing OperationalNode, the connector between the parts that correspond to the lifelines can be assigned by the connector property.

If the InformationExchange ends are both OccurrenceSpecifications, then the connector must go between the Parts represented by the Lifelines of the two InformationExchange ends.

In the case where the InformationExchange is realized by Activity edges, the Activity edges will be stereotyped either OperationalInformationFlow or OperationalControlFlow.

OperationalInformationFlow

In this case, the ends of the flow are object pins whose type is stereotyped InformationElement. The activity elements connected by the OperationalInformationFlow must be either a CallBehaviorAction whose behavior is an OperationalActivity, or a CallOperationAction whose operation is an OperationalTask.

OperationalControlFlow

The source of the OperationalControlFlow is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an OperationalTask, or an InitialNode.

The target of the OperationalControlFlow is one of a CallBehaviorAction whose behavior is an OperationalActivity, a CallOperationAction whose operation is an ActivityTask, or a FinalNode.

10.4.3.8 Notation

No additional notational requirements.

10.4.4 Materiel

10.4.4.1 Extension

- Class (from UML2)

10.4.4.2 Generalizations

- Block (from SysML::Blocks)

10.4.4.3 Description

Materiel is tied to elements, where mechanisms may be used to represent Systems that support OperationalActivities. In addition, further materiel detail may be related to the activities, by relating those activities to the SystemFunctions that are executed by Systems that automate them (wholly or partially). An OperationalCapabilityRealization is defined in terms of the OperationalActivities. Materiel may be associated with UPDM behaviors: an OperationalActivity, a SystemFunction, an EventTrace and StateTrace and with other UPDM elements: an OperationalCapabilityRealization, an OperationalCapability, an OperationalNode, an OperationalNodeSpecification, a System, a SystemInterface, or a CapabilityConfiguration.

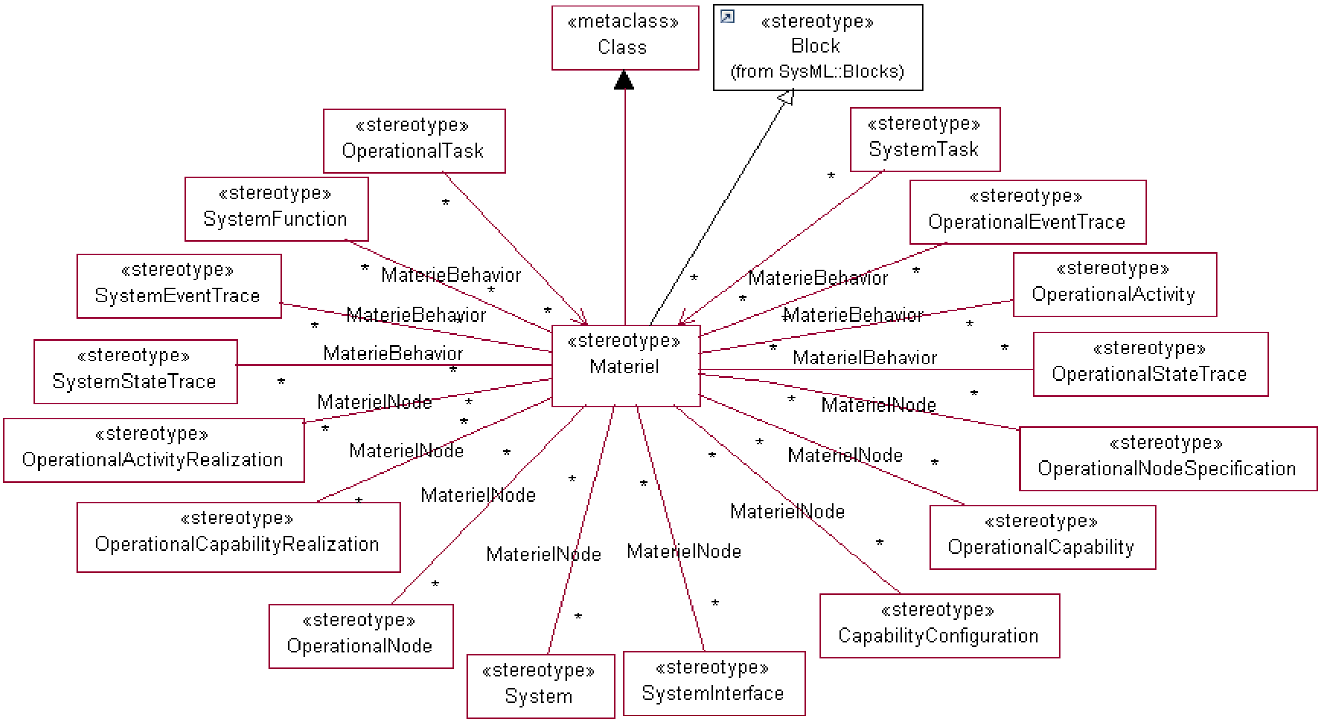

«metaclass» Class
«stereotype» Block (from SysML::Blocks)
«stereotype» OperationalTask
«stereotype» SystemTask
«stereotype» SystemFunction
«stereotype» OperationalEventTrace
«stereotype» SystemEventTrace
«stereotype» OperationalActivity
«stereotype» SystemStateTrace
«stereotype» Materiel
«stereotype» OperationalStateTrace
«stereotype» OperationalActivityRealization
«stereotype» OperationalNodeSpecification
«stereotype» OperationalCapabilityRealization
«stereotype» OperationalCapability
«stereotype» CapabilityConfiguration
«stereotype» OperationalNode
«stereotype» System
«stereotype» SystemInterface
MaterieBehavior
MaterielBehavior
MaterielNode
*

Figure 10.9 - Materiel

10.4.4.4 Attributes

No additional attributes.

10.4.4.5 Associations

- capabilityconfiguration : CapabilityConfiguration [*]
- operationalactivity : OperationalActivity [*]
  Materiel used by this OperationalActivity
- operationalactivityrealization : OperationalActivityRealization [*]
  Materiel used by this OperationalActivityRealization

- operationalcapability : OperationalCapability [*]
  Materiel used by this OperationalCapability
- operationalcapabilityrealization : OperationalCapabilityRealization [*]
  Materiel used by this OperationalCapabilityRealization
- operationaleventtrace : OperationalEventTrace [*]
  Materiel used by this OperationalEventTrace
- operationalnode : OperationalNode [*]
  Materiel used by this OperationalNode
- operationalnodespecification : OperationalNodeSpecification [*]
  Materiel used by this OperationalNodeSpecification
- operationalstatetrace : OperationalStateTrace [*]
  Materiel used by this OperationalStateTrace
- system : System [*]
  Materiel used by this System
- systemeventtrace : SystemEventTrace [*]
  Materiel used by this OperationalEventTrace
- systemfunction : SystemFunction [*]
  Materiel used by this SystemFunction
- systeminterface : SystemInterface [*]
  Materiel used by this SystemInterface
- systemstatetrace : SystemStateTrace [*]
  Materiel used by this SystemStateTrace

10.4.4.6 Constraints

No additional constraints.

10.4.4.7 Semantics

No additional semantic requirements.

10.4.4.8 Notation

No additional notational requirements.

10.4.5 OperationalNode

10.4.5.1 Extension

- Class (from UML2)

10.4.5.2 Generalizations

- Block (from SysML::Blocks)

10.4.5.3 Description

An OperationalNode performs a role or mission; a conceptual architectural element that produces, consumes or processes information. An OperationalNode exists to realize one or more capabilities. Deployment of the capabilities in systems that help to realize those capabilities is allocated to SystemsNodes. An OperationalNode can be a composite structure that is decomposed into finer grained OperationalNodes. To model OperationalNode composition, designate component OperationalNodes as parts of the composite OperationalNode. This is a recursive structure. Level of decomposition is indicated in the decompositionLevel attribute. The level of decomposition, beginning with zero, the default, is recorded in the decompositionLevel attribute. The concept of an actor or resource that realizes an OperationalNode has been made explicit by using OrganizationalResource that plays an OperationalCapabilityRole in the context of an OperationalNode. The OperationalCapabilityRole is an Class that associates Competencies required for that OperationalCapabilityRole.

Information is consumed by an OperationalNode by ingesting information through required DataFlows, exchanged from another OperationalNode, provided DataFlows, that produces the information. If an OperationalNode consumes information it implies that it needs a channel for communication. These channels are called Needlines. They are directional indicating the supplier and consumer of the information exchange.

DoDAF (V1) states: “The relationship between architecture data elements across the System View to the Operational View (OV) can be exemplified as systems are procured and fielded to support organizations and their operations. SV-1 links together the OV and SV by depicting the assignments of systems and systems nodes (and their associated interfaces) to the OperationalNodes (and their associated Needlines) described in OV-2. OV-2 depicts the OperationalNodes representing organizations, organization types, and/or human roles, while SV-1 depicts the physical assets that house OperationalNodes (e.g., platforms, units, facilities, and locations) and the corresponding systems resident at these physical assets and which support the OperationalNodes.”

An OperationalNode is an abstract representation of capabilities that are achieved through the OperationalTasks (the operations on the OperationalNode). Interactions among OperationalNodes are represented as InformationExchanges between the OperationalNodes. These InformationExchanges are accomplished by the execution of an OperationalActivity in the receiving OperationalNode. In an interaction, an OperationalNode may perform one or more OperationalActivities. That group of OperationalTasks defines the role that the OperationalNode plays in the context of that interaction or in the accomplishment of that capability. (See OperationalCapabilityRole)

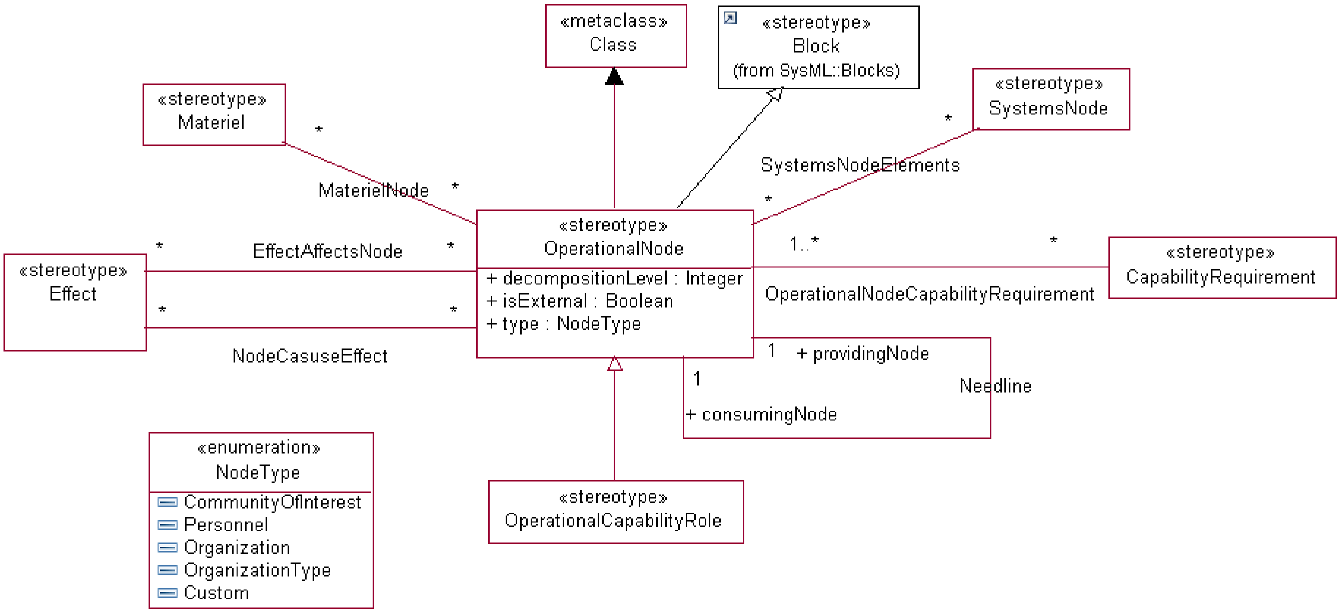


Figure 10.10 - OperationalNode

10.4.5.4 Attributes

- decompositionLevel : Integer [1]
  OperationalNode is a composite element. Level of decomposition of this node. Default value = 0

- isExternal : Boolean [1]
  Is this OperationalNode external to the system being described? Default value = false

- type : NodeType [1]
  An indicator of a type of OperationalNode

10.4.5.5 Associations

- capabilityRequirement : CapabilityRequirement [*]
  See OperationalNodeCapabilityRequirement for more details.

- consumingNode : OperationalNode [1]

- effect : Effect [*]
  See NodeCausesEffect for more details.

- effect : Effect [*]
  See EffectAffectsNode for more details.

- materiel : Materiel [*]
  Materiel required for the OperationalNode

- providingNode : OperationalNode [1]

- systemsnode : SystemsNode [*]

SystemsNode in which the OperationalNode is housed.

10.4.5.6 Constraints

[1] Asserts that there is at least one SystemsNode (a physical asset) that houses this OperationalNode.

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

[2] Asserts that this Node participates in at least one Needline. (If no other Node needs this one or it doesn't need any other Node, then what is its purpose?)

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::Needline')-> notEmpty())->notEmpty()
```

[3] Asserts that an OperationalNode must have at least one element that defines its behavior. One of OperationalEventTrace, OperationalActivity or OperationalStateTrace

```
self.oclAsType(uml::Class).ownedBehavior->asOrderedSet()->exists(a|

a.getAppliedStereotype('UPDM::OperationalThread')->notEmpty() or

a.getAppliedStereotype('UPDM::OperationalEventTrace')->notEmpty()or

a.getAppliedStereotype('UPDM::OperationalState')->notEmpty())
```

[4] Asserts that this Node is required for delivery of an OperationalCapability

```
self.getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::OperationalNodeCapability')-> notEmpty())-
>notEmpty()
```

[5] Asserts that there is at least one OperationalTask defined on an OperationalNode.

```
self.oclAsType(uml::Class).ownedOperation->asOrderedSet() ->

exists(getAppliedStereotype('UPDM::OperationalTask')->notEmpty())
```

[6] Asserts that an OperationalNode realizes interfaces that are OperationalNodeSpecifications, i.e., interfaces

```
self.clientDependency->forAll(spec|

if spec.oclIsKindOf(uml::InterfaceRealization)then

spec.getAppliedStereotype('UPDM::RealizedOperationalSpecification')->notEmpty()

else

true

endif)
```

10.4.5.7 Semantics

To specify the EventTrace of an OperationalNode, create an Interaction in the OperationalNode and stereotype it with EventTrace. Then create a Sequence diagram to populate it. (OV 6-c)

To specify the OperationalActivities of an OperationalNode, create an Activity in the OperationalNode and stereotype it with OperationalActivity. Then create an Activity diagram to populate it.

To determine the required DataFlows (Needlines) of an OperationalNode, refer to ClientDependencies that are stereotyped as Needlines.

Provided Dataflows are a stereotype property, a metaclass association with Needline.

Operations defined on an OperationalNode may be stereotyped as OperationalTask. OperationalTasks are defined as stereotyped operations on OperationalNodes.

10.4.5.8 Notation

No additional notational requirements.

10.4.6 OrganizationalRole

10.4.6.1 Extension

- Property (from UML2)

10.4.6.2 Generalizations

- BlockProperty (from SysML::Blocks)

10.4.6.3 Description

An OrganizationalRole indicates a role played by a Resource or OperationalNode in a specific context. That context is typically provided by another OperationalNode. This also allows for hierarchical decomposition of in particular OperationalNodes.

The same OperationalNode or Resource can play multiple roles within a single context, or may be used differently across multiple contexts.

In the case where the OrganizationalRole is typed by an OperationalNode it can also be connected to other such OrganizationalRoles using Needlines. Needlines attached to OrganizationalRoles are always contextual as well, i.e., they are only valid in the specific context where OrganizationalRoles are used.

An OrganizationalRole may be related to any number of Competencies through CompetenceRelationships.

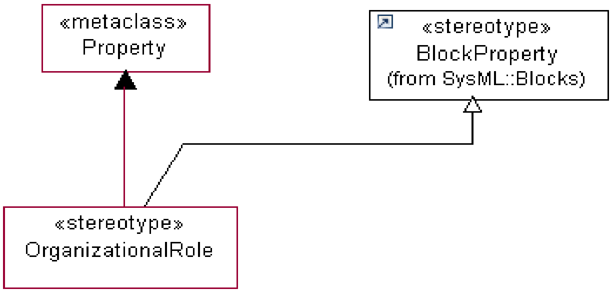


Figure 10.11 - OrganizationalRole

10.4.6.4 Attributes

No additional attributes.

10.4.6.5 Associations

No additional associations.

10.4.6.6 Constraints

[1] Asserts that an OrganizationalRole is typed by a OperationalNode, a Resource or a CapabilityConfiguration.

```
self.type->forAll(getAppliedStereotype('UPDM::OperationalNode')->notEmpty() or
getAppliedStereotype('UPDM::Resource')->notEmpty() or
getAppliedStereotype('UPDM::CapabilityConfiguration')->notEmpty())
```

10.4.6.7 Semantics

No additional semantic requirements.

10.4.6.8 Notation

No additional notational requirements.

10.5 SystemsView Package

10.5.1 Overview

The SystemsView Package contains UML stereotypes that assist the modeler in developing the views defined in the SystemsView viewpoint. These stereotypes support the description of systems and interconnections providing for, or supporting, DoD functions. The SystemsView Package also includes elements to identify the systems resources that support the operational activities and facilitate the exchange of information among operational nodes.

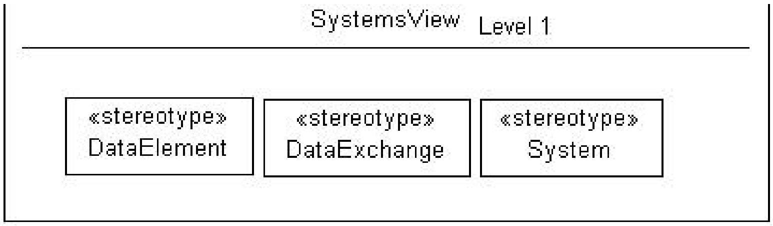


Figure 10.12 - SystemsView Package

10.5.2 DataElement

10.5.2.1 Extension

- Class (from UML2)

10.5.2.2 Generalizations

- Block (from SysML::Blocks)

10.5.2.3 Description

A data element exchanged in between systems.

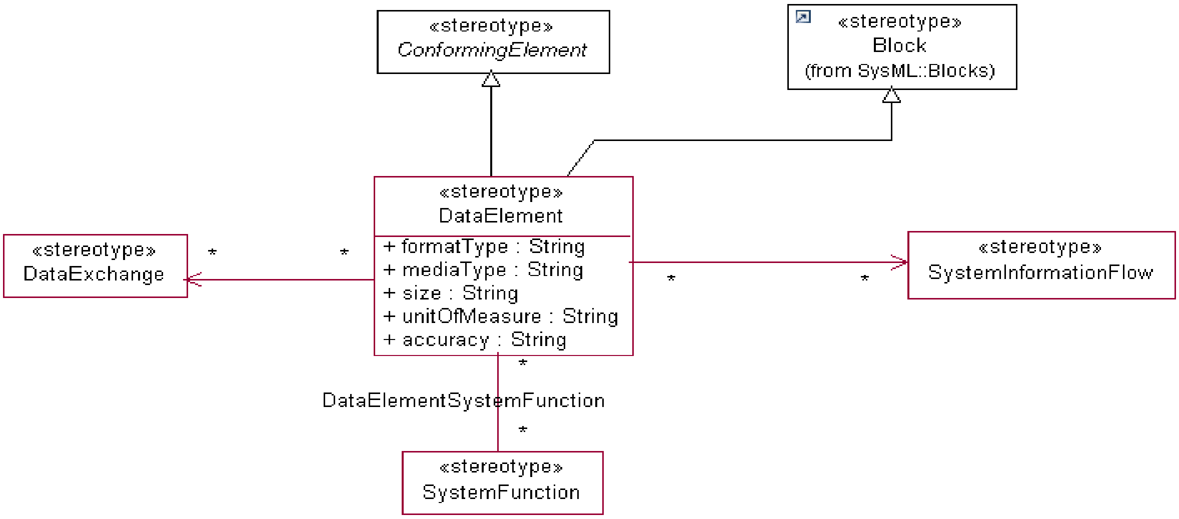


Figure 10.13 - DataElement

10.5.2.4 Attributes

- accuracy : String [1]
  Accuracy requirement for the data element

- formatType : String [1]
  The format for the data element.

- mediaType : String [1]
  The media for the data.

- size : String [1]
  Size of the data element.

- unitOfMeasure : String [1]
  Unit used to measure size.

10.5.2.5 Associations

- dataExchange : DataExchange [*]
  The DataExchanges that move these DataElements

- systemFunction : SystemFunction [*]
  The SystemFunctions that initiate the SystemInformationFlows along which the DataElement flows. See DataElementSystemFunction for more details.

- systemInformationFlow : SystemInformationFlow [*]
  The SystemInformationFlows along which the DataElement flows

10.5.2.6 Constraints

[1] Asserts that there are zero or more DataExchanges that utilize this DataElement

```
self.dataExchange->forAll(getAppliedStereotype('UPDM::DataExchange')->notEmpty())
```

[2] Asserts that a DataElementSystemFunction association exists between DataElement and SystemFunction.

```
self.oclAsType(uml::Class).getAllAttributes()->asOrderedSet()->select(association-
>notEmpty()).association->any

 (getAppliedStereotype('UPDM::DataElementSystemFunction')-> notEmpty())-
>notEmpty()
```

[3] Asserts that this DataElement is associated with zero or more SystemInformationFlows

```
self.systemInformationFlow.getAppliedStereotype('UPDM::SystemInformationFlow')-
>notEmpty()
```

10.5.2.7 Semantics

The DataElement indicates the type and structure of the information transferred between Systems represented by SystemInformationFlows.

10.5.2.8 Notation

No additional notational requirements.

10.5.3 DataExchange

10.5.3.1 Extension

InformationFlow (from UML2)

10.5.3.2 Generalizations

ItemFlow (from SysML::PortsAndFlows)

10.5.3.3 Description

An exchange that can carry information and data. It must be between Systems or SystemFunctionSpecifications.

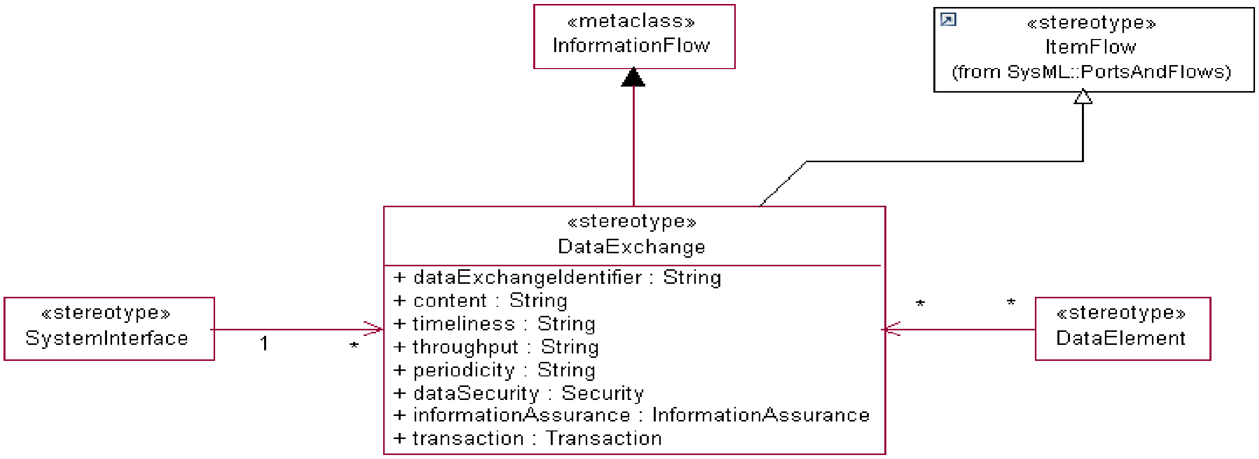


Figure 10.14 - DataExchange

10.5.3.4 Attributes

- content : String [1]
  The content of the flow.
- dataExchangeIdentifier : String [1]
  A user defined identifier - a primary key
- dataSecurity : Security [1]
  Security constraints that govern the DataExchange
- informationAssurance : InformationAssurance [1]
  InformationAssurance characteristics that govern the DataExchange
- periodicity : String [1]
  The update pace for the DataFlow.
- throughput : String [1]
  Amount of information that can travel on the flow.
- timeliness : String [1]
  Time since occurrence.
- transaction : Transaction [1]
  A Transaction that controls the DataExchange

10.5.3.5 Associations

No additional associations.

10.5.3.6 Constraints

[1] Asserts that the resulting value of the DataExchange is a DataElement

```
self.conveyed.oclAsType(uml::Classifier).getAppliedStereotype('UPDM::DataElement')
->notEmpty()
```

[2] Asserts that there is a SystemInterface along which the DataExchange occurs

```
self.realizingConnector->notEmpty() and

self.realizingConnector->forAll(getAppliedStereotype('UPDM::SystemInterface')-
>notEmpty() or

    type.getAppliedStereotype('UPDM::SystemInterface')->notEmpty()

)
```

[3] Asserts that there is InformationAssurance constraints on the exchange

```
self.informationAssurance->forAll(classifier->forAll(name='InformationAssurance'))
or

self.informationAssurance->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='InformationAssurance')))
```

[4] Asserts that there is a Security that governs the exchange

```
self.dataSecurity->forAll(classifier->forAll(name='Security')) or

self.dataSecurity->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Security')))
```

[5] Asserts that this exchange is under transaction control

```
self.transaction->forAll(classifier->forAll(name='Transaction')) or

self.transaction->forAll(classifier->forAll(allParents()->notEmpty() and
allParents()->exists(name='Transaction')))
```

[6] Asserts that this DataExchange occurs between two Systems

```
self.target.oclAsType(uml::Class).getAppliedStereotype('UPDM::System')->notEmpty()
and

self.source.oclAsType(uml::Class).getAppliedStereotype('UPDM::System')->notEmpty()
```

[7] Asserts that all the systemActivityFlows are of type SystemControlFlow or SystemInformationFlow and that they are owned by a SystemFunction

```
self.realizingActivityEdge->
forAll((getAppliedStereotype('UPDM::SystemControlFlow')->notEmpty() or
getAppliedStereotype('UPDM::SystemInformationFlow')->notEmpty() ) and
owner.getAppliedStereotype('UPDM::SystemFunction')->notEmpty()
)
```

[8] Asserts that the Interaction that contains an InformationExchange must be an OperationalEventTrace

```
self.realizingMessage.getAppliedStereotype('UPDM::TaskInvocation')->notEmpty()
```

10.5.3.7 Semantics

No additional semantic requirements.

10.5.3.8 Notation

No additional notational requirements.

10.5.4 System

10.5.4.1 Extension

10.5.4.2 Generalizations

- ConformingElement (from AllViews)
- Block (from SysML::Blocks)

10.5.4.3 Description

A System is any organized assembly of resources and procedures united and regulated by interaction or interdependence to accomplish a set of specific functions.

Systems consist of family of systems (FoS), System of systems (SoS), Networks of systems, individual Systems, and items (e.g., equipment SystemHardware and SystemSoftware).

A System provides the core construct of the SystemView. A System can be expressed in UML using a variety of constructs,, for example, a component or a class. Systems will have a number of operations and functions that they will deliver. The subset of information that relates to SystemFunctions will typically reflect only those functions of interest to link to OperationalActivities.

Two examples of approaches to system modeling indicate the need for extending such a basic UML construct as Class. A Systems View that is reviewing sets of components may model those systems as Components and align potential 'to be' architectures to the SystemFunction definitions. The component can have required and provided SystemFunctionSpecifications, modeled as UML interfaces.

In contrast, other approaches will need to model the detailed physical structure of the Systems and components, requiring detailed drill down into the composite structure of a System.

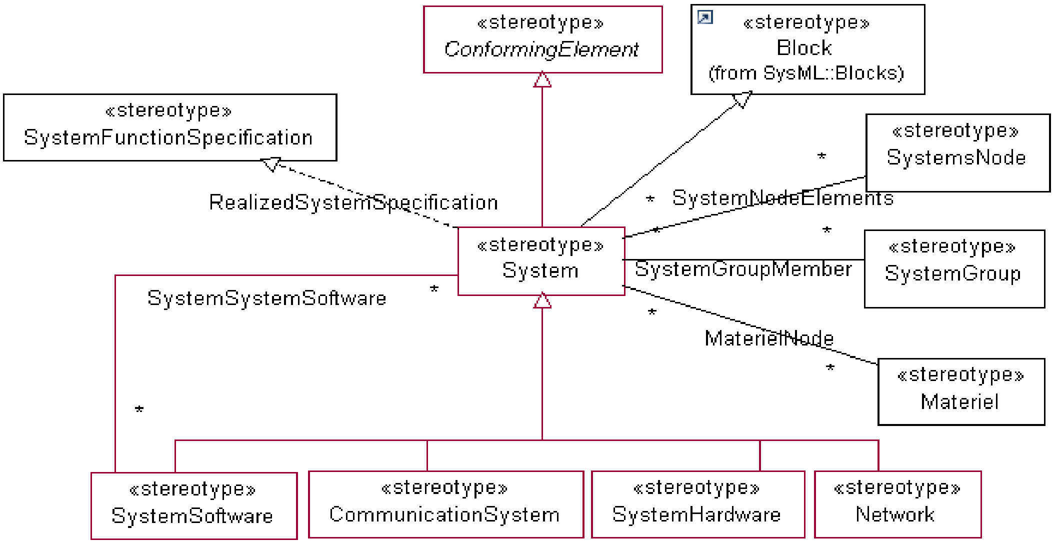


Figure 10.15 - System

10.5.4.4 Attributes

No additional attributes.

10.5.4.5 Associations

- from : CommunicationPath [*]
  The 'from' System in a CommunicationPath
- materiel : Materiel [*]
  Materiel needed by the System
- systemgroup : SystemGroup [*]
  The SystemsGroups to which this system belongs
- systemsnode : SystemsNode [*]
  The SystemsNode on which this system is deployed.
- systemSoftware : SystemSoftware [*]
  Software used by this System. See SystemSystemSoftware for more details.
- to : CommunicationPath [*]
  The 'to' System in a CommunicationPath

10.5.4.6 Constraints

[1] Asserts that if a Needline exists between two OperationalNodes, then there must exist a SystemsNode that hosts each of the OperationalNodes and a SystemInterface must exist between them.

```
self.getAllAttributes()->asOrderedSet()->

select(association->notEmpty()).association->any
```

```
 (getAppliedStereotype('UPDM::SystemInterface')-> notEmpty())->notEmpty()
self.getAllAttributes().association ->
any (getAppliedStereotype('UPDM::SystemSystemSoftware')->notEmpty())->notEmpty()
```

[2] Asserts that for all ports defined for a System, at least one of its stereotypes must be 'SystemPort'

```
self.oclAsType(uml::Class).ownedPort-> asBag()->
 forAll(p|p.getAppliedStereotype('UPDM::SystemPort')->notEmpty())
```

[3] Asserts that a System realizes interfaces that are SystemFunctionSpecifications, i.e., interfaces

```
self.clientDependency->forAll(spec|
if spec.oclIsKindOf(uml::InterfaceRealization)then
spec.getAppliedStereotype('UPDM::RealizedSystemSpecification')->notEmpty()
else
true
endif)
```

[4] Asserts that a System is part of at least one SystemGroup

```
self.getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any
 (getAppliedStereotype('UPDM::SystemGroupMember')-> notEmpty())->notEmpty()
```

[5] Asserts that there is an association between the System and a SystemsNode on which the Systems are deployed.

```
self.getAllAttributes()->asOrderedSet()->
select(association->notEmpty()).association->any
 (getAppliedStereotype('UPDM::SystemsNodeElements')-> notEmpty())->notEmpty()
```

[6] Asserts that there is at least oneSystemTask defined for the System

```
self.oclAsType(uml::Class).ownedOperation->asOrderedSet() ->
exists(getAppliedStereotype('UPDM::SystemTask')->notEmpty())
```

10.5.4.7 Semantics

No additional semantic requirements.

10.5.4.8 Notation

No additional notational requirements.

Part IV - Annexes

This part contains the following annexes:

A - DODAF and MODAF guidance (Non-normative)

B - Domain Metamodel (Non-normative)

C - Usage Example (Non-normative)

D - Bibliography

Annex A
DoDAF and MODAF Guidance

(non-normative)

This section provides guidance on the products that make up DoDAF and MODAF deliverables. The UPDM approach views these work products as queries against a model or models that produces the needed view. Under this approach, the team derives the work products from the model, the work products do not determine the model. Within the UML community, there are a variety of modeling approaches and styles each well suited to a particular set of Use Cases, including SysML, UML with action language, or UML following a Unified Process workflow. UPDM remains neutral in these implementation questions and has been constructed to work with current UML profiles as well as any of those produced in the future.

While such flexibility allows the framework to support a number of approaches, the actual model stereotypes require a rigorous implementation if the DoDAF deliverable is to enable the comparison and sharing of architectural information. Therefore, for each view this section lists the stereotype elements used to produce the deliverable. Depending on how the modeler decides to implement the stereotypes, the final deliverable could use a number of UML diagrams or the information might be exported from the model for manipulation in an external tool, such as a spreadsheet or project planning tool. This section supplements DoDAF and MODAF view descriptions with example UML guidance for each of these views. The section assumes familiarity with UML 2 mechanisms. If users wish to use a different UML style in their views, that will be UPDM compliant so long as they are able to express their concepts using the core stereotypes. The view description includes a table that reviews the typical use for each element in that view.

Note that the view architecture does not call out the “sub-views” such as OV-6a as being views at the same level of the overall view. Many DoDAF implementations start to derive views very much in the spirit of the root view but with special extensions for a particular client. Furthermore, the DoDAF 1.5 draft adds a number of suggested sub-views, especially for the SV-5. Rather than document these as an explosion of new views, the approach here recognizes the capability to visualize additional elements from the model in a new sub-view.

DoDAF has moved to a continuously updated deskbook of best practices and implementation guidance. That approach will provide the most up to date practices for implementation. Since UPDM has been designed to work with the evolution of UML languages, new advances in modeling DoDAF projects should continue to emerge with the improvement in tools, techniques, and supporting hardware. The examples here, then, illustrate the core mechanisms of UPDM and do not reflect the full richness of the possible solutions made available for DoDAF by using UPDM to deliver DoDAF artifaces using UML.

A.1 All Views

A.1.1 Overview and Summary Information (AV-1)

A.1.1.1 Product Definition

The Overview and Summary Information provides 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. [1]

A.1.1.2 UPDM Notation

UPDM will identify anything inside a package stereotyped <<AV>> with type==AV1 as the AV-1. Typically, this will be a URI link to an external resource that provides the information required.

AV-1 contains sufficient textual information to enable a reader to select, for a particular need, one architecture from among many to read in more detail. AV-1 serves two additional purposes: in the initial phases of architecture development, it serves as a planning guide; upon completion of an architectural description, AV-1 provides summary textual information concerning the architecture.

A.1.1.3 Product Detailed Description

The AV-1 product comprises a textual executive summary of a given architecture and provides key documentation. Typical information about the architectural project includes the project name, the architect, and the organization developing the architecture. The summary information can include assumptions and high level constraints. Some projects will require the identification of the approving authority and the completion date, including the level of effort and costs (projected and actual) required to develop the architecture. The scope identifies the views and products that have been developed and the temporal nature of the architecture, such as the time frame covered, whether by specific years or by designations such as current, target, or transitional. Scope also identifies the organizations that fall within the scope of the architecture. As the architectural descriptions increasingly span a number of different architectural projects, with multiple models contributing to an architectural deliverable, the architectural metadata may change. The semantics section includes the profile defined elements useful for integrated and merged document. The AV-1 will also include additional elements. Optional element might include the following.

Purpose and Viewpoint explains the need for the architecture, what it should demonstrate, the types of analyses (e.g., Activity-Based Costing) that will be applied to it, who is expected to perform the analyses, what decisions are expected to be made on the basis of analysis, who is expected to make those decisions, and what actions are expected to result.

Context describes the setting in which the architecture exists. It includes such things as mission, doctrine, relevant goals and vision statements, concepts of operation, scenarios, information assurance context (e.g., types of system data to be protected, such as classified or sensitive but unclassified, and expected information threat environment), environmental conditions, and geographical areas addressed, where applicable. Context also identifies authoritative sources for the rules, criteria, and conventions that were followed[1]. The context also identifies tasking for the architecture project and known or anticipated linkages to other architectures.

Tools and File Formats Used identifies the tool suite used to develop the architecture and file names and formats for the architecture and each product.

Findings states the findings and recommendations that have been developed based on the architecture effort. Examples of findings include identification of shortfalls, recommended system implementations, and opportunities for technology insertion.

[1] All product definitions are taken from the DoDAF Volume II, Version 1 specification or the MODAF Viewpoints Overview, Version 1.0 documents. These are cited in the Bibliography.

[1] See Universal Reference Resources [URR] section in the Deskbook for examples of authoritative sources.

A.1.1.4 Semantics

The purpose of the AV-1 is a clear and concise communication of the scope of the subject system within the context of the enterprise. The following stereotype elements will typically appear in the AV-1. These elements might be handled in any number of ways depending on the nature of the integrated model.

Table A.1 - AV-1 Elements

UPDM Element	Significance to the View
ArchitectureDescription	A collection of architecture views that define the structure and purpose for the project using the architecture. Includes the definition of the Enterprise as well as the relevant view product descriptions.
ArchitectureView	An abstract concept that indicates a portion of the architecture viewed from a particular perspective. Architecture views could include the operational, system, and technical standards, for example.
Concern	The interest in the architecture held by a stakeholder.
Doctrine	Formal statement of the general concepts used by an organization to perform its mission.
Enterprise	The high level unit of analysis for the architecture description.
Goal	A specific, required objective of the enterprise that the architecture represents.
Policy	An established course of action that must be followed. Policies can be the source of rules that show up in the operational activities.
Stakeholder	Key actors with an interest in the enterprise, including actors outside the enterprise governance structure. Some typical stakeholders include: Client; Acquirer; Owner; User; Operator; Architect; System Engineer; Developer; Designer; Builder; Maintainer; Service Provider; Vendor; Subcontractor; Planner.
StakeholderConcern	A relationship that connects a stakeholder to a concern.
StrategicMission	A StrategicMission summarize goals and objectives into a mission statement. A Mission statement defines the broader goals. The mission can remain the same for decades if crafted well.
UPDMAttributes	Establishes a mechanism to associate any element in the model with any number of references that may be external to the model.
Viewpoint	A Viewpoint is a set of conventions for constructing, interpreting and analyzing a view.
Vision	The high level description of the aims of the enterprise. Compared to mission, vision is more specific in terms of objective and future state. Vision is related to some form of achievement if successful.

A.1.1.5 Relationship to other products

The elements in the all view related to all of the products and elements throughout the model.

A.1.2 Integrated Dictionary (AV-2)

A.1.2.1 Product Definition

The Integrated Dictionary contains definitions of terms used in the architecture description as well as the set of abstract base classes and model library used throughout the UPDM. The AV-2 will also include descriptions of any custom views developed for the architectural description. The AV-2 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 that are associated with the developed architecture products.

AV-2 provides a central repository for a given architecture’s data and metadata. AV-2 enables the set of architecture products to stand alone, allowing them to be read and understood with minimal reference to outside resources. AV-2 provides unambiguous definitions and can help show compliance in the architecture with external reference models that define the structure of information and data.

A.1.2.2 UPDM Notation

The AV-2 should provide a glossary easy to access for users of the architecture. With the proliferation of architectural descriptions and the use of common taxonomies across organizations, the AV-2 increasingly needs to handle external taxonomy references. Every UPDM element can indicate it references an external taxonomy or ontology that provides the meaning for that term.

The AV-2 can also be represented as a matrix generated from DoDAF model elements for names, URI references, and any associated documentation.

A.1.2.3 Product Detailed Description

AV-2 defines terms used in an architecture, but it is more than a simple glossary. Many architectural products have implicit or explicit information in the form of a glossary, a repository of architecture data, their taxonomies, and their metadata. Each labeled item (e.g., icon, box, or connecting line) in the graphical representation has a corresponding entry in AV-2. Each item from a textual representation of an architectural product also has a corresponding entry in AV-2. The type of metadata included in AV-2 for each item depends on the type of architectural product from which the item is taken. For example, the metadata for an OperationalNode in AV-2 includes the attributes Name, Description, and Level Identifier. A taxonomy of OperationalNodes applicable to the architecture may be consulted, and the name used for a specific OperationalNode may be chosen from that taxonomy. The AV-2 entry for the node then consists of the metadata data fields (a name field, a description field, and a level identifier field), a value for each of these fields, and the taxonomy for OperationalNodes.

The AV-2 also includes artifacts used for managing model assets, such as the UPDM Model Library. The Model Library provides a set of classes for use through the architectural descriptions when the model calls for an instance of a particular class. The Model Library includes such elements as “Project” and “Milestone.”

The AV-2 also includes the abstract elements used across the UPDM profile. By modeling constructs that appear in multiple views as abstract elements, UPDM provides an extension point for customization. In addition, for those linking UML to some other modeling structure, such as BPMN, a single set of mappings to the Abstract element will work throughout the model.

A.1.2.4 Semantics

The purpose of the AV-2 is to provide a reference to define and to explain all the constructs used in the architecture. UPDM has stereotypes for each of the defined view deliverables. These packages have been summarized here by just indicating the number sign next to the view name.

Table A.2 - AV-2 Elements

UPDM Element	Significance to the View
AcquisitionView-#	Package that includes artifacts for a MODAF Acquisition View.
AllView-#	Package that includes artifacts for one of an AllView.
ArchitectureDescription	The complete set of metadata associated with the architecture is usually attached to the ArchitectureDescription.
ArchitectureView	An abstract concept that indicates a portion of the architecture viewed from a particular perspective. Architecture views could include the operational, system, and technical standards, for example.
CustomView	Package that includes artifacts for a CustomView. A CustomView would be a new set of artifacts defined as extensions to DoDAF to elaborate an important part of the architecture for the enterprise. Examples of custom views include a Logistics View, a Security View, or a Service Oriented View. The CustomView will have a set of view types that would define the precise nature of each artifact needed in the new view.
ExternalReference	A triple that names, identifies and types a reference that can be external to the model or some structure created in the model for reference purposes.
OperationalView-#	Package that includes artifacts for the OperationalView.
Policy	An established course of action that must be followed. Policies can be the source of rules that show up in the operational activities.
Requirement	A requirement can apply to any element in the model. A requirement can also use ExternalReference to link to a requirements management system. SysML users can use this with the SysML Requirement stereotype.
StrategicView-#	Package that includes artifacts for a Strategic View.
SystemView-#	Package that includes artifacts for a System View.
TechnicalStandardsView-#	Package that includes artifacts for a TechnicalStandardsView.
UPDMAttributes	Every element in a model that applies the UPDM profile will have the UPDMAttributes, even if that element does not apply a specific stereotype. This enables the common UML elements in the model to take advantage of the features of UPDM that will link the elements into other models. UPDMAttributes has an array of ExternalReferences to use as needed to integrate the model and support information import and export
Viewpoint	A Viewpoint is a set of conventions for constructing, interpreting and analyzing a view.

Table A.2 - AV-2 Elements

ConformingElement	Base element for any UPDM element that can apply specific technical standards or performance parameters.
Specification	InstanceSpecification used as the root for all UPDM InstanceSpecifications. The element used for the instance resides in the Model Library.

A.1.2.5 Relationship to other products

The elements in the all view related to all of the products and elements throughout the model.

A.2 Operational Views

The Operational View focuses on the processes and information needed to achieve mission success. The view includes the following deliverables.

A.2.1 High-Level Operational Concept Graphic (OV-1)

A.2.1.1 Product Definition

The High-Level Operational Concept Graphic describes a mission and highlights main OperationalNodes (see OV-2 definition) and 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.

A.2.1.2 UPDM Notation

A reference to an external graphic located in package with the stereotype <<OperationalView-1>>. The OV-1 can also provide a Use Case diagram to show the context providing traceability to the collaborations in the other Operational and System Views using Use Case realizations (optional).

A.2.1.3 Product Detailed Description

In some contexts, it might make sense to have the graphic link to elements in the architectural descriptions or to other elements in the model, such as the Nodes or Organizations.

Organizations, organization types, and/or humans depicted in OV -1 could be associated with OperationalNodes and optionally Operational Roles. If the architecture uses a context diagram, OperationalNodes in OV-1 context diagrams should be traceable to OperationalNodes in OV-2; relationships in OV-1 will likely trace to Needlines in OV-2.

OV-1 consists of a graphical executive summary for a given architecture with accompanying text. The product identifies the mission/domain covered in the architecture. OV-1 should convey, in simple terms, what the architecture is about and an idea of the players and operations involved. Therefore, the OV-1 has a minimum number of constraints.

A.2.1.4 Semantics

The purpose of OV-1 is to provide a quick, high- level description of what the architecture is supposed to do, and how it is supposed to do it. This product can be used to orient and to focus detailed discussions. The diagrams should encourage human communication and portray system goals to high-level decision makers.

With the goal of clear and concise communication of the scope of the subject system within the context of the enterprise, the graphical depiction of the OV-1 is typically an artist-rendered product reflecting content derived from multiple sources. The primary information sources for the OV-1 should include the AV-1 Overview and Summary document along with any general overviews of the Enterprise, such as an Operational Context Diagram or an Enterprise Use Case Diagram.

Table A.3 - OV-1 Elements

UPDM Element	Significance to the View
Asset	A generic and very high-level description of an entity that is elaborated in the architecture.
AssetMapping	A relationship that indicates how an asset has been realized in the actual operation or system deployment.
Concern	The concept graphic should clearly indicate the concerns for the enterprise.
Enterprise	Package that includes artifacts for the AllView and any elements owned by the Enterprise.
Goal	A specific, required objective of the Enterprise that the architecture represents.
Policy	An established course of action that must be followed. Policies can be the source of rules that show up in the operational activities.
Stakeholder	Key actors with an interest in the enterprise, including actors outside the Enterprise governance structure. The OV-1 can provide a graphic representation of the most important stakeholders.
StakeholderConcern	A relationship that connects a stakeholder to a concern.
StrategicMission	The concept graphic should communicate the key strategic missions involved in the architecture.
Vision	The high level description of the aims of the enterprise. Compared to mission, vision is more specific in terms of objective and future state. Vision is related to some form of achievement if successful.

A.2.2 Operational Node Connectivity Description (OV-2)

A.2.2.1 Product Definition

The Operational Node Connectivity Description graphically depicts the information connectivity needs between Nodes. The relationship between the Nodes is represented by Needlines that indicate the existence of information exchanges. The product can include internal Nodes (internal to the architecture) as well as external Nodes. Operational Nodes can have a number of types, including organizations, communities of interest, or virtual working groups, such as a task force or tiger team. A Node can also contain OperationalActivities that are elaborated in the OV-5.

A.2.2.2 UPDM Notation

OV2 can be any diagram that can use Nodes and Needlines. In some cases, a modeler might want to stick to a class diagram to show the Needlines between Nodes or they may want to show more details of the model and use a SysML Block Diagram or UML 2 composite structure diagrams. The example below shows the most basic OV-2, with three different operational nodes, one that represents a community of interest tracking medical best practices, a group that is supporting recovery efforts in Pakistan and a service that broadcasts information on South Asia and it trying to provide information to relatives. Each Operational Node has an <<OperationalTask>> that links to the activities in the OV-5 which is then linked to the SystemView using the OperationalActivityRealization. In addition, a set of tasks such as these can rely on existing capabilities and work processes. The <<OperationalCapability>> provides the general usage processes. In order to have a flexible force the responds rapidly such an approach focused on the effects of activities makes sense. See the section in the OV-5 for suggested ways to elaborate effects based planning. These operational nodes could have more than one OperationalTask, as shown. Furthermore, if required, the architect can produce an OV-2 that decomposed the node into more detailed elements, showing the information needs between nested nodes. Such a composite structure allows decomposition.

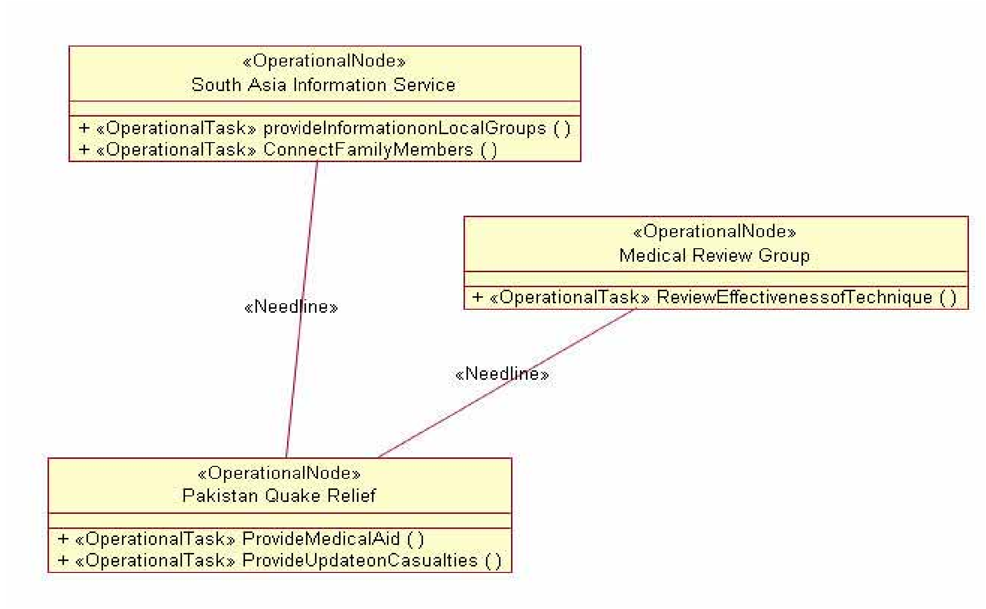

«OperationalNode»
South Asia Information Service
+ «OperationalTask» provideInformationonLocalGroups ( )
+ «OperationalTask» ConnectFamilyMembers ( )
«OperationalNode»
Medical Review Group
+ «OperationalTask» ReviewEffectivenessofTechnique ( )
«Needline»
«Needline»
«OperationalNode»
Pakistan Quake Relief
+ «OperationalTask» ProvideMedicalAid ( )
+ «OperationalTask» ProvideUpdateonCasualties ( )

Figure A.1 - OV2 Example

A.2.2.3 Product Detailed Description

The main features of this product are the Operational Nodes and the Needlines between them that indicate a need to exchange information. The product indicates the key players and the interactions necessary to conduct the corresponding Operational Activities of OV-5.

An Operational Node is an element of the operational architecture that produces, consumes, or processes information. What constitutes an Operational Node can vary among architectures. UPDM will easily support Nodes that represent organizations, Communities of Interest, or personnel. If an architecture has a number of Nodes all engaged in the same OperationalActivity, consider constructing a Community of Interest for that OperationalActivity as this will make it easier to map that activity into the rest of the models. Nodes will typically represent virtual or joint structures that don’t have to map to the organizational structure. However, given than an organization can function as an OperationalNode, UPDM uses the same concept for both to avoid confusion in the comparison of models.

A Needline may represent many information exchanges or dependencies. Accordingly, once a Needline has been identified between any two operational nodes, no other Needline is appropriate. Use only a single Needline to represent the interactions of all Information Exchanges that have a common source and destination pair of Nodes. A modeler could use two Needlines between nodes to represent the different source and destination, if required by the project.

The OV-2 diagram can include substantial additional detail depending on the needs of the project. With UML's capabilities for integrated modeling with activities and context specific composite structures, the diagram elements in the OV-2 can provide insight into the organizational resources as well as the planning and training needs for the operational nodes. UPDM includes stereotypes to identify such information, but these are not required for an OV-2. In a typical model, a user will select the elements for the diagram that best communicate the point to illustrate while maintaining the complete model in the background. These elements also provide the substantial detail needed to run simulations against the operational model.

A.2.2.4 Semantics

OV-2 displays the need to exchange information between Nodes in order to achieve certain operational activities. The OV-2 represents an essential diagram for any UPDM model. OV-2 describes the connectivity requirements between the Operational Nodes as well as providing an overview of the major activities and tasks that take place at the Nodes. For Net-centric operations, the OV-2 can show providing and consuming nodes.

Table A.4 - OV-2 Elements

UPDM Element	Significance to the View
Competence	Competence in a specific discipline that can be provided by the indicated Resources. Gaps in competence to provide an OperationalCapability will generate training requirements for those involved in the mission. OperationalCapabilityRoles require Competencies to meet their declared responsibilities and perform their OperationalTasks.
CompetenceKind	An enumeration that indicates whether the competence is something desired, provided, or required. The user can also have a custom type of competence relationship if needed.
CompetenceRelationship	A Resource or OrganizationalRole can have a set of related Competencies that are required or provided to meet their declared responsibilities and perform their OperationalTasks.
InformationExchange	The existence of a Needline implies at least one InformationExchange between the Nodes. If desired, an OV-2 graphic could list the InformationExchanges traveling along a Needline.
LineKind	Indicates how one OrganizationalResource relates to another resource.
Needline	Indicates that one Node “needs” information from another Node.
NodeType	The Node can be a CommunityofInterest, Personnel, Organization, OrganizationType, or Custom. A Custom NodeType should include a description of that NodeType somewhere in the model.

Table A.4 - OV-2 Elements

OperationalNode	An OperationalNode performs a role or mission; a conceptual architectural element that produces, consumes or processes information. An OperationalNode exists to realize one or more capabilities. Deployment of the capabilities in systems that help to realize those capabilities is allocated to SystemsNodes.
OperationalNodePort	A UML port on an OperationalNode. This is used for details of the Needline in the context of a composite structure.
OperationalNodeSpecification	An abstract specification of the tasks that can be implemented or carried out at any number of operational nodes.
OperationalRole	An OperationalRole associates a Node to the OrganizationalResource that actually perform the operations of the Node. The responsibility played by a node when interacting with other elements. A role could include a service provider or service consumer role.
OperationalServiceConsumer	A node or any other classifier used in the OV-2 can act as a service consumer. Applying this stereotype indicates that these nodes will work with net-centric services.
OperationalServiceProvider	A node or any other classifier used in the OV-2 can act as a service provider. Applying this stereotype indicates that these nodes will work with net-centric services.
OperationalTask	A stereotype of an Operation on a Node that indicates the OperationalActivity that is elaborated in the OV-5 and OV-6. The task provides the means to invoke the operational activities.
RealizedOperationalSpecification	A realization that indicates that an OperationalNode is satisfying the specifications for OperationalTasks in an OperationalNodeSpecification.
ResourceCompetence	Indicates that the resource on one end of the association is meant to satisfy the competence on the other end of the association.

A.2.2.5 Relationship to other products

Table A.5 - Relationship of OV2 to Other Products

View	Relationship Details
OV-1	Organizations, organization types, and/or humans depicted in OV-1 should be associated with operational Nodes and optionally OperationalRoles. Operational Nodes in OV-1 context diagrams should be traceable to operational Nodes in OV-2; relationships in OV-1 should trace to Needlines in OV-2.
OV-3	A Needline in OV-2 maps to one or more information exchanges in OV -3. The set of InformationExchanges (stereotyped Message) from one lifeline (operational Node) to another in OV-6c maps to a Needline connecting those same two operational Nodes in OV-2.
OV-4	Organizations, organization types, and/or human roles in an OV-4 may map to one or more operational Nodes in an OV-2, indicating that the node represents the organization.
OV-5	The activities annotating an operational Node in an OV-2 map to the activities described in an OV -5.
OV-6	Lifelines in OV-6c map to operational Nodes in OV-2.

Table A.5 - Relationship of OV2 to Other Products

SV-4	The OperationalActivityRealization provides a container for the SystemActivities that support these Nodes.
SV-10c	The OperationalActivityRealization provides a container for the SystemEventTraces that support these Nodes.

A.2.3 Operational Information Exchange Matrix (OV-3)

A.2.3.1 Product Definition

The Operational Information Exchange Matrix details information exchanges and identifies “who exchanges what information, with whom, why the information is necessary, and how the information exchange must occur.” The OV-3 is a matrix of Information Exchange Requirements (IERs) that collectively represent the Needlines of the OV-2. Each row of the matrix represents an IER, which is comprised of data characteristics of that data transferred between roles/objects in any of the dynamic models. A distinct IER is identified for each pair of objects or roles that interact and exchange information. Specific characteristics of the IER are typically associated with non-functional requirements or design constraints. Accordingly, each of the Information Elements in the matrix should trace to the Logical Data Model, OV-7.

The emphasis of the OV-3 is on the logical and operational characteristics of the information exchanged. The product is not intended to provide exhaustive capture of all details of information exchanged within the architecture, but as a mechanism to understand the most important aspects of significant exchanges. Certain aspects of the information exchange can be crucial to the operational mission and should be tracked as attributes in OV-3. For example, if the subject architecture concerns tactical battlefield targeting, then the timeliness of the enemy target information is a significant attribute of the information exchange.

A.2.3.2 UPDM Notation

UPDM offers elements with the required attributes to enable producing the OV-3 matrix. The OV-3 uses the model library for defining common security and information assurance attributes. Recognize that UPDM is not the place to define the common models for IA or Security: an individual model will implement the security features of a particular enterprise, extending the elements in the model library. The information needed for OV-3 can be exported to a spreadsheet from the information in the model.

Product Detailed Description

OV-3 identifies Information Elements and relevant attributes of the information exchange and associates the exchange to the producing and consuming Operational Nodes and activities and to the Needline that the exchange satisfies. Information exchange is an act of exchanging information between two distinct Operational Nodes and the characteristics of the act, including the Information Element that needs to be exchanged and the attributes associated with the Information Element (e.g., Scope), as well as attributes associated with the exchange (e.g., Transaction Type). A Needline represents one or more information exchanges.

UPDM insists that modelers apply InformationAssurance and Security aspects to any Information Exchange in order to produce a proper OV-3. However, UPDM does not define the key features of Information Assurance and Security elements in anticipation that a variety of government agencies will maintain these definitions. These attributes can become both quite complex and also very important for the success of a model.

A.2.3.3 Semantics

The main element in the OV-3 is the information exchange. The security and information assurance requirements related to that exchange are defined as DataTypes in the model library.

Table A.6 - OV3 Elements

UPDM Element	Significance to the View
InformationElement	A formalized representation of information used in an operational process. This is the type of the information used on the exchange.
InformationExchange	The main unit of analysis for the OV-3, the exchange takes place between two Operational Nodes. This matrix elaborates the definition of the exchange. If desired, an OV-2 graphic could list the InformationExchanges traveling along a Needline. An InformationExchange must have security and information assurance attributes.
Needline	An information exchange will always have an associated Needline to reference in the matrix.
NodeType	Indicates the type of node for the information exchange.
OperationalNode	The elements that exchange information elements using information exchanges.
Requirement	A holder for additional requirements on the element or the exchange. Typically this will link to an external requirements management system.
Security	Details on the security attributes associated with the exchange.

A.2.3.4 Relationship to other products

Table A.7 - Relationship of OV-3 to Other Products

View	Relationship Details
OV-2	A Needline in OV-2 maps to one or more InformationExchanges in OV-3.
OV-5	An InformationExchange in OV-3 can map to one or more information flows (an external input, an external output, or an output from one OperationalActivity mapped to an input to another) in OV-5, The Object Flow in the OV-5 can also represent an InformationExchange.
OV-6b	Events in OV-6b map to triggering events in OV-3.
OV-6c	The set of InformationExchanges (stereotype Message) from one (operational Node) to another in OV-6c maps to a Needline in OV-2.
OV-7	An InformationElement in OV-3 should be constructed of entities described in OV-7.
SV-6	If any part of an InformationElement in OV-3 originates from, or flows to an OperationalActivity that is to be automated, then that InformationElement should map to one or more system data elements in SV-6.

A.2.4 Organizational Relationships Chart (OV-4)

A.2.4.1 Product Definition

The Organizational Relationships Chart 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. This product clarifies the various relationships that can exist between organizations and sub-organizations within the architecture and between internal and external organizations.

A.2.4.2 UPDM Notation

This diagram represents information generally developed and maintained using techniques and tools better suited to the task than UML. Usually organizations other than system engineering teams maintain this information using tools such as LDAP. Therefore, this diagram will typically rely on the UPDM features to allow external references.

Organizations in UPDM represent on type of OperationalNode. This view can be easily supplemented by additional views to show issues related to training and information requirements for organizations.

That said, the OV-4 can add some relationships to the model useful for elaborating the organizational interactions. In cases where the Nodes on the operational side represent communities of interest, the Organizational Relationships Chart becomes the only view that provides insight into organizational boundaries.

The UML elements for OV-4 may be represented as a Composite Structure diagram where the Classes represent the organizations and the internal parts play roles that indicate the organizational relationships.

A.2.4.3 Product Detailed Description

OV-4 illustrates the relationships among organizations or resources in an architecture. These relationships can include supervisory reporting, command and control relationships, and command-subordinate relationships. Another type of relationship is a coordination relationship between equals, where two organizations coordinate or collaborate without one having a supervisory or command relationship over the other. Others may be defined depending on the purpose of the architecture. Architects should feel free to supplement this diagram with additional relationships necessary to support the goals of the architecture.

Table A.8 - OV-4 Elements

UPDM Element	Significance to the View
Competence	Also known as “training.” Any role carried out by a resource requires a “competence.”
LineKind	Indicates how one OrganizationalResource relates to another.
Location	Represents the physical location of an organization. Often represented on the SV-1. In some cases, the location of the organization represents a significant attributed for the organizational relationships.
Materiel	The physical resources used to accomplish a particular mission. For the OV-4, these will be OrganizationalResources that play a role that provides a physical asset.
NodeType	The type of the operational Node. The Node can be a CommunityofInterest, Personnel, Organization, OrganizationType, or Custom. The OV-4 will typically not elaborate a Community o interest.

Table A.8 - OV-4 Elements

OperationalNode	When the NodeType is an organization, the operational Node represents an organization. The Node can have a relationship to an OrganizationalResource that can either be illustrated on this diagram or included in a supporting diagram not displayed to the DoDAF user.
OrganizationalRelationship	A relationship between two OrganizationalResources.
OrganizationalResource	A resource that plays a role in a Node where NodeType will usually equal Organization. This can be eleaborated in a composite structure diagram.
Resource	The generalized classifier for any resource in the model, typically specialized in this diagram to the OrganizationalResource. The OV-4 could be used to elaborate any relationships between an organization and resources.

A.2.4.4 Relationship to other products

Table A.9 - OV-4 Relationship to other products

View	Relationship Details
OV-2	The operational Nodes from the OV-2 can be elaborated and decomposed in OV-4.

A.2.5 Operational Activity Model (OV-5)

A.2.5.1 Product Definition

The Operational Activity Model describes the operations normally conducted in the course of achieving a mission or a business goal. It describes Capabilities, OperationalActivities (or tasks), input and output (I/O) flows between Activities, and flows to and from activities that are outside the formal boundaries of the Enterprise.

OV-5 can place these activities in the context of capabilities. The activities provide detailed that helps to relate those capabilities to mission accomplishment. The DoD Dictionary of Military Terms [DoD JP 1-02, 2001] defines a capability as “the ability to execute a specified course of action. (A capability may or may not be accompanied by an intention.)” A capability can be defined by one or more sequences of activities, referred to as a scenario. A capability may use the OV-6 views for elaboration in terms of the attributes required to accomplish the set of activities (such as the sequence and timing of OperationalActivities or materiel that enable the capability), in order to achieve a given mission objective. Capability-related attributes may be associated with specific activities or with the information flow between activities, or both. When represented by a set of OperationalActivities, a capability can also be linked to an Operational Node.

A.2.5.2 UPDM Notation

The OV-5 represents the core business processes for the Enterprise. The architect has a variety of options in the OV-5 for organizing the scenarios and operational activities in the model. A Use Case diagram can provide the high level organization needed to describe the value propositions for the users. In UPDM the high level elements that function as Use Cases take the stereotype <<OperationalCapability>>. An architect can also use the <<OperationalCapabilityRealization>> to provide for the containers for the activities that come together to deliver that operational capability. The diagram below shows Use Cases with actors who are able to provide useful capabilities.

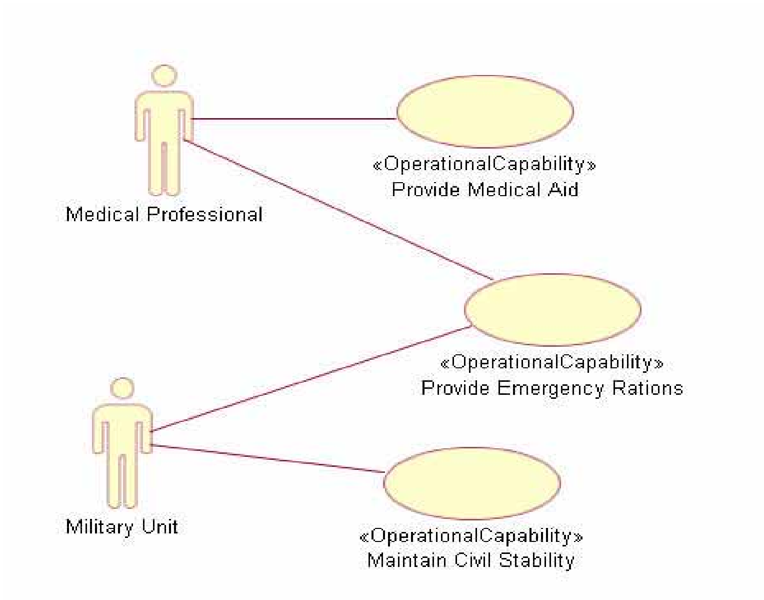


Figure A.2 - OV-5 Use Case Example

These use cases are expressed with a class diagram show the realizations of the Use Case.

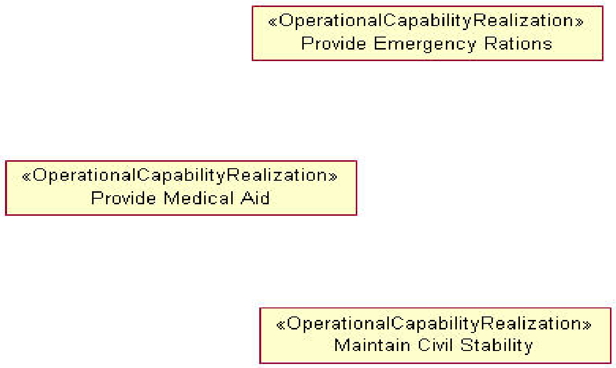

«OperationalCapabilityRealization»
Provide Emergency Rations
«OperationalCapabilityRealization»
Provide Medical Aid
«OperationalCapabilityRealization»
Maintain Civil Stability

Figure A.3 - OV-5 Realization Classes

In the OV-5, the model will have one or more activity diagrams for each OperationalCapability. These activity diagrams relate to the specific operational nodes through OperationalActivities in the activity diagram. In addition, effects based planning provides a robust mechanism for linking the OperationalNodes, OperationalTasks, and OperationalActivities. Note how the realization can include an operational task with the composite structure compartment showing the links to the nodes.

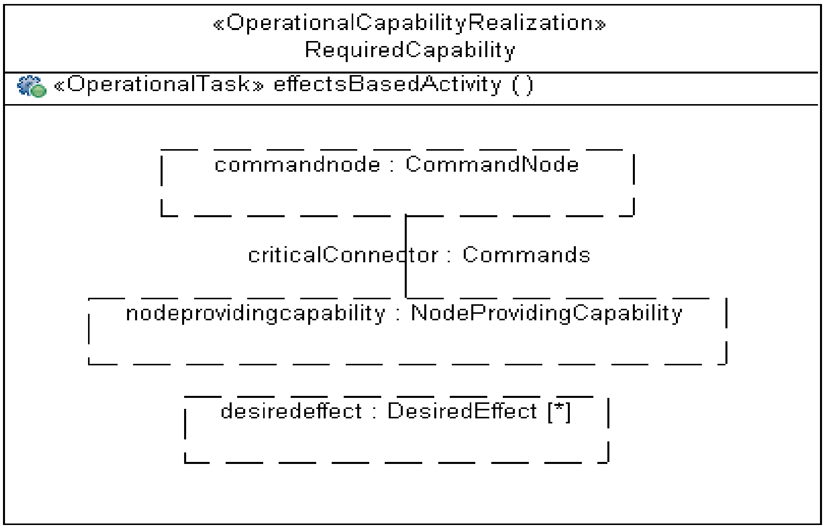

«OperationalCapabilityRealization»
RequiredCapability
«OperationalTask» effectsBasedActivity ( )
commandnode : CommandNode
criticalConnector : Commands
nodeprovidingcapability : NodeProvidingCapability
desiredeffect : DesiredEffect [*]

Figure A.4 - Realization Task with Composite Structure

The example below shows the elaboration of one activity. Note that the activities that need to be stereotyped as UPDM <<OperationalActivity>> depend on the level of detail needed in the architecture. That need is driven especially by the mapping needed in the SV-5 between SystemFunction and OperationalActivity.

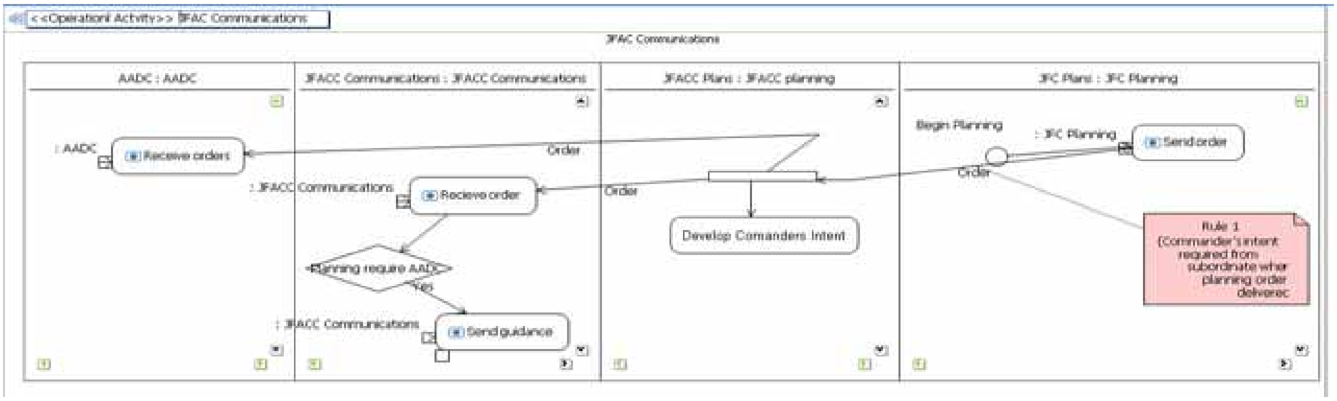


Figure A.5 - OV-5 Elaborated Activity

There are two distinct sets of ActivityDiagrams, one to show the decomposition of the Activities independent of where they are performed and one to show the performance on a real mission. By factoring and displaying these two concerns, the Activity diagrams are easier to understand and to model. In the example above, Develop Commanders Intent provides an example of a Called Behavior which references another Operational Activity. The stereotype for UPDM is on the activity, not the called behavior. Note the activity in this example includes “CalledOperations.” The flexible and annotative nature of UPDM allows such modeling, although those operations are not UPDM stereotypes for OperationalActivities.

In order to link an OperationalTask from UPDM into activities, UPDM uses effects based planning in order to maintain a precise mapping. For activities, the diagram below shows the general approach to implementing Effects Based Planning.

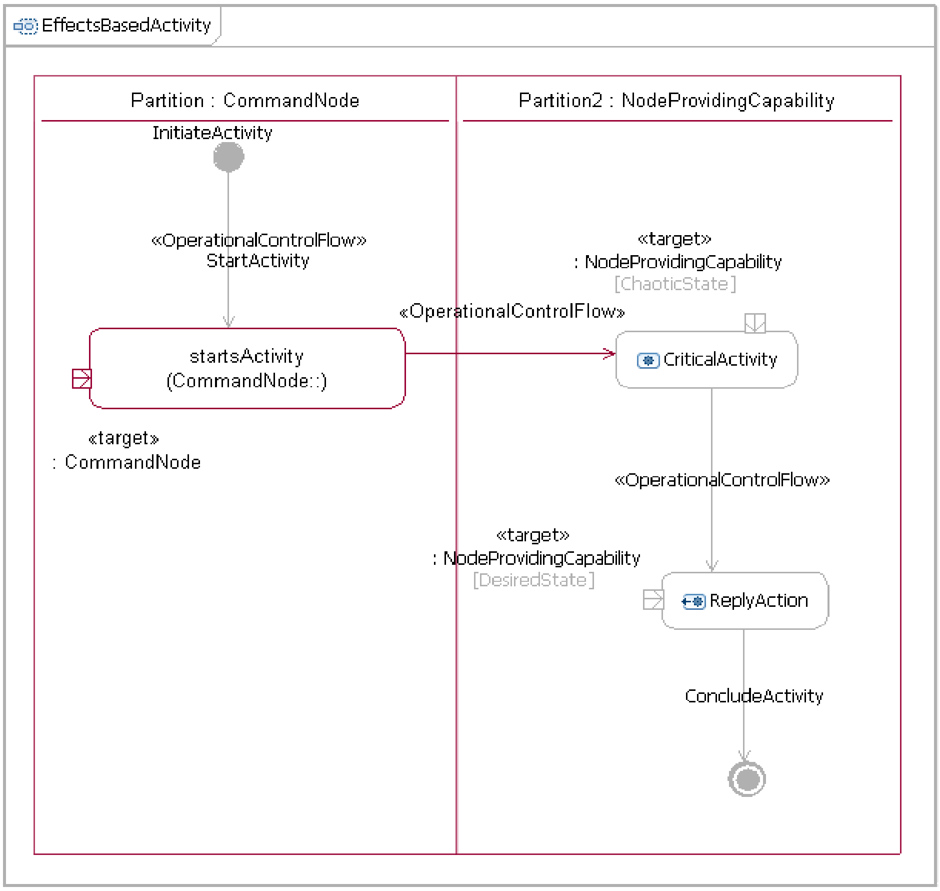


Figure A.6 - Effects Based planning Activity Diagram

A basic thread through this approach is as follows. The OperationalCapabilityRealization, **RequiredCapability**, is realized with the OperationalActivity, **EffectsBasedActivity** whose initial action is a **StartActivity** OperationalControlFlow. **The OperationalNode, CommandNode,** initiates the action that calls its OperationalTask, startsActivity(), through the **OperationalControlFlow**. The OperationalControlFlow passes to **NodeProvidingCapability**, where it performs the CriticalActivity by invoking its OperationalTask, **criticalActivity**() through a call operation action. The **DesiredEffect** is specified by the **criticalActivity** OperationalTask.

Effects based planning can also be used with the sequence diagrams and StateMachines as shown in the next section.

A.2.5.3 Product Detailed Description

OV-5 describes capabilities, Operational Activities (or tasks), 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 a Capability, OperationalThread, or a Business Sub-Function. OV-5 can be used to:

- Clearly delineate lines of responsibility for activities when coupled with OV-2
- Show areas where a non-materiel solution may allow for mission success.

- Uncover unnecessary Operational Activity redundancy
- Make decisions about streamlining, combining, or omitting activities
- Refine or flag issues, opportunities, or Operational Activities and their interactions (information flows among the activities) that need to be scrutinized further
- Provide a necessary foundation for depicting activity sequencing and timing in OV-6a, OV-6b, and OV-6c

A.2.5.4 Semantics

The purpose of the OV-5 is the clarification of roles, responsibilities and order of execution with respect to accomplishment of key mission objectives in the context of the operational enterprise. The OV-5 is a graphical presentation of the externally visible behavior of the operational enterprise, represented by flows of activities allocated to component systems. Significant information flow associated with those activities is also provided in order to develop a strong sense of coupling between behavior and supporting information. The OV-5, coupled with the textual content of requirements and use case specifications provides guidance to the systems development to define success. Many DoDAF approaches suggest starting with the OV-5 to define the missions.

Table A.10 - OV-5 Elements

UPDM Element	Significance to the View
ActivityRealization	Maps an OperationalActivity to the collection of Systems, people, and Resources used to implement that activity.
Effect	An Effect is an action that causes a change in the state of some other element, such as an enemy or physical control over space. Some approaches to activity planning stress achieving certain conditions, such as air supremacy or control of strategic points. OperationalTasks and collaborating OperationalNodes will bring about the effect.
InformationElement	A formalized representation of information used in an operational process. This is the type of information used on the exchange. On activity diagrams, the InformationElement can appear on the input or output pins or as an activity parameter.
InformationExchange	While the InformationExchange is usually associated with the OV-3, on the OV-5 a user can show information exchanges between Nodes, where the nodes appear as Activity Partitions on the Activity Diagram.
Materiel	The physical resources used to accomplish a particular mission. In the OV-5, materiel can include physical resources critical to mission success.
MaterielBehavior	Relates Materiel to some behavior that uses that materiel. An association where one end is a Materiel and the other end is a Behavior stereotyped as either an OperationalActivity, a SystemFunciton, an EventTrace or StateTrace.
MaterielNode	Relates Materiel to a node.
OperationalActivity	A core activity for the mission, this is the main unit of analysis for the OV-5. When determining the granularity of the OperationalActivity, each Activity must trace to a number of supporting SystemFunctions as elaborated in the SV-5.

Table A.10 - OV-5 Elements

OperationalCapability	A set of processes essential to the core mission of the Enterprise. While the specific work flows inside the stereotyped Collaboration that represents the task might change, the basic capability will not. For DoDAF 1.0, using the OperationalCapability is optional. However, the procurement focus on capability analysis suggests that the architect should make use of this mechanism to link mission activities to core capabilities.
OperationalCapabilityEffect	Specifies that an OperationalCapabilityRealization delivers or realizes the Effect
OperationalCapabilityRealization	The collection of activities performed among OperationalNodes allows for the implementation of Operational Capabilities. This realization provides the overview of interactions and nodes on the operational side key to delivering a capability. The activities inside this realization can be further extended to System entities using the OperationalActivityRealization.
OperationalCapabilityRole	An OperationalCapabilityRole defines an association between the concept represented by an OperationalNode and the OrganizationalResource (OperationalActors) that actually perform the operations of the OperationalNodes to accomplish the node's capabilities.
OperationalCapabilityRoleCompetence	Specifies that an OperationalCapabilityRole requires Competence in some area. Lack of this competence, or incompetence, will present a danger to carry out the tasks needed by the mission.
OperationalCapabilityRoleResource	An association showing which resources play an OperationalRole.
OperationalControlFlow	A flow of control from one activity to another in an OperationalActivity.
OperationalInformationFlow	OperationalInformationFlow is the flow of objects, InformationElements, or even DataElements within OperationalActivities. Note that not all modelers maintain a distinction between data and information, so that in some cases have a flow that is "data:" is reguired on the operations side.
OperationalNode	An OperationalNode can define an ActivityPartition, showing that the activities are carried out at the node.
OperationalRole	An OperationalRole can define an ActivityPartition, showing activities that could be carried out by a service provider or a service consumer.
RealizedOperationalCapability	Specifies that an OperationalCapabilityRealization realizes an OperationalCapability.
ResultsOfEffect	In effects based planning, the focus is on the outcome of an Effect. This is that end state, the ResultsOfEffect.
Trigger	An event that will trigger the behavior found in an activity.

A.2.5.5 Relationship to other products

Table A.11 - OV-5 Relationship to other products

View	Relationship Details
OV-2	The activities annotating an operational Node in an OV-2 map to the OperationalActivities described in an OV -5. Similarly, OV-5 should document the operational Nodes that participate in each OperationalActivity.
OV-3	An InformationExchange in OV-3 should map to one or more information flows (an external Input, an external output, or an output from one OperationalActivity mapped to an input to another) in OV-5, if OV-5 decomposes to a level that permits such a mapping. Above that level of decomposition, a single information flow in an OV -5 may map to more than one InformationExchange (or none, if the information flow does not cross Node boundaries). Note that a physical flow is NOT represented as an information flow. A review of physical exchanges in a model requires a custom view.
OV-6	A rule may define conditions that constrain the execution an OperationalActivity in a specific way, or constrain the organization or human role authorized to execute an OperationalActivity. Effects in the OV-6b map to OperationalActivities in OV-5.
SV-5	Operational Activities in SV-5 match OperationalActivities in OV-5.

A.2.6 Operational Rules, State Transitions and Event-Trace Description (OV-6)

A.2.6.1 Product Definition

Operational Rules Model

The Operational Rules Model specifies operational or business rules that constrain operations. While other OV products (e.g., OV-1, OV-2, and OV-5) describe the structure of the operational mission—what the mission activities achieve—for the most part, they do not describe what the operations must do, or what it cannot do. At the mission level, OV-6a may consist of doctrine, guidance, rules of engagement, and so forth. At the operation level, rules may include such things as a military Operational Plan. At lower levels, OV-6a describes the rules under which the architecture or its nodes behave under specified conditions. Such rules can be expressed in a textual form, for example, “If (these conditions) exist, and (this event) occurs, then (perform these actions).”

At a top level, rules should at least embody the concepts of operations defined in OV-1 and should provide guidelines for the development and definition of more detailed rules and behavioral definitions that will occur later in the architecture definition process. Rules are statements that define or constrain some aspect of the mission, or the architecture. They assert operational structure or to control or influence the mission thread. As the product name implies, the rules captured in OV-6a are operational (i.e., mission-oriented) not systems-oriented. These rules can include such guidance as the conditions under which operational control passes from one entity to another, or the conditions under which a human role is authorized to proceed with a specific activity.

Operational State Transition Description

The Operational State Transition Description describes the state of an operation, including any of the Nodes or entities external to the system that influence mission success. Each transition can specify an event. An Operational State Transition Description shows the explicit impact of the Operational Activities on the State of the participating Actors or nodes.

Operational Event Trace Description

The Operational Event- Trace Description provides a time-ordered examination of the information exchanges between participating Nodes as a result of a particular scenario.

A.2.6.2 UPDM Notation

Rules Model

UML allows the expression of a constraint with the user indicating the language used. UPDM does not attempt to alter the UML structure of a constraint. While the constraints will be applied to the individual elements in the model, this view typically is delivered as a document that exports all the operational rules in the architectural description. A template would typically be provided and tailored to the organizational audience. To resue constraints throughout the model, have the constraint reference an object in the model.

State Transition Description

A StateMachine Diagram can be used to elaborate the OperationalActivities from the OV-5. UML 2 State Diagrams offer a number of features for showing complex state changes and some implementations use executable UML from a State Diagram. A StateMachine cold also apply to an operational node or some other operational element. More advanced features of StateMachines, such as Composite and Orthogonal States coupled with action languages on transitions will enable advanced capabilities.

A StateMachine can also be used for assessing the effects of the activities. In such a case, the state transitions are triggered by the impact of the mission activities. The diagram below shows the most basic steps through such an approach, moving from a chaotic to a desired state. Note the UPDM stereotype is at the level of the state machine, so EffectsBasedStates has the UPDM stereotypes, although not displayed.

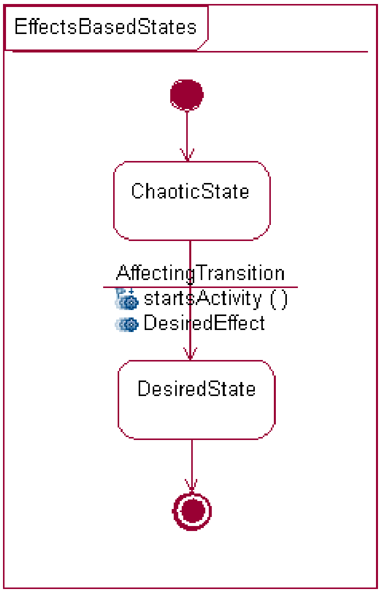


Figure A.7 - State Transition Diagram

This flow starts with a state, ChaoticState: A Trigger (Event) initiates a Transition, AffectingTransition, from one State to other State, the DesiredState. A Trigger is the execution of an OperationalTask, i.e., the operation of the event - startsActivity(). An OperationalNode executes the operation, that is, it performs the OperationalActivity specified by the OperationalTask. (i.e., is the Method of the OperationalTask). Performing this OperationalActivity starts the Transition.

The Transition's Effect, DesiredEffect, is the element that effects the change in state. The Effect is defined as an OperationalTask, criticalActivity(), that specifies an OperationalActivity (the OperationalTasks' Method) that, when performed, effects a change, the transition from ChaoticState to DesiredState. Performing the OperationalActivity results in the end state, DesiredState of the Transition, AffectingTransition.

Operational Event Trace Description

A Sequence Diagram that can elaborate the OperationalActivities as well as show information exchanges between Operational Nodes. Depending on the nature of the architecture project, a modeler can start with the precise detail of the information exchanges in a Sequence Diagram or they can fill them out later after having shown the high level exchanges in the OV-2 and OV-3. For the Sequence Diagram, the modeler could populate the diagram with elements reflecting Nodes that collaborate in each flow or scenario. The diagram below shows a specific message exchange that reflects the Needline shown in the OV-2.

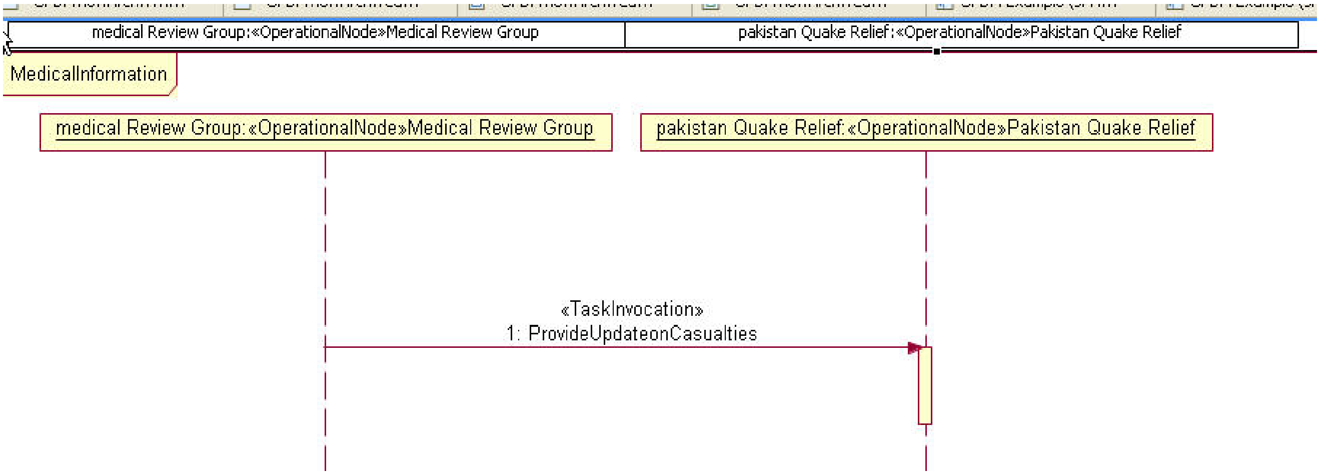

medical Review Group:«OperationalNode»Medical Review Group
pakistan Quake Relief:«OperationalNode»Pakistan Quake Relief
MedicalInformation
medical Review Group:«OperationalNode»Medical Review Group
pakistan Quake Relief:«OperationalNode»Pakistan Quake Relief
«TaskInvocation»
1: ProvideUpdateonCasualties

Figure A.8 - Event Trace Example

The Sequence diagram can also be used to review effects based planning, as shown below. The initial activity invokes the startsActivity OperationalActivity, startsActivity() that in the CommandNode. In the startsActivity, a message, a TaskInvocation is sent to NodeProvidingCapability invoking the criticalActivity(). Its methods specify the OperationalActivity, CriticalActivity, and the DesiredEffect, an OpaqueBehavior that allows specification of the Effect in terms that may take any desired form reflecting the ActivityDiagram that diagrams the CriticalActivity.

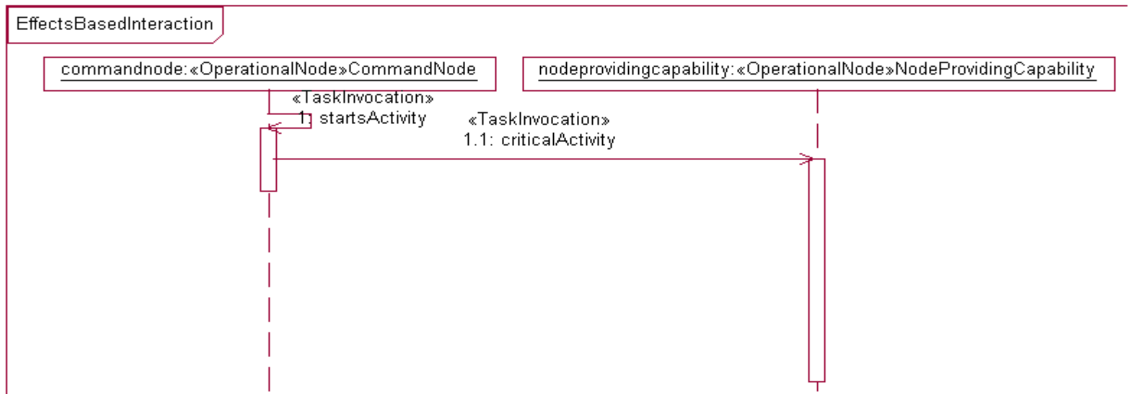

EffectsBasedInteraction
commandnode:«OperationalNode»CommandNode
nodeprovidingcapability:«OperationalNode»NodeProvidingCapability
«TaskInvocation»
1: startsActivity
«TaskInvocation»
1.1: criticalActivity

Figure A.9 - Sequence Diagram

A.2.6.3 Semantics

The purpose of the OV-6 is to capture constraints, operational conditions and sequences related to the operational processes used to achieve mission success. These diagrams can store a precise overview of the messages and information exchanges needed to carry out operational activities.

The OV-6 should fully describe externally visible behavior (from the perspective of the subject system) for each of the flows and scenarios associated with the operational activities and any Use Cases.

Table A.12 - OV-6 Elements

UPDM Element	Significance to the View
CausesEffect	For effects based planning and business modeling, this shows that behavior associated with a transition causes the effect of switching the StateMachine for the element to a desired state.
Effect	An Effect is an action that causes a change in the state of some other element, such as an enemy or physical control over space. Some approaches to activity planning stress achieving certain conditions, such as air supremacy or control of strategic points. OperationalTasks and collaborating OperationalNodes will bring about the effect.
InformationElement	A formalized representation of information used in an operational process. This is the type of information used on the exchange. On activity diagrams, the InformationElement can appear on the input or output pins or as an activity parameter.
InformationExchange	Information Flows on a Sequence Diagram or State Machine represent an InformationExchange. A user can show information exchanges between Nodes by the placement of these flows in the diagrams.
Location	A physical location may be key to an operational mission, and the state of that location may also be relevant. For example, the high ground at Gettysburg needed to be held by the folks from Maine.

Table A.12 - OV-6 Elements

Materiel	The physical resources used to accomplish a particular mission.
MaterielBehavior	A relationship where one end is a Materiel and the other end is a Behavior stereotyped as either an OperationalActivity, a SystemFunciton, an EventTrace or StateTrace.
MaterielNode	Indicates a relationship between materiel and a node.
NodeType	Indicates the type of OperationalNode.
OperationalActivity	A core process for the enterprise mission, the OV-6 artifacts elaborate the OV-5 operational activities. .
OperationalEventTrace	A trace of the sequences used in carrying out an Operational Activity.
OperationalInformationFlow	A flow of information between activities.
OperationalNode	Any of the OperationalNodes make prime candidates for State Machines or for use as owners of Lifelines in a sequence diagram.
OperationalRole	The responsibility played by a node when interacting with other elements. A role could include a service provider or service consumer role. Unanticipated Users or other variables like this can be put into the OV-6 and used in simulating possible operational interactions.
OperationalStateTrace	A State Machine showing how an element responds to changes in the operational environment.
OrganizationalResource	A Resource that can be deployed to SystemsNodes and can play OperationalCapabilityRoles.
Policy	An established course of action that must be followed. Policies can be the source of rules that show up in the operational activities.
Resource	Something that is able to supply functionality, information or materiel.
ResultsOfEffect	In effects based planning, the focus is on the outcome of an Effect. This is that end state, the ResultsOfEffect.
Rule	A constraint that can be applied to any element in the operational view of the UPDM.

A.2.7 Logical Data Model (OV-7)

A.2.7.1 Product Definition

The Logical Data Model describes the structure of an architecture domain’s information and the structural business process rules (defined in the architecture’s Operational View) that govern that information. This data model provides a definition of architecture domain data types, their attributes or characteristics, and their interrelationships.

A.2.7.2 UPDM Notation

OV-7 may be modeled in UML using a class diagram. Classes that appear in an OV-7 class diagram correlate to OV-3 Information Elements and OV-5 or OV-6 inputs, outputs, and controls. The model may in addition link to an external source that could provide a database schema or some other domain description of information elements. Given the difficulty of managing multiple information stores at scale within the defense domain indicates this area will likely develop new mechanisms to describe and relate information. UPDM can incorporate these new mechanisms and describe the information use within the model as new methods emerge.

A.2.7.3 Product Detailed Description

OV-7 defines the architecture domain's system data types (or entities) and the relationships among the system data types. For example, if the domain is missile defense, some possible system data types may be trajectory and target with a relationship that associates a target with a certain trajectory. On the other hand, architecture data types for the DoDAF (i.e., DoDAF-defined architecture data elements, AV-2 data types, and CADM entities) are things like an Operational Node or Operational Activity. OV-7 defines each kind of system data type associated with the architecture domain, mission, or business as its own entity, with its associated attributes and relationships. These entity definitions correlate to OV-3 Information Elements and OV-5 inputs, outputs, and controls.

A.2.7.4 Semantics

The purpose of the OV-7 is to reflect the structure and flow of key information being used to achieve the functionality expressed in the Enterprise Use Cases. The content of this product is should be directly attributable to the IO Entities identified during construction of the OV-6. Often, different organizations may use the same entity name to mean very different kinds of system data with different internal structure. This situation will pose significant interoperability risks, as the data models may appear to be compatible. An OV-7 may be necessary for interoperability when shared system data syntax and semantics form the basis for greater degrees of information systems interoperability, or when a shared database is the basis for integration and interoperability among business processes.

The elaboration of the InformationElements is the minimum requirement for the OV-7. Additional definition could be added to this view as needed

Table A.13 - OV-7 Elements

UPDM Element	Significance to the View
InformationElement	The information items used by the operational activities and sent between Nodes.

A.2.7.5 Relationship to other products

Table A.14 - Relationship of OV-7 to other products

View	Relationship Details
OV-3	An InformationElement in OV-3 should be constructed of entities in OV-7.
OV-6a	The rules in OV -6a may reference the elements of OV-7 to constrain their structure and validity.

A.3 Systems View

The SystemsView (SV) provides graphical and textual products that describe systems that support operations. System resources support the OperationalActivities and facilitate the exchange of information among operational Nodes. SV products focus on the planning for and representation of actual systems, moving from “as is” to “to be” architectures. The relationship between architecture data elements across the SV to the OV can be exemplified as systems are procured and fielded to support organizations and their operations.

A.3.1 Systems Interface Description (SV-1)

A.3.1.1 Product Definition

The Systems Interface Description depicts the communication needs between systems to support operations. With the need to build architectures that support services as well as net-centric operations, the communication requirements increase in complexity with more layers to make it easier to plug and play elements in the architecture. UPDM considers the SV-1 in terms of any communication requirements between Systems. UPDM also introduces the notion of SystemFunctions as provided and required interfaces rather than operations on a System, although ultimately the System must have an operation that implements the interfaces. UPDM also provides mechanisms for Systems that provide and consume services, supporting the component based architectures now common across major enterprises.

The SV-1 also includes information about the SystemFunctions that these Systems implement. The SV-1 can also provide information about the location of the Systems if relevant. The confusingly named DoDAF System Interface represents a communication need between Systems, not an interface in the UML sense of the term. In SV-1, the SystemInterfaces are between Systems, not SystemNodes, but the communication needs might well pass between SystemNodes if the two Systems reside on different nodes.

The term *system* in the framework is used to denote a family of systems (FoS), system of systems (SoS), nomenclatured system, or a subsystem. An item denotes a hardware or software item. Only systems, subsystems, or hardware/software items and their associated standards are documented in this product, where applicable. Details of the communications infrastructure (e.g., physical links, communications networks, routers, switches, communications systems, satellites) are documented in the Systems Communication Description (SV-2).

A.3.1.2 UPDM Notation

The SV-1 is expressed as a Structure Diagram, use the Component Diagram, Deployment Diagram, or Composite Structure Diagram as needed. Those using SysML can also use an Internal Block Diagram. Given that the key entity in the SV-1 is a System, the architect should choose the type of diagram most appropriate to the type of classifier used to represent a System. If the Systems are Components, then a Component Diagram could be used with the SystemInterfaces modeled as associations. For decomposition, the SV-1 can be a Composite Structure Diagram with SystemNodes (localities-physical elements) and Internal and External Ports joined together with SystemInterfaces modeled as Connectors. The diagram below shows the most basic SV-1, with SystemInterfaces indicating the connectivity needs between Systems. If the SystemInterface represents a key interface, this is indicated on the attributes of the association between the Systems.

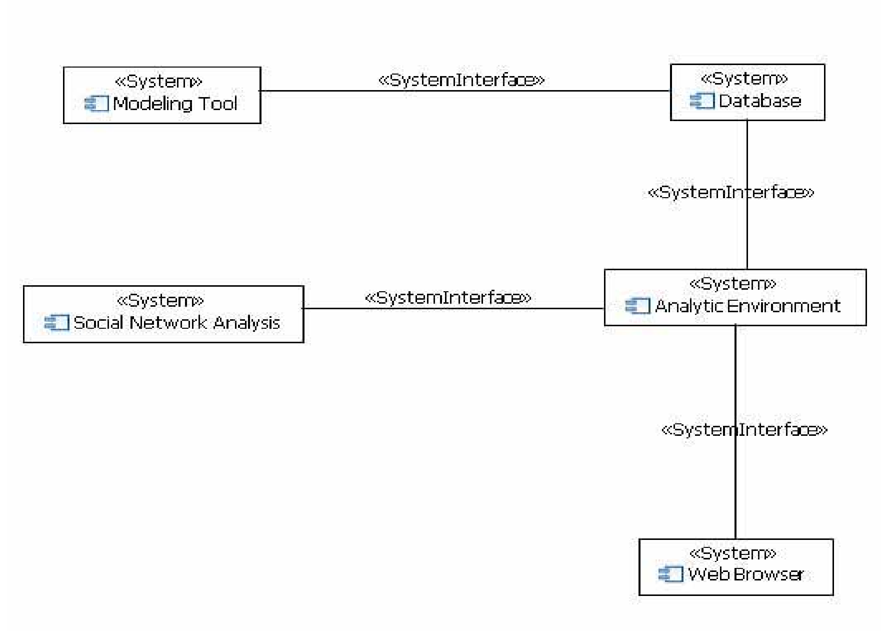


Figure A.10 - Basic SV-1 Diagram

This diagram shows the components required and provided interfaces to show the System Functions carried out or needed by that component. UPDM was designed to allow the modeler to use the richness of UML for component and system design and deployment and then mark-up those models to indicate the DoDAF significance of the relationship. Within the modeling tool, these same relationships could show additional detail as in the example below.

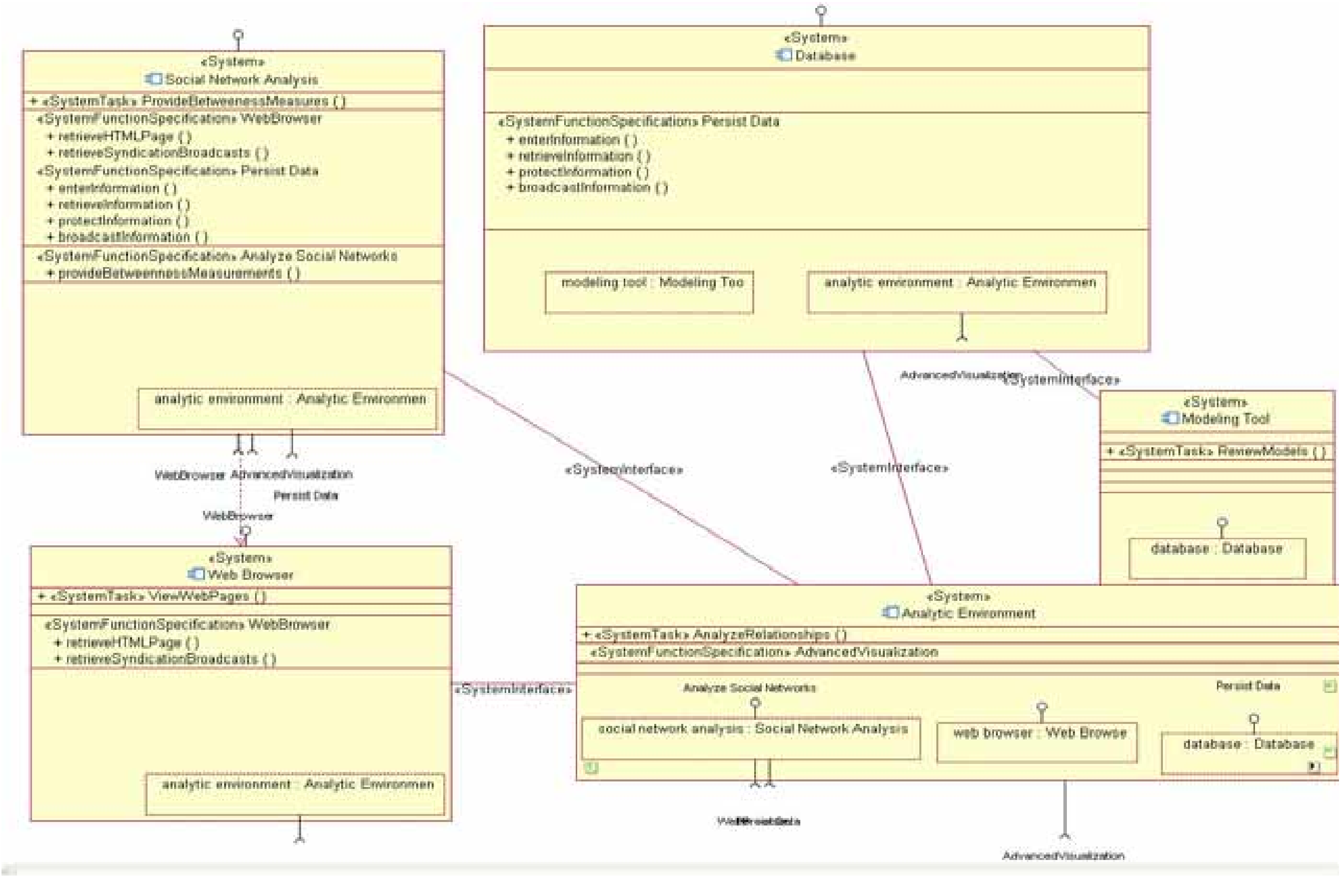

«System»
Social Network Analysis
+ «SystemTask» ProvideBetweenessMeasures ( )
«SystemFunctionSpecification» WebBrowser
+ retrieveHTMLPage ( )
+ retrieveSyndicationBroadcasts ( )
«SystemFunctionSpecification» Persist Data
+ enterInformation ( )
+ retrieveInformation ( )
+ protectInformation ( )
+ broadcastInformation ( )
«SystemFunctionSpecification» Analyze Social Networks
+ provideBetweennessMeasurements ( )
analytic environment : Analytic Environmen
«System»
Database
«SystemFunctionSpecification» Persist Data
+ enterInformation ( )
+ retrieveInformation ( )
+ protectInformation ( )
+ broadcastInformation ( )
modeling tool : Modeling Too
analytic environment : Analytic Environmen
AdvancedVisualization
«SystemInterface»
«System»
Modeling Tool
+ «SystemTask» ReviewModels ( )
database : Database
WebBrowser
AdvancedVisualization
Persist Data
WebBrowser
«SystemInterface»
«SystemInterface»
«System»
Web Browser
+ «SystemTask» ViewWebPages ( )
«SystemFunctionSpecification» WebBrowser
+ retrieveHTMLPage ( )
+ retrieveSyndicationBroadcasts ( )
analytic environment : Analytic Environmen
«SystemInterface»
«System»
Analytic Environment
+ «SystemTask» AnalyzeRelationships ( )
«SystemFunctionSpecification» AdvancedVisualization
Analyze Social Networks
Persist Data
social network analysis : Social Network Analysis
web browser : Web Browse
database : Database
AdvancedVisualization

Figure A.11 - SV-1 Diagram in Modeling Tool Detail

The more detail on this SV-1 still contains the SystemInterface relationship between components but adds more information which may or may not be relevant to those viewing the SV-1. This version includes information on provided and required interfaces that take the UPDM stereotype <<SystemFunctionSpecification>> which extends UML Interface. A modeler could also include a Service Specification in this context if desired. The <<SystemTask>> indicates an operation on a System which will invoke a SystemFunction. The above diagram, while containing a richer information set, may not communicate the needed SystemInterfaces as efficiently.

The diagram below comes from the example planning model used to test out UPDM.

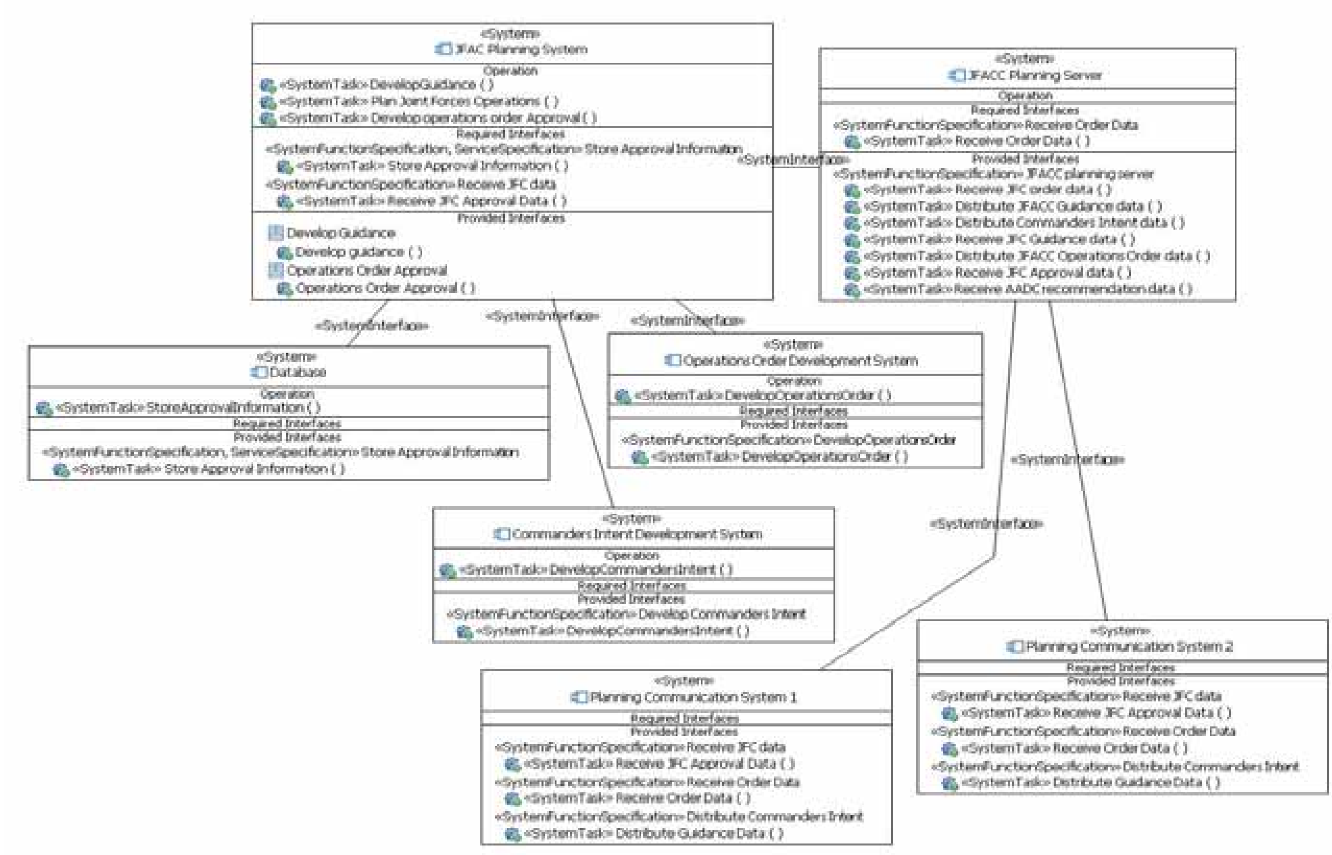

«System»
JFAC Planning System
Operation
«SystemTask» DevelopGuidance ( )
«SystemTask» Plan Joint Forces Operations ( )
«SystemTask» Develop operations order Approval ( )
Required Interfaces
«SystemFunctionSpecification, ServiceSpecification» Store Approval Information
«SystemTask» Store Approval Information ( )
«SystemFunctionSpecification» Receive JFC data
«SystemTask» Receive JFC Approval Data ( )
Provided Interfaces
Develop Guidance
Develop guidance ( )
Operations Order Approval
Operations Order Approval ( )
«SystemInterface»
«System»
JFACC Planning Server
Operation
Required Interfaces
«SystemFunctionSpecification» Receive Order Data
«SystemTask» Receive Order Data ( )
Provided Interfaces
«SystemFunctionSpecification» JFACC planning server
«SystemTask» Receive JFC order data ( )
«SystemTask» Distribute JFACC Guidance data ( )
«SystemTask» Distribute Commanders Intent data ( )
«SystemTask» Receive JFC Guidance data ( )
«SystemTask» Distribute JFACC Operations Order data ( )
«SystemTask» Receive JFC Approval data ( )
«SystemTask» Receive AADC recommendation data ( )
«SystemInterface»
«SystemInterface»
«SystemInterface»
«System»
Database
Operation
«SystemTask» StoreApprovalInformation ( )
Required Interfaces
Provided Interfaces
«SystemFunctionSpecification, ServiceSpecification» Store Approval Information
«SystemTask» Store Approval Information ( )
«System»
Operations Order Development System
Operation
«SystemTask» DevelopOperationsOrder ( )
Required Interfaces
Provided Interfaces
«SystemFunctionSpecification» DevelopOperationsOrder
«SystemTask» DevelopOperationsOrder ( )
«SystemInterface»
«SystemInterface»
«System»
Commanders Intent Development System
Operation
«SystemTask» DevelopCommandersIntent ( )
Required Interfaces
Provided Interfaces
«SystemFunctionSpecification» Develop Commanders Intent
«SystemTask» DevelopCommandersIntent ( )
«System»
Planning Communication System 2
Required Interfaces
Provided Interfaces
«SystemFunctionSpecification» Receive JFC data
«SystemTask» Receive JFC Approval Data ( )
«SystemFunctionSpecification» Receive Order Data
«SystemTask» Receive Order Data ( )
«SystemFunctionSpecification» Distribute Commanders Intent
«SystemTask» Distribute Guidance Data ( )
«System»
Planning Communication System 1
Required Interfaces
Provided Interfaces
«SystemFunctionSpecification» Receive JFC data
«SystemTask» Receive JFC Approval Data ( )
«SystemFunctionSpecification» Receive Order Data
«SystemTask» Receive Order Data ( )
«SystemFunctionSpecification» Distribute Commanders Intent
«SystemTask» Distribute Guidance Data ( )

Figure A.12 - SV-1 System and SystemInterface example

The SV-1 can also show some of the interfaces as ServiceSpecifications if needed. For the JFAC example, the SystemFunctionSpecifications, or UML interfaces, can also be Service Specifications.

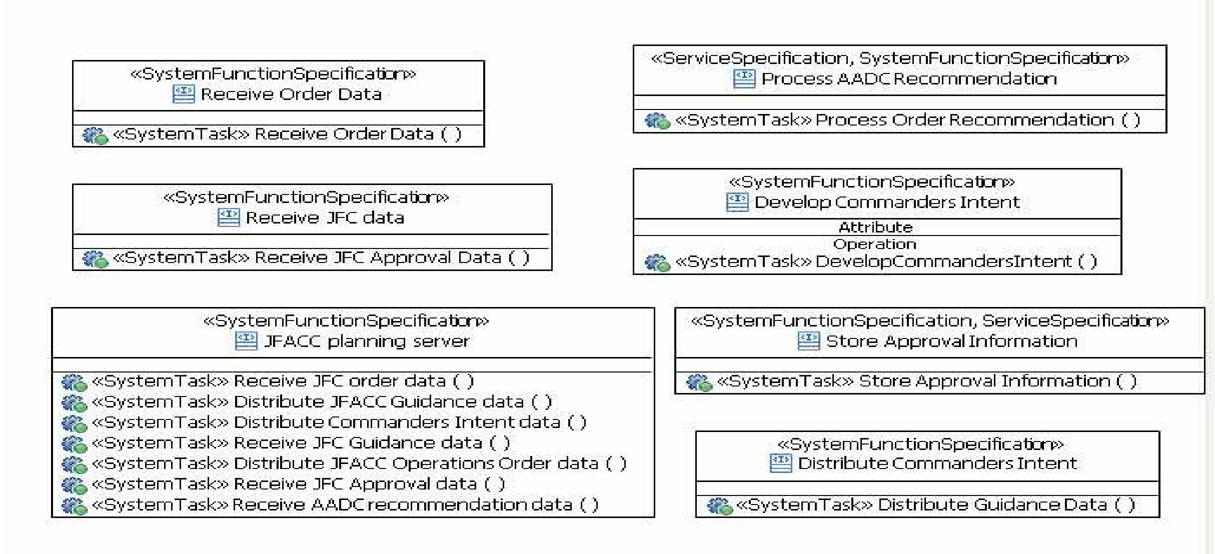

«SystemFunctionSpecification»
Receive Order Data
«SystemTask» Receive Order Data ( )
«ServiceSpecification, SystemFunctionSpecification»
Process AADC Recommendation
«SystemTask» Process Order Recommendation ( )
«SystemFunctionSpecification»
Receive JFC data
«SystemTask» Receive JFC Approval Data ( )
«SystemFunctionSpecification»
Develop Commanders Intent
Attribute
Operation
«SystemTask» DevelopCommandersIntent ( )
«SystemFunctionSpecification»
JFACC planning server
«SystemTask» Receive JFC order data ( )
«SystemTask» Distribute JFACC Guidance data ( )
«SystemTask» Distribute Commanders Intent data ( )
«SystemTask» Receive JFC Guidance data ( )
«SystemTask» Distribute JFACC Operations Order data ( )
«SystemTask» Receive JFC Approval data ( )
«SystemTask» Receive AADC recommendation data ( )
«SystemFunctionSpecification, ServiceSpecification»
Store Approval Information
«SystemTask» Store Approval Information ( )
«SystemFunctionSpecification»
Distribute Commanders Intent
«SystemTask» Distribute Guidance Data ( )

Figure A.13 - Service Specification Diagram Example

A.3.1.3 Semantics

SV-1 identifies the needs of systems to communicate with each other to deliver key functionality for the system. Those system interfaces especially critical to the system are identified as “key interfaces” and require additional documentation and explanation.

The SV-1 creates the foundation for the internal architecture of the subject system. It depicts systems and the interfaces that exist within and between them.

Table A.15 - SV-1 Elements

UPDM Element	Significance to the View
DataExchange	Data Exchanges between Systems from the SV-10c indicate the need for SystemInterfaces that will satisfy the requirement for that Data Exchange. An integrated model will store those relationships, making it easy to determine the exchanges that drive the SystemInterfaces.
Resource	A physical asset used to organize capabilities. In most cases, the SV-1 will use the extension of Resource in SystemsNode, but a modeler could elaborate other Resources on the diagram, potentially to show constraints on the connectivity requirements outlined in this view.
System	The core element that organize the Systems View. Systems deliver the SystemsFunctions that make operations possible.
SystemConnector	If the SV-1 is delivered inside of composite structures, then the conduits between the Systems are modeled as SystemConnectors. The existence of a SystemConnector between two Systems inside a composite structure indicates the requirement for a SystemInterface.

Table A.15 - SV-1 Elements

SystemFunction	Defines the behavior deliverd by a System that enables the execution of operational activities.
SystemFunctionSpecification	A description of the functionality that can be implemented by Systems. In UML terms, an Interface with the implementation in the System.
SystemInterface	A requirement for connectivity between two Systems. A SystemInterface goes both directions and is very different from a UML Interface. In this context, it is more like a conduit that allows for connectivity than a programmatic interface. Note that a SystemInterface does not require connectivity at all times. Such a requirement would be elaborated in the description of why the interface is a KeyInterface.
SystemInterfaceImplements Needline	A relationship that indicates a SystemInterface implements a Needline.
Service	A port that can reside on a System that provides the basis for Service interaction, especially for Web Services. A Service must provide at least one service specification.
ServiceSpecification	A type of interface that specifies a service that a System can either provide or consume. A service may be provided from outside the Enterprise. The SV-2 will detail the connectivity impact of the services.
SystemPort	
SystemServiceConsumer	Any System may act as the consumer of a service, and that includes another service. While this stereotype is optional, it may help you identify elements of a model -- that are not services themselves -- as clients of services.
SystemServiceProvider	Any System may act as the provider of a service. While this stereotype is optional, it may help you identify elements of a model -- that are not services themselves -- as a provider of services.
SystemsNode	A type of Resource that can house Systems. SystemsNodes can provide important boundaries and constraints for some System deployments. SystemsNodes can represent facilities, platforms, units, and locations.
SystemsNodeElements	Specifies the elements association with a SystemsNode.
SystemTask	An operation on the System that can invoke a SystemFunction.
RealizedSystemSpecification	Specifies that an System or one of its sub stereotypes realizes an interface stereotyed SystemFunctionSpecification or one of its sub stereotypes

A.3.1.4 Relationship to other products

Table A.16 - SV-1 Relationship to other products

View	Relationship Details
SV-2	A SystemInterface in SV -1 is implemented by communications link(s) or communications network(s) in SV -2. The relationship between a System and the physical hardware that the system is deployed on can also be shown using the SV-2.

Table A.16 - SV-1 Relationship to other products

SV-3	One entry in SV-3 matrix represents one SystemInterface in the SV-1.
SV-4	The SV-4 defines the flow of SystemFunctions and the operations on SystemFunctions that are executed by Systems defined in SV -1.
SV-5	The System Function Interfaces implements by the Systems in SV-1 map to the SystemFunctions in SV-5.
SV-6	Each System DataElement appearing in a System DataExchange is graphically depicted by one of the SystemInterfaces in SV-1; the communication network supports one or more System DataExchanges.
SV-7	The performance parameters of SV-7 apply to Systems, subsystems, and SystemHardware/software items of SV-1.
SV-8	The Systems, subsystems, and SystemHardware/software items of SV-8 match the corresponding elements in SV-1.
SV-9	Timed technology forecasts in SV-9 impact Systems, subsystems, and SystemHardware/software items of SV -1.
SV-10	The messages between Systems or SystemFunctions as shown in the SystemEventTrace indicate the need for a SystemInterface between the two elements on the SV-1.

A.3.2 Systems Communications Description (SV-2)

A.3.2.1 Product Definition

The Systems Communications Description 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 the communication needs between Systems as described in SV-1. SV-2 shows the communications details of SV-1 connection requirements, what DoDAF calls the SystemInterface.

SV-2 documents the specific communications links or communications networks (e.g., Interlink or Joint Worldwide Intelligence Communications System [JWICS]) and the details of their configurations through which systems interface. The SV-2 can also show the deployment of the Systems in SV-1. While SV-1 depicts communication requirements between Systems, SV-2 contains a detailed description of how each SV-1 communication requirement is implemented (e.g., composing parts of the implemented interface including communications systems, multiple communications links, communications networks, routers, and gateways).

Additional SV-2 views can define the ports for each System and the protocol/hardware stack that is specified for each of these ports. The SV-2 can also define the protocol stack used by a connection between two ports. The SV-2 can also show System connectivity clusters that group connectors between ports into logical connections between Nodes.

A.3.2.2 UPDM Notation

The SV-2 can use a Deployment Diagram, Composite Structure Diagram, or SysML Internal Block Diagram with allocations. Any of these approaches will provide the information needed to isolate the CommunicationPaths needed to support the Systems.

Deployment Diagram

A Deployment Diagram will not only describe the communication system in the SV-2 but also show the Systems with the SystemHardware and Software. A deployment diagram can also store relevant information used in the performance parameters and other SV artifacts. Most systems of any scope require a Deployment Diagram; the SV-2 is the place for that diagram in UPDM. For more detailed decomposition, one can provide details on the specific communication needs related to specific Systems.

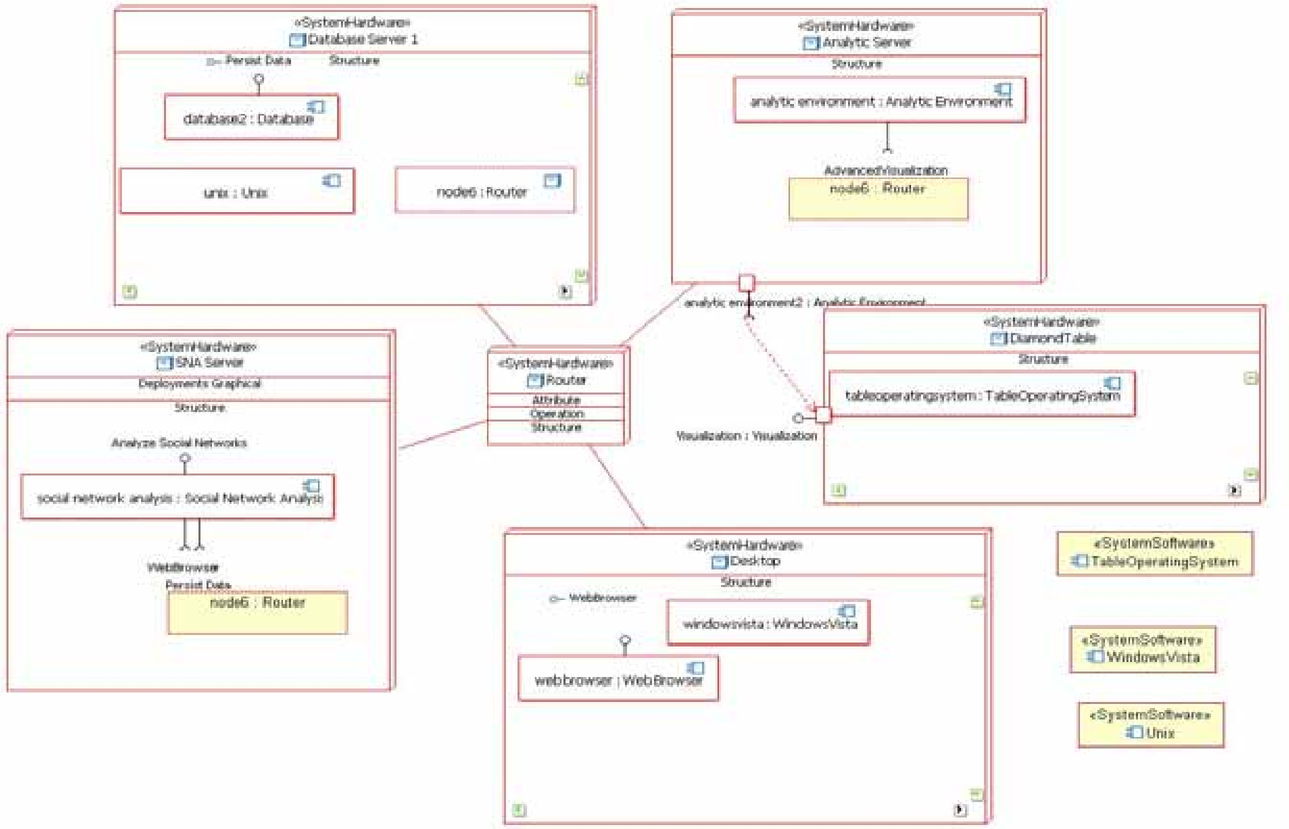


Figure A.14 - Deployment Diagram

The Deployment Diagram shows six UPDM <<SystemHardware>> elements. SystemHardware in UPDM extends System, so these are also types of systems, but the SystemHardware stereotype allows for mapping these elements to system evolution views that explicitly call for SystemHardware. Of course, one projects hardware in the background may be another projects System under development, so the boundaries here will vary. The SystemHardware elements here are stereotypes of the UML Node (not to be confused with the DoDAF Node). The associations between the SystemHardware elements show the possible routes for communication. The components described in the SV-1 are shown here deployed on a specific server. The communication links available to the server need to provide the connectivity needed for the SystemInterfaces associated with that System. The system routes most of the information across the router, but there is one relationship between the Analytical environment and the Diamond Table that has hardware dependencies associated with the Port.

The key for this illustration is not to build a deployment diagram that replicates the style of this diagram, but to build a typical UML deployment diagram and then stereotype the Node and Component elements with the proper UPDM stereotypes to make them compliant DoDAF deliverables. Deployment diagrams offer a rich set of features—more than most reviewers want to look at in a single diagram. The SV-2 view should emerge naturally from the normal use of deployment diagrams on a project.

Note that the deployment diagram supports a number of modeling constructs. Some modelers may be able to deliver the SV-1 and SV-2 with the same diagram. The SV-2 also easily handles the SystemSoftware on the UML Nodes. For example, the Desktop is using Windows Vista.

Composite Structure Diagram

For further analysis of some Systems, a Composite Structure might provide the best view. The diagram below includes ports on the system components. The port can also include the ProtocolStack by creating a relationship between the port and an instance of ProtocolStack from the Model Library. This allows for the reuse of common types of ProtocolStacks. For additional information on the Port, extend the Model Library with a custom type.

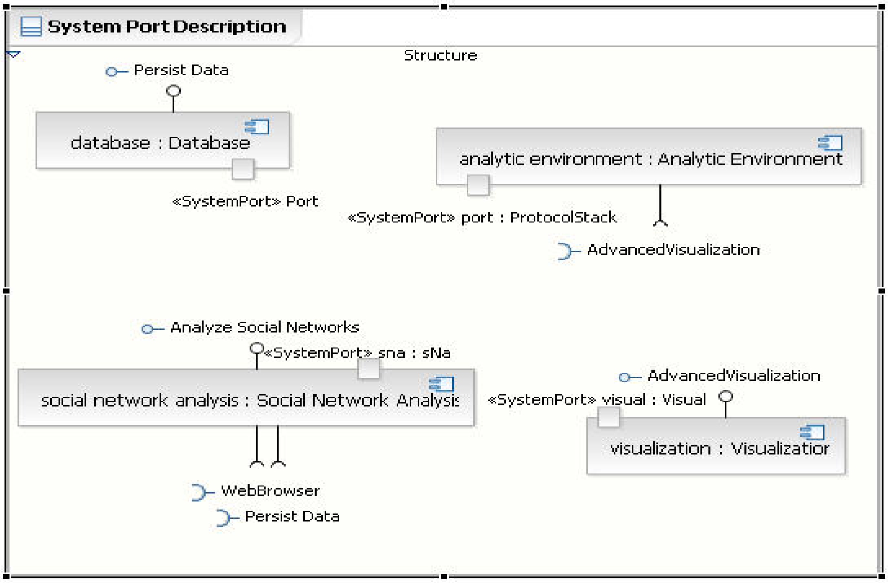


Figure A.15 - Composite Structure Diagram

Internal Block Diagram (SysML)

The internal block diagram provides system implementation detail critical for those building Systems that involve substantial physical elements. If the System stereotype is applied to a SysML block, which can be done with core UPDM and the SysML profile, then a modeler can deliver a block diagram to communicate their SV-2. If the user uses Level 0 UPDM they can continue to use Deployment Diagrams in addition to the Block Diagram. If the user is following the advanced SysML constraints in Level 1, then they are restricted to using Block Diagrams exclusively.

A.3.2.3 Semantics

SV-2 can be used to document how system communication requirements (described in SV-1) are supported by computational resources. SV-2 can also detail the ports for the Systems, allowing detailed specification of protocols and enabling accurate assessment of hardware dependencies for Systems. The SV-2 also supports rigorous analysis of the communications infrastructure to support a communication network for the systems. Note that MODAF elaborates three sub-views of the SV-2, including 2a, System Port Description; 2b, System to System Port; and 2c System Connectivity Clusters.

Table A.17 - SV-2 Elements

UPDM Element	Significance to the View
CapabilityConfiguration	The CapabilityConfiguration is all the resources and physical assets that combine to provide a Capability. The SV-2 includes the SystemHardware and SystemSoftware deployment components of the CapabilityConfiguration.
CommPathFromTo	Asserts that there is an association between a CommunicationPath and a System.
CommunicationLink	A single physical connection from one system to another.
CommunicationPath	A CommunicationPath is a set of all the communication links and the systems that implement on path of the needed connection between two Systems. Two systems deployed on different hardware may have a number of different CommunicationPaths. These paths can be used to verify the communications options available for SystemInterfaces. When the SystemInterfaces are key interfaces, then the modeler should look to have multiple CommunicationPaths.
CommunicationPathRealizes SystemInterface	Asserts that a CommunicationPath realizes a SystemInterfacde
CommunicationPathRealization	Asserts that the communicationPath is realized by a CommunicationsSystem described in terms of CommunicationsSystems and and Systems connected by CommincationLinks
CommunicationPort	The port that defines the hardware requirements for any connection between a System and any SystemHardware used in the implementation of the CommunicationPath.
CommunicationSystem	A system with a primary function to control the transfer and movement of system data rather thanperforming mission application processing.
Network	One or more CommunicationPaths that can be used to support the communication requirements outlined in the SV-1.
NetworkPaths	
ProtocolStatck	The Protocol that is required for connection of the CommunicationPorts to which this ProtocolStack is applied.
SV-2	The package that contains the information needed to build an SV-2.
Service	A port that can reside on a System that provides the basis for Service interaction, especially for Web Services. A Service must provide at least one service specification.
ServiceSpecification	The SV-2 will elaborate the connectivity needs for Services to deliver their functionality.

Table A.17 - SV-2 Elements

System	A System is a deployed Resource that can use the communication Network to satisfy any of the SystemInterfaces with other Systems.
SystemHardware	The UML Node that provides computational resources and must have CommunicationPaths that support the communication requirements of the deployed Systems.
SystemInterface	Indicates that two systems require a communications network in order to perform their SystemFunctions.
SystemPort	A type of port that can support the logical and physical requirements for a UPDM System.
SystemServiceConsumer	Any System may act as the consumer of a service, and that includes another service. While this stereotype is optional, it may help you identify elements of a model -- that are not services themselves -- as clients of services.
SystemServiceProvider	Any System may act as the provider of a service. While this stereotype is optional, it may help you identify elements of a model -- that are not services themselves -- as a provider of services.
SystemsNode	A type of physical asset that can house people and machines needed to implement a mission.
SystemsNodeElements	Specifies the elements associated with a SystemsNode.
SystemSoftware	A type of system that provides the supporting software for Systems, such as operating systems or monitoring software. The SV-2 can show the deployment of SystemSoftware.
SystemSystemSoftware	Asserts a relationship between a System and SystemsSoftware.

A.3.2.4 Relationship to other products

Table A.18 - Relationship of SV-2 to Other Products

View	Relationship Details
SV-1	The communications requirements in SV-1 is implemented by communications link(s) or communications network(s) in SV -2.
SV-7	The performance parameters of SV -7 apply to the SystemHardware and communications performance in the SV-2.
SV-8	The Systems, subsystems, and SystemHardware/software items of SV -8 match the corresponding elements in SV-2.
SV-9	Timed technology forecasts in SV -9 impact Systems, subsystems, and SystemHardware/software items of SV -2.

A.3.3 Systems-Systems Matrix (SV-3)

A.3.3.1 Product Definition

The Systems-Systems Matrix provides detail on the system communication characteristics described in SV-1 for the architecture, arranged in matrix form.

A.3.3.2 UPDM Notation

Text matrix derived from Systems and their SystemInterfaces.

A.3.3.3 Product Detailed Description

SV-3 allows a quick overview of all the System Interface characteristics presented in all the Systems. SV-3 can be organized in a number of ways (e.g., by domain, by operational mission phase) to emphasize the association of groups of system pairs in context with the architecture purpose. SV-3 can be a useful tool for managing the evolution of systems and system infrastructures, the insertion of new technologies/functionality, and the redistribution of systems and processes in context.

Since many architectural descriptions do not present SV-1 as one diagram, but rather display select portions of the SV-1, this matrix provides a complete overview of the communication requirements between Systems.

Many types of communication information can be presented in the cells of SV-3. The system-system interfaces can be represented using a number of different symbols and/or color codes that depict different interface characteristics, such as if the connection represents a “key interface”. In addition, protocol or port information from the more detailed MODAF SV-2 series could also appear in this matrix.

A.3.3.4 Semantics

The SV-3 is a matrix view of the system-to-system relationships that exist at any specified level of system decomposition. At a minimum, the matrix should identify which systems have communication needs with other systems.

Table A.19 - SV-3 Elements

UPDM Element	Significance to the View
System	The Systems that deliver the SystemFunctions to achieve the operational missions.
SystemInterface	A requirement for connectivity between two Systems.

A.3.3.5 Relationship to other products

Table A.20 - Relationship of SV-3 to other products

View	Relationship Details
SV-1	A relationship on this matrix indicates a SystemInterface in SV-1.
SV-10	If this diagram includes real Systems and SystemActivityFunctions, then messages between Systems indicate the necessity of a SystemInterface.

A.3.4 Systems Functionality Description (SV-4)

A.3.4.1 Product Definition

The Systems Functionality Description documents functional hierarchies and supporting data flows between SystemFunctions. Although there is a correlation between Operational Activity Model (OV-5) or business-process/ mission definition and the system functional description in SV-4, it need not be a one-to-one mapping. The Operational Activity to Systems Function Traceability Matrix (SV-5) provides the artifact that displays these relationships from the model.

A.3.4.2 UPDM Notation

The functionality description will consist of activity diagrams. Activity diagrams provide a number of features to show the dynamic behavior of the different Systems. In UPDM, the model holds the link between the OperationalActivities and the SystemFunction in a collaboration stereotyped as <<OperationalActivityRealization>>. The realization has a relationship back to the <<OperationalActivity>> with a relationship precisely defining the activity realized by the elements in the realization. Within this collection, UML dynamic diagrams can provide the choreography of Systems needed to carry out the mission activity. One of those dynamic diagrams is the <<SystemFunction>>, a stereotyped Activity. The diagram of that activity provides the SV-4.

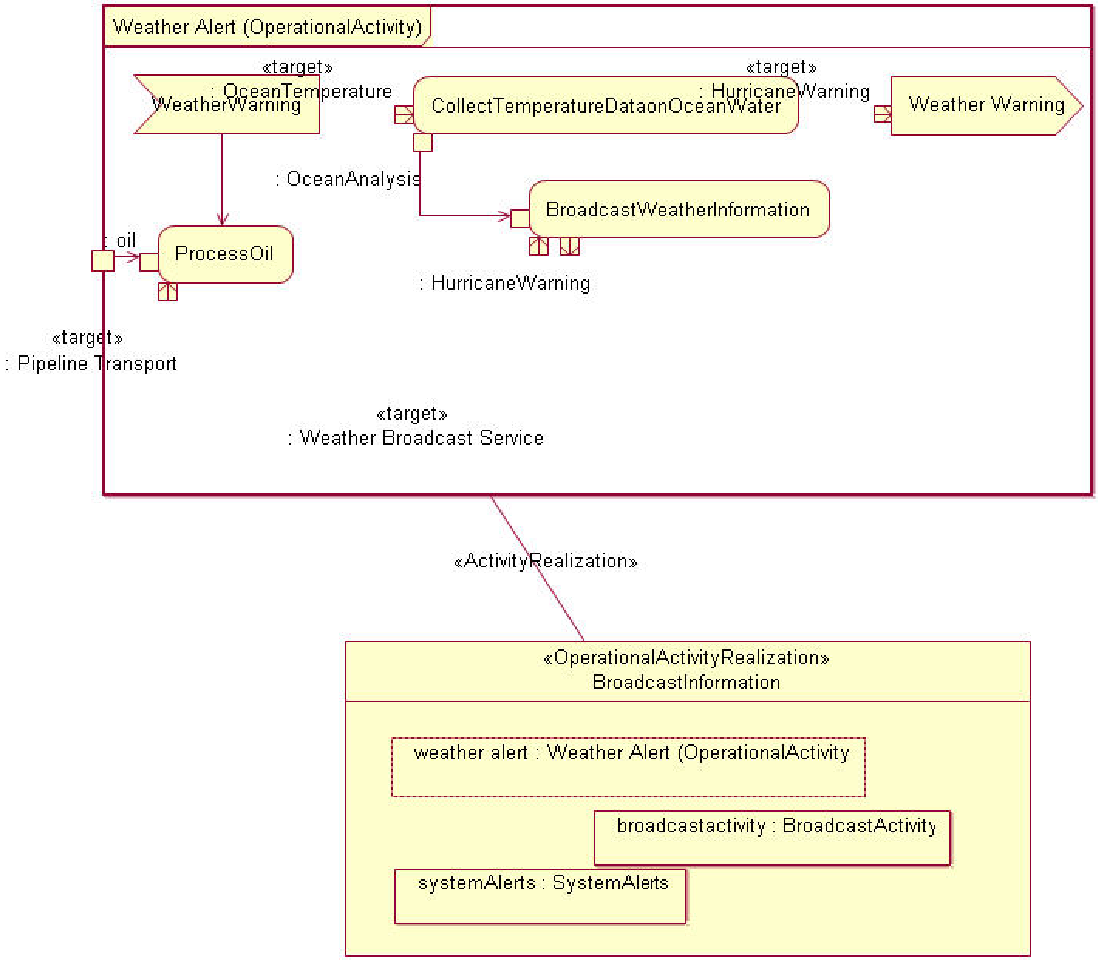

Weather Alert (OperationalActivity)
«target»
: OceanTemperature
WeatherWarning
CollectTemperatureDataonOceanWater
«target»
: HurricaneWarning
Weather Warning
: OceanAnalysis
BroadcastWeatherInformation
: oil
ProcessOil
: HurricaneWarning
«target»
: Pipeline Transport
«target»
: Weather Broadcast Service
«ActivityRealization»
«OperationalActivityRealization»
BroadcastInformation
weather alert : Weather Alert (OperationalActivity
broadcastactivity : BroadcastActivity
systemAlerts : SystemAlerts

Figure A.16 - Simple OperationalActivity Realization

In the above diagram, note that the SystemAlerts and BroadcastActivity are either the SystemFunctions or an activity that acts as a container for SystemFunctions or activities that will call SystemFunctions if stored in a separate hierarchy. The elaboration of the SV-4 element BroadcastActivity, which is a SystemFunction, is done in an activity diagram.

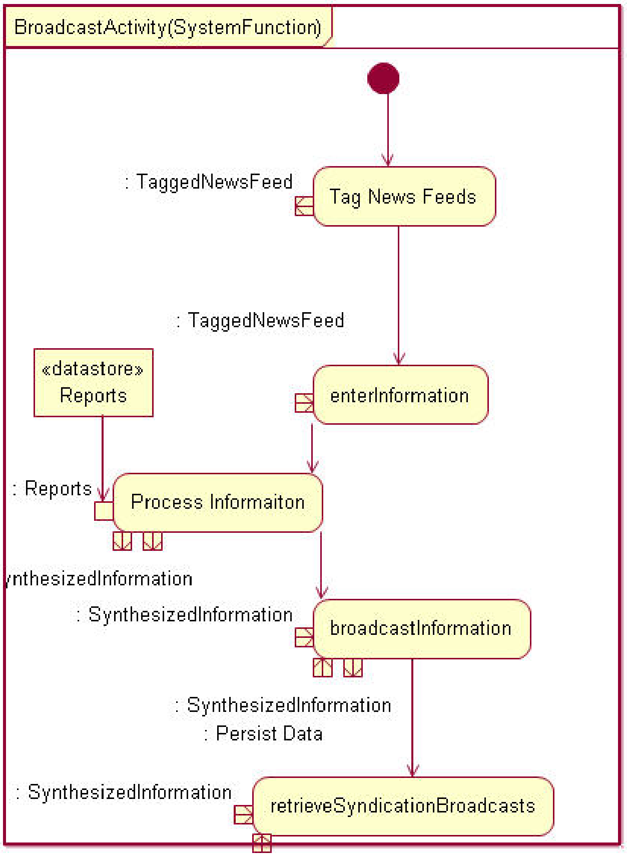
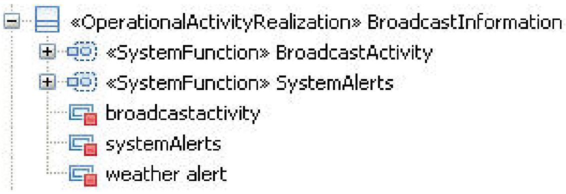


Figure A.17 - Activity Diagram example

Note that the UML visualization may not always show the stereotype for the activity, which have been indicated here by a parenthetical reference. The XMI includes the stereotype and the visualization can be adapted to show either the stereotype or an iconic representation of the stereotype as needed. The toolbar view may indicate the breakdown of the stereotypes. The diagram below shows the way the SystemFunctions live in an OperationalActivityRealization to provide the link to the OV-5.



«OperationalActivityRealization» BroadcastInformation
«SystemFunction» BroadcastActivity
«SystemFunction» SystemAlerts
broadcastactivity
systemAlerts
weather alert

Figure A.18 - Activity Realization and System Function Relation example

The SV-4 in UPDM has been made more explicitly analogous to the OV-5. Indeed, one could mix OV and SV elements on the same activity diagram if required. Some service-oriented approaches start this mixture. Although breaking the separation between system and operational concerns violates some basic DoDAF principles, mixing the elements on the same diagram might be more efficient and with UPDM can be done without losing the system or operational identity of the individual elements. In other words, the distinction between operational and system elements are applied with stereotypes at the ELEMENT level, not at the Diagram level, opening up the user to more efficient use of modeling techniques, especially with regard to the composition of services.

The diagram below shows the breakout of the SystemFunctions from the JFACC planning example. There are many ways to build up a SystemFunction hierarchy, so the point of this is to illustrate a flow of SystemFunctions. In this case, these activities are closely tied to the SystemTasks from the SV-1: the SystemFunctions evoke the SystemTasks which are the operations on the SystemFunction interface. The end result is a set of SystemFunctions decoupled from the deployment detail, but still linked through model elements to the actual Systems that deploy them using a realization relationship to show that the SystemTasks are implementing a SystemFunction specification or a service specification.

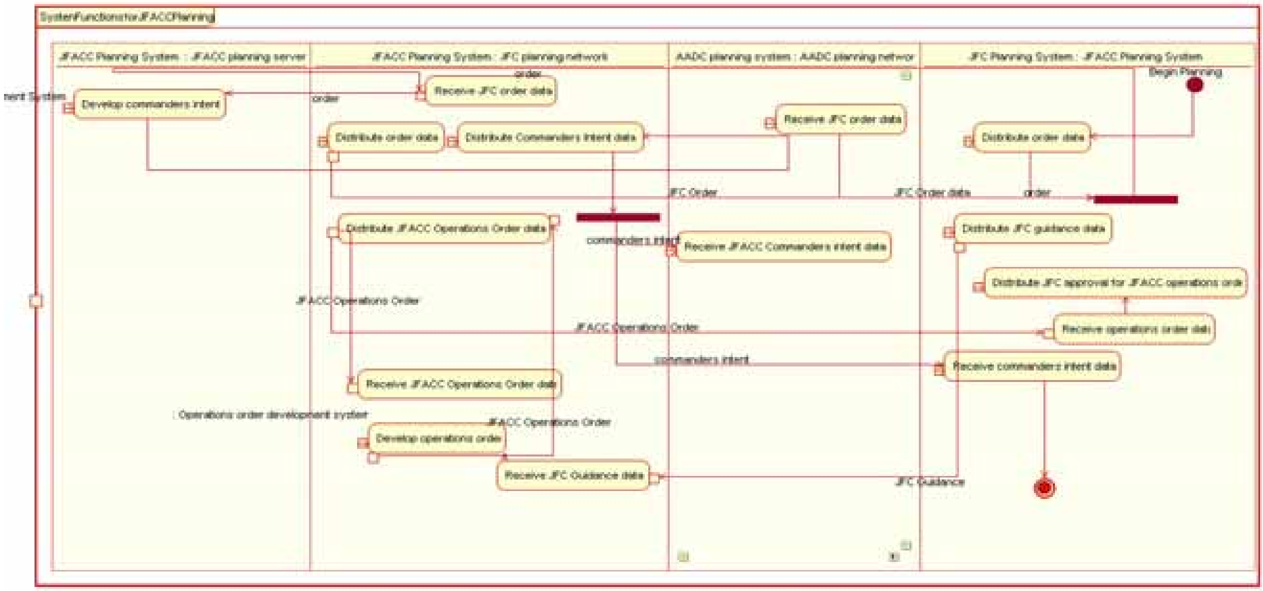

SystemFunctionsforJFACCPlanning
JFACC Planning System : JFACC planning server
JFACC Planning System : JFC planning network
AADC planning system : AADC planning networ
JFC Planning System : JFACC Planning System
Begin Planning
nent System
Develop commanders intent
order
order
Receive JFC order data
Distribute order data
Distribute Commanders intent data
Receive JFC order data
Distribute order data
JFC Order
JFC Order data
order
Distribute JFACC Operations Order data
commanders intent
Receive JFACC Commanders intent data
Distribute JFC guidance data
Distribute JFC approval for JFACC operations orde
JFACC Operations Order
JFACC Operations Order
Receive operations order dat
commanders intent
Receive commanders intent data
Receive JFACC Operations Order dat
: Operations order development syster
JFACC Operations Order
Develop operations order
Receive JFC Guidance data
JFC Guidance

Figure A.19 - SV-4 Activity Diagram

A.3.4.3 Product Detailed Description

SV-4 describes System Functions and the flow of system data among those functions. It is the SV counterpart to OV-5. The scope of this product may be logical, without regard to which Systems implement which functions, or it may be specific to a System's operations. Either way, the actual function called is stereotyped as a <<SystemActivityFunction>> representing a called operation to either the System or the SystemFunction. Typically, the SV-4 will focus at the logical level outlining the System Function Interfaces needed in the architecture.

A.3.4.4 Semantics

The SV-4 can (a) develop a clear description of the necessary system dynamics and the I/O for each system, (b) ensure that the functional connectivity is complete (i.e., that a system's required inputs are all satisfied), and (c) ensure that the functional decomposition reaches an appropriate level of detail.

Object flows are added to the activity flow in order to indicate object inputs and outputs necessary to specified activities. The information content of the SV-4 provides an alternate perspective that allows physical flows in contrast to the content of the SV-10c sequence diagrams with their information flows through messages and parameters. Note that the physical flows in the SV-4 require custom views for additional elaboration. For example, some DoDAF projects use special logistics views if physical flows are important.

Table A.21 - SV-4 Elements

UPDM Element	Significance to the View
ActivityRealization	Maps a SystemFunction to the collection of Systems, people, and Resources used to implement that activity.
DataElement	The elements on the Pins represent DataElements. ObjectFlows can also be used with DataElements.
DataElementSystemSpec Function	Defines the association between a SystemFunction and the DataElements that are are the result of the execution of its SystemActivityFuncitions or the DataElements that are used by those SystemActivityFuncitions
DataExchange	An exchange that carries data elements between Systems or SystemFunctionSpecifications.
Service	The SystemFunction flow in the activity model can include Services that provid functionality.
ServiceSpecification	Details the set of operations provided by the Service.
System	The basic container for resources that perform SystemTasks to deliver SystemFunctions. In the Activity Diagram for the SV-4 they can define activity partitions.
SystemTask	An operation on the System that can invoke a SystemFunction.
SystemFunction	Defines the behavior deliverd by a System that enables the execution of operational activities.
SystemFunctionSpecification	The Activity Diagram can include call operation actions to operations that are on the SystemFunctionSpecification. This allows modeling of System behavior using UML interfaces.
SystemControlFlow	A flow of control from one activity or SystemFunction to another.
SystemInformationFlow	Provides for the flow of data elements between SystemFunctions or other behavior in the Activity Diagram.
SystemServiceConsumer	Any System may act as the consumer of a service, and that includes another service. The service consumer can then take an active role in the description of the system functions.
SystemServiceProvider	Any System may act as the provider of a service. The service provider can play a role in the Activity Diagram that explains SystemFunction behavior.

Table A.21 - SV-4 Elements

SystemsNode	A resource that provides the physical context for an activity. It can own an activity partition, or swimlane. SystemsNodes can represent facilities, platforms, units, and locations.
SystemsNodeElements	Specifies the elements that are a part of a SystemsNode.
Trigger	A Trigger initiates an action for the SystemFunction or any other activity outlined in the ActivityDiagram for SV-4.

A.3.4.5 Relationship to other products

Table A.22 - Relationship of SV-4 to Other Products

View	Relationship Details
OV-3	If any part of an information element in OV -3 originates from or flows to an operational activity that is to be automated, then that Information Element should map to one or more system data elements in SV -6, shown on an object flow in the SV-4.
SV-1	SV-4 shows the dynamic aspect of System Functions that are realized by Systems defined in SV -1.
SV-5	System Functions in SV-4 should map one-to-one to system functions in SV -5.
SV-6	System data flows in SV-4 should map to system data elements appearing in system data exchanges of SV-6.
SV-8	If SV-8 identifies system functions to be implemented in each phase of a system development, they should match the SystemFunctions in SV-4.
SV-10a	If system rules in SV-10a deal with system behavior, they should reference system functions in SV-4.
SV-10b	SystemFunctions in SV-4 may be associated with either states or state transitions in SV-10b.
SV-10c	Events in SV-10c map to system data flows in SV-4.

A.3.5 Operational Activity to Systems Functionality Traceability Matrix (SV-5)

A.3.5.1 Product Definition

The Operational Activity to Systems Function Traceability Matrix specifies the relationships between the set of OperationalActivities and the set of SystemFunctions that support them. The matrix can also be used to show additional relationships between SystemFunctions and OperationalView constructs easily queried from the integrated model. This could include mapping the SystemFunctions to OperationalCapabilities, Doctrine, or StrategicCapability.

A.3.5.2 UPDM Notation

A spreadsheet matrix with no UML diagram for the view. The SV-5 will show the traceability from the enterprise's operational activities and the SystemFunctions. The model maintains this relationship via the OperationalActivityRealization Collaboration that contains SystemFunctions either in the SV-10c package or SV-4 package. See the description for those views for additional modeling details. Any System Function contained in the OperationalActivityRealization will show traceability from mission down to SystemFunction.

A.3.5.3 Product Detailed Description

The Framework uses the terms activity in the OVs and function in the SVs to refer to the same kind of thing-both activities and functions are tasks that are performed, accept inputs, and develop outputs. However, the System Functions depict the main UML interfaces that deliver systems while the operational side shows the business activity mapping. A guiding principle for DoDAF and MODAF is to present technical systems in terms of what operations they enable, rather than evolving an IT infrastructure based purely on technical inertia. The DoDAF descriptions recommend reviewers start with the SV-5 as it provides that major overview of systems and activities. SV-5 provides explicit links between the OV and SV for those reviewing a program.

A.3.5.4 Semantics

The SV-5 provides traceability between OperationalActivities and the functionality of Systems. As the deliverable primarily responsible for showing the links between the Operational View and System View, the SV-5 is often one of the most important deliverables on any project.

Table A.23 - SV-5 Elements

UPDM Element	Significance to the View
OperationalActivity	The core mission activities for the enterprise outlined in the OperationalView.
OperationalActivityRealization	Provides a collection that can detail all the interactions among Systems and SystemFunctions needed to realize a particular activity.
OperationalCapability	A set of capabilities that reference or own OperationalActivities. An SV-5 can include these as an extra reference in the matrix.
OperationalCapabilityRealization	The collaboration among OperationalNodes allows for the implementation of Operational Capabilities. This realization provides the overview of interactions and nodes on the operational side key to delivering a capability. The activities inside this realization can be further extended to System entities using the OperationalActivityRealization. This allows for full traceability from operational capability down to System Function.
SystemFunction	The core functionality delivered by the Systems.
SystemFunctionSpecification	For a “to be” architecture, the SystemFunctions might reflect defined sets of interfaces that will drive the acquisition of futures Systems.

A.3.5.5 Relationship to other products

Table A.24 - Relationship of SV-5 to Other Products

View	Relationship Details
OV-5	Operational activities in SV-5 match operational activities in OV -5.
OV-6	A capability associated with a specific sequence in an OV-6c matches a capability in SV-5.
SV-1	SystemFunctions in SV-1 should map one-to-one to system functions in SV -5.
SV-4	A SystemFunction inside the OperationalActivityRealization will show up in the SV-5.
SV-10	A SystemFunction inside the OperationalActivityRealization will show up in the SV-5.

A.3.6 Systems Data Exchange Matrix (SV-6)

A.3.6.1 Product Definition

The Systems Data Exchange Matrix specifies the characteristics of the data exchanged between Systems. This product focuses on automated exchange of information (from OV-3) as actually implemented from a variety of data sources. Non-automated exchange of information, such as verbal orders, are captured in the OV products only.

A.3.6.2 UPDM Notation

This view will use the UPDM mechanisms for external reference of a view presentation. This view does not have an associated UML diagram. The UPDM Model Library includes some templates that can provide a start to managing the information required in the SV-6.

A.3.6.3 Product Detailed Description

SV-6 describes, in tabular format, data exchanged between systems. The focus of SV-6 is on how the system data exchange is implemented, in system-specific details covering periodicity, timeliness, throughput, size, information assurance, and security characteristics of the exchange. In addition, the system data elements, their format and media type, accuracy, units of measurement, and system data standard are also described in the matrix. SV-6 relates to, and grows out of, the OV-3. The operational characteristics for the OV-3 information exchange are replaced with the corresponding system data characteristics. For example, the Levels of Information Systems Interoperability (LISI) level required for the operational information exchange is replaced by the LISI level achieved through the system data exchange(s). Similarly, performance attributes for the operational information exchanges are replaced by the actual system data exchange performance attributes for the automated portion(s) of the information exchange.

A.3.6.4 Semantics

The SV-6 is a matrix of Data Exchanges (similar to the OV-3) that represent the behavior based-interactions between component systems and subsystems of the subject system. Each row of the matrix represents a Data Exchange, which is comprised of data characteristics of that data transferred in an interaction from the SV-10c sequence diagrams or the SV-4. A distinct Data Exchange is identified for each pair of objects or roles that interact and exchange information. Specific characteristics of the Data Exchange are typically associated with non-functional requirements or design constraints. The content of each IER is representative of an instantiation of a Data Object, where the attributes represent the data characteristics required by the DoDAF.

The emphasis of the SV-6 is on the logical and operational characteristics of the data exchanged. The product is not intended to provide exhaustive capture of all details of data exchanged within the architecture, but as a mechanism understand the most important aspects of significant exchanges.

Table A.25 - SV-6 Elements

UPDM Element	Significance to the View
DataElement	The core data used in machine communication between Systems.
DataElementSystemSpecFunction	Defines the association between a SystemFunction and the DataElements that are are the result of the execution of its SystemActivityFuncitions or the DataElements that are used by those SystemActivityFuncitions.
DataExchange	A message between two Systems. A DataExchange will contain a set of DataElements.
SystemInformaitonFlow	In this context, only DataElements traveling along a SystemInformatinoFlow are eligible for the matrix.
Transaction	The set of information and data exchanges considered as a unit for the purpose of satisfying a request or ensuring database integrity.

A.3.6.5 Relationship to other products

Table A.26 - Relationship of SV-6 to Other Products

View	Relationship Details
SV-1	Each system data element appearing in a system data exchange is graphically depicted by one of the System Interfaces in SV -1.
SV-2	System data flows in SV-4 should map to system data elements appearing in system data exchanges of SV-6.
SV-7	Performance parameters in SV-7 that deal with interface performance and system data exchange capacity requirements should trace to system data exchanges in SV-6.
SV-10b	Events in SV-10b map to triggering events in SV-6.
SV-10c	Events in SV-10c map to triggering events in SV -6.

A.3.7 Systems Performance Parameters Matrix (SV-7)

A.3.7.1 Product Definition

The Systems Performance Parameters Matrix product specifies the quantitative characteristics of Systems as well as supporting hardware and software. Performance parameters include all technical performance characteristics of Systems for which requirements can be developed and specification defined. The complete set of performance parameters may not be known at the early stages of architecture definition, so it should be expected that this product will be updated throughout the system's specification, design, development, testing, and possibly even its deployment and operations life-cycle phases.

A.3.7.2 UPDM Notation

This view will use the UPDM mechanisms for external reference of a view presentation. No UML diagram associated with this view.

The PerformanceMeasure is implemented as a static variable on either a System {xor} SystemHardware. Querying all elements with the static variable typed as PerformanceMeasure produces the SV-7.

A.3.7.3 Product Detailed Description

SV-7 builds on SV-1, SV-2, SV-4, SV-6 and SV-10c by specifying performance parameters for Systems (defined in SV-1), communications details (defined in SV-2), their functions (defined in SV-4 and SV-10c), and their system data exchanges (defined in SV-6). A System could be a family of systems (FoS), system of systems (SoS), network of systems, or an individual System. If the future performance expectations are based on expected technology improvements, then the performance parameters and their time periods should be coordinated with a Systems Technology Forecast (SV-9). If performance improvements are associated with an overall system evolution or migration plan, then the time periods in SV-7 should be coordinated with the milestones in the Systems Evolution Description (SV-8).

In the US federal government, these performance parameters may come from external reference models. (See ExternalReference)

A.3.7.4 Semantics

These particular parameters can often be the deciding factors in acquisition and deployment decisions, and will figure strongly in systems analyses and simulations done to support the acquisition decision processes and system design refinement.

SV-7 describes characteristics considered critical to successful support of the mission objectives assigned to the subject System. Specific content of this view will be determined by the application domain. A notional example is available for reference in the DoDAF specification.

Table A.27 - SV-7 Elements

UPDM Element	Significance to the View
ConformingElement	Any stereotype that extends this element can have related performance parameters.
MeasurementType	An enumeration indicating if the PerformanceMeasure is an actual measure, a baseline measure, or a target measure for a "to be" architecture.
PerformanceMeasure	A specific performance measurement that includes such system information as anticipated response time, graphic quality, or any other performance aspect of a System.
PerformanceMeasurementSet	A set of actual performance measurements related to a specific period of time.
PerformanceParameterSet	A set of performance measures assigned to the System.
System	Performance Parameters will typically provide insight into the requirements for the Systems.

A.3.7.5 Relationship to other products

Table A.28 - Relationship of SV-7 to Other Products

View	Relationship Details
SV-1	The performance parameters of SV-7 apply to systems, subsystems, and system hardware/software items of SV -1.
SV-2	Performance parameters of SV -7 that deal with communications systems, communications links, and communications networks should map to the corresponding elements in SV-2.
SV-6	Performance parameters in SV-7 that deal with interface performance and system data exchange capacity requirements should trace to system data exchanges in SV-6.
SV-8	If required performance ranges defined in SV-7 are associated with an overall system evolution or migration plan defined in SV-8, then the time period should correspond to the milestones in SV-8.
SV-9	If the future performance expectations (goals) defined in SV-7 are based on expected technology improvements, then the performance parameters and their time periods in SV-7 should be coordinated with the timed technology forecasts defined in SV-9.

A.3.8 Systems Evolution Description (SV-8)

A.3.8.1 Product Definition

The Systems Evolution Description typically reviews bundles of Systems, System Hardware, and System Software in a System Group as it evolves over time.

A.3.8.2 UPDM Notation

No UML diagram supports the SV-8. UPDM contains a number of elements to support key concepts from the SV-8, such as milestones, in the model. Using the standard UPDM link mechanism, information from the model can be sent out to a scheduling tool. Alternatively, the architect may link to the output from an external scheduling tool.

A.3.8.3 Product Detailed Description

SV-8 describes plans for modernizing system functions over time. Such efforts typically involve the characteristics of evolution (spreading in scope while increasing functionality and flexibility) or migration (incrementally creating a more streamlined, efficient, smaller, and cheaper suite) and will often combine the two thrusts. This product builds on other products and analyses in that planned capabilities and information requirements that relate to performance parameters (of SV-7) and technology forecasts (of SV-9) are accommodated in this product. If the architecture describes a communications infrastructure, then a planned evolution or migration of communications systems, communication links, and their associated standards can also be described in this product.

A.3.8.4 Semantics

SV-8, when linked together with other evolution products such as SV-9 and TV-2, provides a clear definition of anticipated evolution. The product can be used as part of an architecture evolution project plan or transition plan. The SV-8 is a plan/schedule for the evolution of a System Group. Key milestones are associated with incremental implementations of changes to the structure and/or behavior of the key System Groups.

Table A.29 - SV-8 Elements

UPDM Element	Significance to the View
Forecast	A description of the actual or predicted status of a System at a Project Milestone.
Milestone	An event in a project with associated goals and assessment criteria.
Project	A plan by an organizational unit to procure systems related to operations scheduled for a finite period of time.
System	The resource that controls the delivery of System Functipons.
SystemGroup	A collection of System elements, including hardware and supporting software, that forms a unit for tracking and forecast purposes.
SystemGroupMember	Specifies that the element is within the referenced SystemGroup.
SystemHardware	The computational platform for the Systems that must be tracked for future evolution.
SystemSoftware	Supporting software that must be tracked for the evolution of the SystemGroup.
TimePeriod	A period of time, defined by start and end dates.

A.3.8.5 Relationship to other products

Table A.30 - Relationship of SV-8 to Other Products

View	Relationship Details
SV-1	The systems, subsystems, and system hardware/software items of SV -8 should match the corresponding elements in SV-1.
SV-2	Communication network items can be in System Groups.
SV-4	If SV-8 identifies system functions to be implemented in each phase of a system development, they should match the system functions in SV-4.
SV-7	If required performance ranges defined in SV-7 are associated with an overall system evolution or migration plan defined in SV-8, then the time periods in SV-7 should correspond to the milestones in SV-8.
SV-9	The time periods associated with timelines and milestones in SV-8 should be coordinated with time periods associated with timed technology forecasts in SV-9.

A.3.9 Systems Technology Forecasts (SV-9)

A.3.9.1 Product Definition

The Systems Technology Forecast defines the underlying current and expected supporting technologies for the enterprise. 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.

A.3.9.2 UPDM Notation

A table and matrix with information exported from model elements.

A.3.9.3 Product Detailed Description

SV-9 provides a detailed description of emerging technologies related to supporting hardware and software products. It contains predictions about the availability of emerging technological capabilities and about industry trends in specific time periods. The specific time periods selected (e.g., 6-month, 12-month, 18- month intervals) and the technologies being tracked should be coordinated with architecture transition plans (see SV- 8). That is, insertion of new technological capabilities and upgrading of existing systems may depend on or be driven by the availability of new technology. The forecast includes potential technology impacts on current architectures and thus influences the development of transition and target architectures. The forecast should be focused on technology areas that are related to the purpose for which a given architecture is being described and should identify issues that will affect the architecture. If standards are an integral part of the technologies important to the evolution of a given architecture, then link the SV-9 to the Technical Standards Forecast (TV-2).

A.3.9.4 Semantics

The purpose of the SV-9 is to identify emerging technology likely to impact the structure or behavior of the system in its enterprise context. The focus should be on the supporting technologies that may most affect the capabilities of the architecture or its systems.

Table A.31 - SV-9 Elements

UPDM Element	Significance to the View
Forecast	A statement about the future state of one or more types of system or standard.
SystemHardware	The computational platform for the Systems.
SystemSoftware	Supporting software that enables Systems in the architecture.
System	While the SV-9 focuses on supporting systems for the components in the model, some architectural descriptions will not make that distinction. In cases where the operating systems and system hardware represent an integral part of the architecture, the System will be a part of the SV-9 analysis.
TimePeriod	A period of time, defined by start and end dates--sometimes termed an “Epoch” in the MOD. Time periods may overlap.

A.3.9.5 Relationship to other products

Table A.32 - Relationship of SV-9 to Other Products

View	Relationship Details
SV-1	SystemHardware and SystemSoftware could appear on the SV-1, although they do not have to. A user may want to query which Systems are influenced by the hardware or operating system changes.
SV-2	System connectivity in this diagram might rely on supporting systems outlined in the SV-9.
SV-8	Incremental changes in technology are correlated with the milestones in the SV-8 to provide an overview of important technical changes for the Systems.

A.3.10 Systems Rules, State Transitions and Event-Trace Description (SV-10)

A.3.10.1 Product Definition

Detailed system behavior and interactions as well as the state of any System features at particular times define a number of constraints for the core elements of the System View. 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 under certain circumstances. The SV-10 provides such insight.

The interactions in the SV-10 can be placed inside an OperationalActivityRealization or OperationalCapabilityRealization, providing traceability back to the OperationalView and making it easy to derive the information needed for the SV-5. The interactions could also be inside of an ActivityRealization to show a more detailed realization of SystemFunctions. See the section in the SV-4 to show how to use the ActivityRealization.

The Systems Rule Model explains system rules. The Systems State Transition Description describes a System state as it changes in response to various events. The Systems Event Trace provides the sequences of actions and details of exchanges and flows between Systems.

A.3.10.2 UPDM Notation

A.3.10.2.1Systems Rules Model

Text of rules exported from model. There is no UML diagram for the rules section of the SV-10. Rules are statements that define or constrain some aspect of the enterprise. In contrast to the Operational Rules Model, the System focuses on constraints imposed by System implementation or other aspects of System function. At a lower level of detail, it focuses on some aspects of systems design or implementation. Thus, as the operational rules can be associated with the activity models on the operational side the systems rules in SV-10 can also be associated with any of the dynamic diagrams in the Systems View.

A.3.10.2.2Systems State Transition Description

The State Transition Description works best as a UML StateMachine Diagram. The diagram basically represents how any element in the Systems View can respond to sets of events and conditions. Each transition specifies an event and an action. UML-based methodologies that use action languages in order to drive the creation of code in a system will use these diagrams to express their actions inside the DoDAF or MODAF structure. The SV StateMachine can also link to the Activity Diagrams in the SV-4 using the same mechanisms outlined in the OV-6.

The explicit state of Systems in response to external and internal events is not fully expressed in SV-4. SV-10 can be used to describe the explicit state of elements interacting with the system functions. StateMachines can be unambiguously converted to structured textual rules that specify timing aspects of systems events and the responses to these events, with no loss of meaning.

SV-10 could have a diagram for each System, subsystem, or SystemFunction whose behavior is event-driven and sufficiently complex to warrant state-based analysis. Use events, actions, and state transitions to show the impact of the sent. The diagram below shows state transitions for the JFACC Planning System. Because the diagram does not show the stereotype, this is indicated in parentheses for clarification.

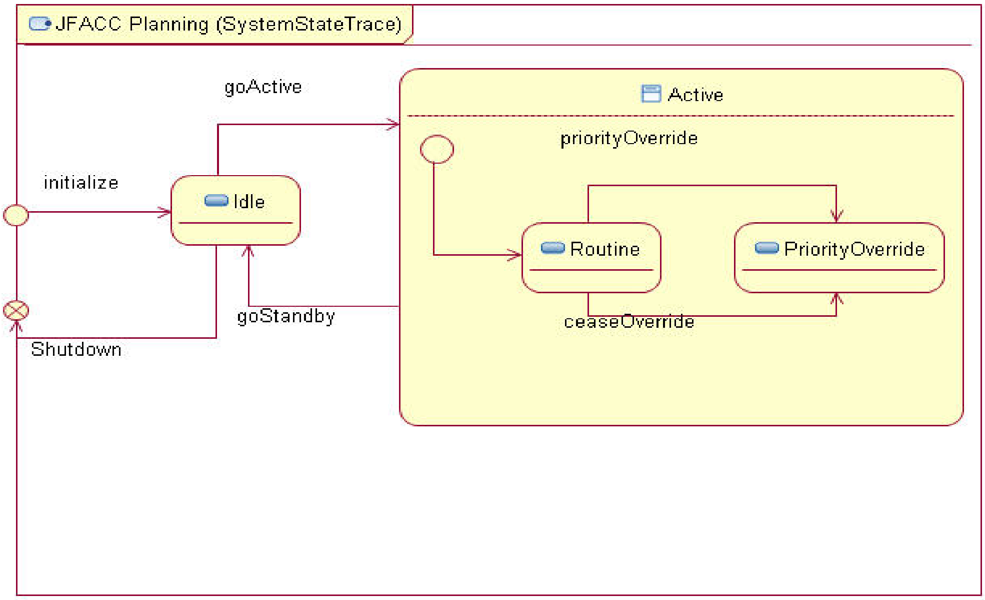


Figure A.20 - SV-10 State Transition Diagram

A.3.10.2.3Systems Event Trace

The Sequence Diagram allows the tracing of actions in a scenario or critical sequence of events. With time proceeding from the top of the diagram to the bottom, a specific diagram lays out the sequence of system data exchanges that occur between systems, system functions, or human roles for a given scenario. Different scenarios should be depicted by separate diagrams. The System Event Trace can be used by itself or in conjunction with the State Transition Description to explore the dynamic behavior of system processes or system function threads

The diagram below shows a <<SystemEventTrace>> that elaborates SystemTasks that call a SystemFunction. Note that any SystemFunction referenced within a Sequence Diagram that is inside an OperationalActivityRealization could provide the link to the OperationalActivity, even if the SystemTask resides inside a fragment on the event trace.

The SystemTask might well invoke a SystemFunction which provides another mechanism to provide a link between OperationalActivity and SystemFunction.

The distinction between the SystemsEventTrace and the OperationalEventTrace is that in this case all the of the messages that invoke tasks call on tasks owned by Systems. On the Operational side, the OperationalTask invokes OperationalActivities, not SystemFunctions.

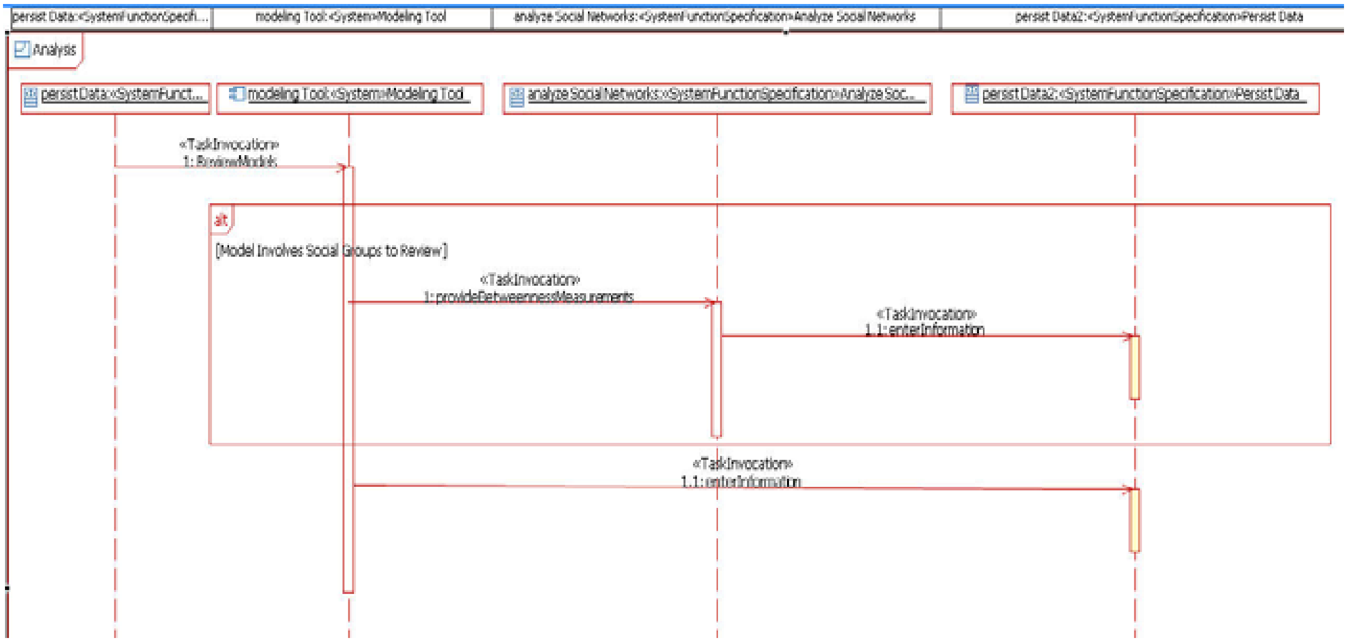

persist Data:«SystemFunctionSpecifi...
modeling Tool:«System»Modeling Tool
analyze Social Networks:«SystemFunctionSpecification»Analyze Social Networks
persist Data2:«SystemFunctionSpecification»Persist Data
Analysis
persist Data:«SystemFunct...
modeling Tool:«System»Modeling Tool
analyze Social Networks:«SystemFunctionSpecification»Analyze Soc...
persist Data2:«SystemFunctionSpecification»Persist Data
«TaskInvocation»
1: ReviewModels
alt
[Model Involves Social Groups to Review]
«TaskInvocation»
1: provideBetweennessMeasurements
«TaskInvocation»
1.1: enterInformation
«TaskInvocation»
1.1: enterInformation

Figure A.21 - Example of a <<SystemEventTrace>> Elaboration of dynamic invocation of SystemTasks

A.3.10.3 Semantics

The SV-10 captures the key behaviors and constraints related to dynamic processing between Systems.

When the behavior of one or more key system elements is event-driven, modeling with State Machines can be especially useful in understanding that behavior.

SV-10 products are also valuable for providing a collaboration that shows the link between System Functions and Operational Activities. The view defines a sequence of functions and system data interfaces, while showing that each participating System, System Function, or human role has the necessary elements it needs, at the right time, in order to perform its assigned functionality.

Table A.33 - SV-10a Elements

UPDM Element	Significance to the View
ActivityRealization	Maps an activity to the collections that provide the elements needed to realize that activity. This can work with operational activities or system functions.
DataElement	The core data used in machine communication between Systems.
DataExchange	A flow of information between two Systems. A DataExchange will contain a set of DataElements. The Systems Event Trace typically details the sequences of these exchanges.
OperationalActivityRealization	A set of interactions of System elements that implement or realize the OperationalActivity. These interactions are often delivered as part of the SV-10.

Table A.33 - SV-10a Elements

OperationalCapabilityRealization	A set of interactions that realize the OperationalCapability. The OperationalCapability is often composed of Operational interactions designed to achieve a capability for mission success. The OperationalCapabilityRealization links to the SV-10 via the OperationalActivities it contains.
Rule	A general policy that can be applied to any element in UPDM.
System	The StateMachines and System Event Traces work with the behavior and structure of the Systems in the architecture.
SystemEventTrace	Depicts the events in the interactions between SystemTasks and Systems. This references SystemFunctions through any link between SystemTask and SystemFunction.
SystemFunction	The core activities of the Systems that deliver the functionality used to carry out the mission activities defined in the Operational View.
SystemInterface	Indicates that two systems require a DataExchange to function. Any DataExchange possible between two systems will indicate a SystemInterface between those Systems.
SystemStateTrace	A StateMachine showing the impact of dynamic events on Systems.
SystemTask	An operation owned by a System. This task can invoke a SystemFunction or it can implement a ServiceSpecification. DataExchanges between Systems that create the requirement for a SystemInterface are usually between SystemTasks on the two Systems.

A.3.10.4 Relationship to other products

Table A.34 - Relationship of SV-10a to Other Products

View	Relationship Details
OV-5	If inside an OperationalActivityRealization or OperationalCapabilityRealization then the interaction defined creates links between the operational activities in the OV-5 and the system details in the SV-10.
SV-4	If system rules in SV -10a deal with system behavior, they should reference system functions in SV-4.
SV-5	The SV-10 can provide the mapping needed to show connections between the Operational and System views.

A.3.11 Physical Schema (SV-11)

A.3.11.1 Product Definition

The Physical Data Model describes the structure of an architecture domain’s system data types and the structural business process rules that govern the system data. It provides a definition of architecture domain data types, their attributes or characteristics, and their interrelationships.

A.3.11.2 UPDM Notation

SV-11 may be modeled in UML using a class diagram. Classes that appear in an SV-11 class diagram correlate to SV-6 DataElements.

A.3.11.3 Product Detailed Description

SV-11 defines the architecture domain’s system data types (or entities) and the relationships among the system data types.

A.3.11.4 Semantics

SV-11 contains the domain’s system data types or entity definitions and can be used by other organizations to determine data compatibility. Often, different organizations may use the same entity name to mean very different kinds of system data with different internal structure.

Table A.35 - SV-11 Element

UPDM Element	Significance to the View
DataElement	The data element indicates the type and structure of the information transferred between Systems on DataFlows.

A.3.11.5 Relationship to other products

Table A.36 RElationship of SV-11 to Other Products

View	Relationship Details
SV-4	The flows and data elements in the SV-4 activity model can be constrainted by the SV-10..
SV-6	A DataElement in SV-6 should be constructed of entities from the SV-11.
SV-10	The rules in the model may reference the elements of SV-11 to constrain their structure and validity.

A.4 Technical Standards View

A.4.1 Technical Standards Profile (TV-1)

A.4.1.1 Product Definition

The (TV-1) collates the various Systems and Standards that implement and constrain the choices that can be made in the design and implementation of an Architectural Framework.

A.4.1.2 UPDM Notation

Typically a document that includes all the technical standards applicable to the System.

A.4.1.3 Product Detailed Description

The TV-1 deliverable will usually be a document. The model can contain information about the application of a technical standard to every relevant element in the model.

A.4.1.4 Semantics

The main purpose of the TV-1 is to show compliance of the architecture with existing external technical standards or external reference models.

Table A.37 - TV-1 Elements

UPDM Element	Significance to the View
ConformingElement	Abstract element. Any stereotype that extends ConformingElement can link elements to a specific Standard in the context of a profile. Most of the core system elements can demonstrate compliance with a standard.
ReferenceModel	An external model that provides standards for the architecture. In the US Federal Enterprise Architecture, these are called reference models. The FEA (Federal Enterprise Architecture) consists of a number of reference models that apply to any architecture in the US Federal Government. A project may have any number of such models. Increasingly, these models have an expression in UML, making it possible for automated assessment of compliance. The reference models will link to the standards in the model.
Standard	A standard represents an external rule or constraint applied to elements in the System indicating compliance with rules regarding the type of deployments that can be accepted. Standards will typically stem from an external source or reference model.
TechnicalStandardsProfile	A collection of standards that represents the set of relevant technical standards for the Systems in the architectural description.

A.4.1.5 Relationship to other products

Table A.38 - Relationship of TV-1 to Other Products

View	Relationship Details
TV-2	The projection of standards into the future in this view shows the possible evolution of these standards.
SV-*	Any ConformingElement in the SystemView may be constrained by the technical standards for the organization. This includes nearly all the key elements for the System View.

A.4.2 Technology Standard Forecast (TV-2)

A.4.2.1 Product Definition

The Technical Standards Forecast portrays expected changes in technology-related standards and conventions, which are documented in the TV-1 Product. The forecast for evolutionary changes in the standards need to be correlated against the time periods mentioned in the SV-8 and SV-9 Products.

A.4.2.2 UPDM Notation

A document overview of the standards that apply to the Systems of the enterprise.

A.4.2.3 Product Detailed Description

A structure diagram can show the time periods associated with the new standards.

A.4.2.4 Semantics

The TV-2 shows the possible planned technology standards. This allows those planning new systems to adopt new technologies with a reasonable expectation of compliance with future standards.

A technical standard is typically imposed from outside the architecture. The technical standards are often rolled into a technical standards profile.

Table A.39 - TV-2 Element

UPDM Element	Significance to the View
ConformingElement	Abstract element. Any stereotype that extends ConformingElement can link profile elements to a specific Standard in the context of a profile. Most of the core system element can demonstrate compliance with a standard.
Forecast	A statement about the future condition of a standard that applies to a System. A forecast will usually have an associated date. For example, the next version of the UML tool will be accredited for classified use in six months.
ForecastStandardsProfile	An association between a forecast and the StandardsProfile.
ReferenceModel	An external model that provides standards for the architecture. In the US Federal Enterprise Architecture, these are called reference models. The FEA (Federal Enterprise Architecture) consists of a number of reference models that apply to any architecture in the US Federal Government. A project may have any number of such models. Increasingly, these models have an expression in UML, making it possible for automated assessment of compliance. The reference models will link to the standards in the model.
Standard	A standard represents an external rule or constraint applied to elements in the System indicating compliance with rules regarding the type of deployments that can be accepted. Standards will typically stem from an external source or reference model.
TechnicalStandardsProfile	A collection of standards that represents the set of relevant technical standards for the Systems in the architectural description.

A.4.2.5 Relationship to other products

Table A.40 - Relationship of TV-2 to Other Products

View	Relationship Details
TV-1	The current application of the standards projected in the TV-2.

Table A.40 - Relationship of TV-2 to Other Products

SV-*	Any ConformingElement in the SystemView may be constrained by the technical standards for the organization. This includes nearly all the key elements for the System View.

A.5 Strategic View

The strategic view stems is a MODAF view and is not a part of the DoDAF. Many of the features in the Strategic View are important to any architecture in the domain and a good model will include many elements similar to those found in the Strategic View. In the United States federal government, many of the elements in these views relate to the Federal Enterprise Architecture reference models. Also, the capability sections are similar to the US Department of Defense -- Joint Capabilities Integration and Development System (JCIDS). These strategic view deliverables do not explicitly integrate with either of these, but the stereotypes in this section could be used to generate additional information about the capabilities covered in the model. UPDM provides both MODAFMODAF and DoDAF structures, allowing for the model to product both types of views.

For detailed examples of the MODAF views please see the MODAF documentation.

A.5.1 Capability Vision (StV-1)

A.5.1.1 Product Definition

The Capability Vision (StV-1) defines the strategic context for a group of capabilities. The views should target the general user audience with definitions referenced in the glossary.

A.5.1.2 UPDM Notation

StV-1 Views are usually textual descriptions of the capabilities being analyzed or planned.

A.5.1.3 Product Detailed Description

The entities described in the text for this map to a number of core concepts in the model that are found in other views. This view should describe the elements the show up the related products.

A.5.1.4 Semantics

A strategic capability enables capability based analysis. Typically, the strategic level will reflect doctrine and high level capabilities while the operational views will focus on mission-oriented capabilities.

Table A.41 - StV-1 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise’s abilities that allow them to execute missions.
Enterprise	The organizational unit for the architectural description.
Goal	The Enterprise will have a number of goals that represent the required objectives for the Enterprise.

Table A.41 - StV-1 Elements

OperationalCapability (Enduring Task)	The set of tasks and activities that represent a continuing active that is part of the core mission of the enterprise.
Vision	A high level description of how the capabilities will help the Enterprise achieve goals.

A.5.1.5 Relationship to other products

Table A.42 - Relationship of StV-1 to Other Products

View	Relationship Details
OV-5	Operational capabilities in the OV-5 can link to this strategic view for the OperationalCapability.
StV-*	The capability vision applies throughout the strategic view.

A.5.2 Capability Taxonomy (StV-2)

A.5.2.1 Product Definition

The Capability Taxonomy (StV-2) view provides a structured list of capabilities and subcapabilities (known as capability functions) that are required within a capability area during a certain period of time. This taxonomy provides a more structured approach to the vision outlined in the StV-1.

A.5.2.2 UPDM Notation

Composite structure diagrams support the capability vision. Please see the MODAF documentation for an example. UPDM uses the composite/component pattern as explained in the All View for all elements that have multiple levels of decomposition, like Capability.

A.5.2.3 Product Detailed Description

A structure diagram that breaks down capabilities into sub-functions. Additional attributes show the desired performance characteristics for the Capability and any StrategicCapabilityFunctions, or sub-functions.

A.5.2.4 Semantics

The StV-2 View is used to support a capability audit process, providing a structured set of capabilities. In addition it can be used as a source document for the development of high level use cases and key user requirements.

The taxonomy can also include metrics and desired performance targets for the capabilities.

Table A.43 - StV-2 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise’s abilities that allow them to execute missions.

Table A.43 - StV-2 Elements

CapabilityRequirement	Contains criteria for the performance of the capability, which might include detailed metrics or the elucidation of the impact expected from the capability.
CapabilityFunction (Not a UPDM Element, but used in MODAF)	UPDM does not have a distinct element showing that a capability resides inside of another capability, as UPDM and UML provide such information through normal usage. UPDM includes the notion of a composite element that shows elements such as Capability that implement the composite component pattern.

A.5.2.5 Relationship to other products

Table A.44 - Relationship of StV-2 to Other Products

View	Relationship Details
StV-1	The StV-2 works closely with the StV-1 to provide the metrics and the decomposition of the high level capability vision.
StV-3	The phasing of capabilities is found in the StV-3.

A.5.3 Capability Phasing (StV-3)

A.5.3.1 Product Definition

The Capability Phasing (StV-3) View provides a representation of the available military capability at different points in time or during specific periods of time. This is a time phased view of capabilities and anticipated capabilities.

A.5.3.2 UPDM Notation

Text document or table that shows the evolution of a capability over time. No UML Diagram is available for this View.

A.5.3.3 Product Detailed Description

The capability functions identified in this view are structured according to the Capabilities in the StV-2. The View is created by analyzing programmatic project data to determine when systems and any supporting organizational resources can provide elements of the capabilities to be delivered or upgraded. The Systems can be found in the StV-5 links.

A.5.3.4 Semantics

This view shows how capabilities will be evolving over time and will support the assessment of priorities for the enterprise. It is similar to the SV-8 in that it shows evolution over time, but linking that evolution into the enterprise capabilities and analyzing at a level that would cross a number of programs throughout the enterprise.

Table A.45 - StV-3 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise's abilities that allow them to execute missions.
CapabilityConfiguration	The combination of organizational resources, equipment, training, competencies, and mission processes needed to deliver a capability.
CapabilityConfigurationCapabilities	Defines an association between a CapabilityConfiguration and organizing Capability.
CapabilityFunction (Not a UPDM Element, but used in MODAF)	UPDM does not have a distinct element showing that a capability resides inside of another capability, as UML provides such information through normal usage. UPDM includes the notion of a composite element that shows elements such as Capability that implement the composite component pattern.
CapabilityRequirement	Contains criteria for the performance of the capability, which might include detailed metrics or the elucidation of the impact expected from the capability.
CapabilityRequirementCapability	Asserts that a Capability Requirement is related to an operational capability.
Milestone	An event in a project with associated goals and assessment criteria.
Project	An organizational unit to secure Systems and SystemFunctions that deliver a strategic Capability. A project references timeperiods as well as milestoners to define key project delivery dates.
Resource	Anything that supplies functionality, information, or materiel. On OrganizationalResource is a type of Resource specifically scoped to operational concerns.
ResourceCapability	An association between Resource and a Capability.
ResourceCapabilityConfiguration	An association between a CapabilityConfiguration and a Resource.
System	The collections of technology and processes that deliver the SystemFunctions. The project delivers these Systems.
SystemFunction	The functionality that a System can implement. The StV-3 will reference evolution of SystemFunction in term of expected industry performance. The projects deliver these SystemFunctions.

A.5.3.5 Relationship to other products

Table A.46 - Relationship of StV-3 to Other Products

View	Relationship Details
AcV-2	Data may be provided in part by an SoS Acquisition Programs (AcV-2).
OV-5	The CapabilityRequirements and delivery of capabilities relate to the operational capabilities outlined in the OV-5
StV-2	The approach must follow the capability taxonomy in the StV-2.
StV-5	Where a capability cannot be equated on a one-to-one mapping with a particular System, further analysis can be assisted using the information provided in a Capability to Systems Deployment Mapping (StV-5) Product.

A.5.4 Capability Clusters (StV-4)

A.5.4.1 Product Definition

Capability clusters describe logical groupings of capabilities and show dependencies between different capabilities.

A.5.4.2 UPDM Notation

Composite Structure Diagram or a matrix showing functional dependencies. Use UML dependencies to indicate a dependency between different capabilities. See the MODAF documentation for an example.

A.5.4.3 Product Detailed Description

The capabilities reviewed here show general dependencies between capabilities. Capabilities can be nested inside of other capabilities. The items in this description do not represent Systems, but capabilities.

A.5.4.4 Semantics

Capability clusters enables the analysis of dependencies between different capabilities.

Table A.47 - StV-4 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise’s abilities that allow them to execute missions.
CapabilityFunction (Not a UPDM Element, but used in MODAF)	UPDM does not have a distinct element showing that a capability resides inside of another capability, as UML provides such information through normal usage. UPDM includes the notion of a composite element that shows elements such as Capability that implement the composite component pattern.

A.5.4.5 Relationship to other products

The capability clusters in this product relate to any Capabilities in this view.

A.5.5 Capability to Systems Deployment Mapping (StV-5)

A.5.5.1 Product Definition

The Capability to Systems Deployment Mapping (StV-5) shows deployment of Systems in types of organizations (e.g. echelons), and the connections between those Systems to satisfy the needed capability for a particular period of time (or Epoch).

A.5.5.2 UPDM Notation

A matrix showing issues with systems deployed for capability. No UML diagram for this view.

A.5.5.3 Product Detailed Description

The systems identified in this view will have their deployment status derived from the type of UPDM element as well as the relationship of that system to deployed system hardware. In other words, if a system is part of the "to be" architecture, then that system will not be actually deployed.

A.5.5.4 Semantics

This view shows gaps in systems deployment in terms of the strategic capabilities for the entire enterprise.

Table A.48 - StV-5 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise's abilities that allow them to execute missions.
CapabilityConfiguration	The combination of organizational resources, equipment, training, competencies, and mission processes needed to deliver a capability.
CapabilityConfigurationCapabilities	Defines an association between a CapabilityConfiguration and organizing Capability.
CapabilityFunction (Not a UPDM Element, but used in MODAF)	UPDM does not have a distinct element showing that a capability resides inside of another capability, as UML provides such information through normal usage. UPDM includes the notion of a composite element that shows elements such as Capability that implement the composite component pattern.
CapabilityOperationalCapability	Indicates a relationship between a Capability and the Operational Capabilities that support it.
CapabilityRequirement	Contains criteria for the performance of the capability, which might include detailed metrics or the elucidation of the impact expected from the capability.

Table A.48 - StV-5 Elements

CapabilityRequirementCapability	Asserts that a Capability Requirement is related to an operational capability.
Milestone	An event in a project with associated goals and assessment criteria.
OperationalCapability	A use case that specifies the requirements for a Capability.
OrganizationalResourceCapability Configuration	Identifies the relationship between a capability configurations and the Organizaitons that support that configuration.
Project	An organizational unit to secure Systems and SystemFunctions that deliver a strategic Capability. A project references timeperiods as well as milestoners to define key project delivery dates.
Resource	Anything that supplies functionality, information, or materiel. On OrganizationalResource is a type of Resource specifically scoped to operational concerns.
ResourceCapability	An association between Resource and a Capability.
ResourceCapabilityConfiguration	An association between a CapabilityConfiguration and a Resource.

A.5.5.5 Relationship to other products

Table A.49 - Relationship of StV-5 to Other Products

View	Relationship Details
AcV-2	Data may be provided in part by an SoS Acquisition Programs (AcV-2).
OV-2	The OV-2 can provide detail on the OperationalNode activities that deliver capabilities.
OV-5	The CapabilityRequirements and delivery of capabilities relate to the operational capabilities outlined in the OV-5
StV-2	The approach must follow the capability taxonomy in the StV-2.
StV-5	Where a capability cannot be equated on a one-to-one mapping with a particular System, further analysis can be assisted using the information provided in a Capability to Systems Deployment Mapping (StV-5) Product.
SV-8	A StrategicCapabilityConfiguration with a relationship to a milestone of SystemEvolutionMilestone will show the anticipated configuration of capabilities at that point.

A.5.6 Capability Function to Operational Activity Mapping (StV-6)

A.5.6.1 Product Definition

The Capability Function to Operational Activity Mapping maps the functions outlined for the capabilities to the high level operational capabilities that organize the operational activities in the operational view.

A.5.6.2 UPDM Notation

No formal diagram. The model will have information to produce a matrix showing the relationship between the Strategic Capabilities and the enduring tasks (OperationalActivities) that organize the actual missions supported by the capabilities.

A.5.6.3 Product Detailed Description

The model will include relationships between the capabilities and the operational capabilities, or enduring tasks. From the relationship to the operational capabilities, a user could extend to a number of additional queries in the operational view for custom views. The link between the capability and the operational capability allow the link between the strategic view and the operational view.

A.5.6.4 Semantics

This view shows the operational capabilities and activities that rely on the organization's capability functions.

Table A.50 - StV-6 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise's abilities that allow them to execute missions.
CapabilityFunction (Not a UPDM Element, but used in MODAF)	UPDM does not have a distinct element showing that a capability resides inside of another capability, as UML provides such information through normal usage. UPDM includes the notion of a composite element that shows elements such as Capability that implement the composite component pattern.
CapabilityOperationalCapability	Identifies a relationship between a Capability and the supporting Operational Capabilities.
OperationalActivity	The processes used to execute a key mission objective. A set of operational tasks that are supported by SystemFunctions.
OperationalCapability (Enduring Task)	A set of processes essential to the core mission of the Enterprise. These are the core Use Cases for a Capability as carried out in an operational context.
OperationalCapabilityRealization	The combination of activities and tasks at OperationalNodes that allow for the execution of an OperationalCapability.
PerformanceEnabler (Not a UPDM element, but derived from the OperationalCapabilityRealization and used in MODAF)	The MODAF PerformanceEnabler provides the connection between a Capability and an OperationalActivity. In UPDM, this is achieved using the OperataionalCapabilityRealization for each OperationalCapability that is linked to the Capability.

Relationship to other products

Table A.51 - Relationship of StV-6 to Other Products

View	Relationship Details
StV-2	This follows the capability taxonomy outlined in this view.
OV-2	The Operational Activities are on the Nodes in the OV-2.
OV-5	The Operational Activities and the Operational Capabilities are defined in the OV-5.

A.6 Acquisition View

A.6.1 SoS Acquisition Clusters (AcV-1)

A.6.1.1 Product Definition

This view describes clusters of systems that fit together to form acquisition clusters, a measure used to assess procurement in MODAF.

A.6.1.2 UPDM Notation

This will typically link to an external artifact using information in the model. This view uses a Composite Structure Diagram to construct the model relationships with an export to an external tool using the standard UPDM export mechanisms.

A.6.1.3 Product Detailed Description

The acquisition clusters do not have to be in a diagram in UML, however the clusters should be organized and include all of the acquisition projects delivering systems or SoS within the acquisition program under consideration   Other systems and systems of systems which may have a bearing on the Architecture should also be included. The view will indicate how the systems will be best integrated into acquisition clusters. The view should also show how the acquisition clusters can be composed of other clusters in a hierarchy.

A.6.1.4 Semantics

This helps guide the acquisition process so that the systems develop in an effective manner. Some leaders may want to see a number of interdependent projects while others will want to be able to move functionality into a modular environment. The product also emphasizes the organizations which have projects to acquire new capabilities.

Table A.52 - AcV-1 Elements

UPDM Element	Significance to the View
DeliveredCapability	Relates a Capability to a Project that delivers that Capability.
OrganizationalRelationship	A set of attributes on the association between two organizations.
OrganizationalResource	OrganizationalResources own Projects and are used to configure Capabilities.

Table A.52 - AcV-1 Elements

Project	An organizational unit to secure Systems and SystemFunctions that deliver a StrategicCapability. A project references TimePeriods as well as Milestones to define key project delivery dates.
ProjectType	A category of a Project.
System	This view emphasizes the Systems that depend on the listed Projects.

A.6.1.5 Relationship to other products

Table A.53 - Relationship of AcV-1 to Other Products

View	Relationship Details
OV-4	Organizations are the main sponsors for projects that will deliver new capabilities.
StV-2	A project can relate to a Capability from the StrategicView.
SV-8	The projects outlined here can be linked to any SystemGroups in the Systems Evolution Description.
SV-9	The technology forecasts can have an impact on the projects in this view.
TV-2	The forecast on the relevant standards for Systems might well have an impact on projects.

A.6.2 SoS Acquisition Program (AcV-2)

Product Definition

This provides an overview of planned projects along a timeline. The product intends to show many aspects of a project, including maturity, status, planned phases, and cross project dependencies.

A.6.2.1 UPDM Notation

No UML diagram. UML elements can export to a tool that can provide the timeline chart with pie charts or other icon showing the maturity of the projects. See the MODAF documentation for the description of the view.

A.6.2.2 Product Purpose

This view supports acquisition planning that aligns with the System of Systems architecture as expressed across the organization in the System Views.

A.6.2.3 Product Detailed Description

Attributes on the project and capability entity will allow the production of the key information. The visualization will be in another tool. The MODAF DLOD typically includes the following eight elements: training, equipment, logistics, infrastructure, organization, doctrine/concepts, information, and personnel. The product should indicate for each milestone and each MODAF DLOD whether there are any areas of concerns and whether the projects have plans to address those areas of concern. The pie charts or other display icons should use color to communicate the status, with green indicating no areas of concern and red indicating an urgent issue that requires changes in project plan. Yellow indicates there are outstanding areas of concern, but there are plans in place to address these concerns. White cells indicate that the DLOD status is not known while black cells indicate that the DLOD status is not required.

A.6.2.4 Semantics

This view supports acquisition planning that aligns with the System of Systems architecture. As acquisition planning involves a number of features that indicate "maturity," this view encourages the representation of a project at a specific milestone with an icon. As the MODAF technical handbook states, "The colored icon can be a segmented circular pie chart, a regular polyhedron or any appropriate graphic providing that the graphic is explained and covers all MoD DLODs." (MODAF, 166)

Table A.54 - AcV-2 Elements

UPDM Element	Significance to the View
Capability	A high level organizing description of the Enterprise's abilities that allow them to execute missions.
CapabilityConfiguration	The combination of organizational resources, equipment, training, competencies, and mission processes needed to deliver a capability.
CapabilityRequirement	Contains criteria for the performance of the capability, which might include detailed metrics or the elucidation of the impact expected from the capability.
DLODElements	Defence Logistics Operations material with the attributes to show the status of a particular capability in terms of whether it is available. These are the attributes used in the pie charts to illustrate project maturity at lifelines: training, equipment, logistics, infrastructure, organization, doctrine/concepts, information, personnel.
Milestone	An event in a project with associated goals and assessment criteria.
Project	An organizational unit to secure Systems and SystemFunctions that deliver a StrategicCapability. A project references TimePeriods as well as Milestones to define key project delivery dates.
ProjectType	
TimePeriod	A period of time, defined by start and end dates--sometimes termed an "Epoch" in the MOD. Time periods may overlap. In this view, both Milestones and Projects will have relationships with TimePeriod.

A.6.2.5 Relationship to other products

Table A.55 - Relationship of AcV-2 to Other Products

View	Relationship Details
SV-8	The Projects should relate to the Systems Evolution Description

A.7 Using Level-0 and Level-1 Compliant Profiles

UPDM defines two compliance levels. Level 0 extends only those metaclasses defined in UML4SYSML. This assures that the UPDM stereotypes will be compatible with those defined in SysML.

Level 1 defines a specialization of SysML stereotypes on ten of the UPDM stereotypes (See Table 6.2). To accomplish this, the SysML profile is imported into the Level 0 UPDM profile, resulting in the Level 1 Profile.

Access to the UPDM stereotypes is obtained by applying either the Level 0 or Level 1 profile. Although the Level 1 profile includes the imported SysML profile, it is only applicable at design time of the UPDM profile to specify the specializations; not at runtime when the profile is applied to a model. If it is desired to access the UPDM stereotypes and the SysML stereotypes defined in the SysML profile, the SysML profile must also be applied to the model.

There are 4 scenarios of applying UPMD:

1. Apply UPDM Level 0 by itself
2. Apply UPDM Level 1 by itself
3. Apply UPDM Level 0 and also apply SysML
4. Apply UPDM Level 1 and also apply SysML

A.7.1 Apply UPDM Level 0 by Itself

There are many projects that address only architecture issues, and practitioners model systems and produce DoDAF and MODAF artifacts using only UML. All of UML will be available to the modeler. For these users, a tool that applies only UPDM Level 0 is sufficient.

For example, a modeler could create a Class and stereotype it with the UPDM stereotype

<<System>>: Class <<System>> MySystem

This gives MySystem the UPDM::System stereotype features .

A.7.2 Apply UPDM Level-1 by Itself

In many projects, the architects will model the initial phases of a project and produce DoDAF and MODAF artifacts, but in the MDA style of development, the follow-on phases will be modeled by System Engineers who will also need to deliver more detailed DoDAF and MODAF artifacts. When applying **only** the UPDM Level 1, all of UML will be available to the modeler but none of the SysML stereotypes will be available. However, these architect/modelers will not utilize SysML,. By using Level 1 of UPDM, the model will already contain elements that are primed for use in a SysML environment.

For example, the modeler applies the Level 1 stereotype

<<System>>

to a class:

Class <<System>> MySystem

This gives MySystem the UPDM::System stereotype features and the SysML::Block stereotype features because in Level-1, UPDM::System also specialize SysML::Block.

A.7.3 Apply UPDM Level 0 and SysML Profiles

If the model is initially created using only UPDM Level 0, and at some point the model will be picked up by System Engineers who will apply SysML to the model; then the modelers will want to apply SysML stereotypes to the UPDM Level 0 artifacts. The SysML profile can be applied to the model and the SysML stereotypes applied where appropriate, thus preparing the model for use with UPDM Level 1 and SysML. When applying the UPDM Level 0 and the SysML profile, all of UML will be available to the modeler and all of the SysML stereotypes will also be available. This means the modelers can also apply other SysML stereotypes like Parametrics and Allocation to the SysML and UPDM and UML elements.

For example, if the model contains a Class that has already been stereotyped <<System>>, the modeler can now apply the SysML stereotype <<Block>> to the element:

Class <<System, Block>> MySystem

This gives MySystem the UPDM::System stereotype features and the SysML::Block stereotype features because we apply both stereotypes, UPDM::System and SysML::Block.

The modeler can also apply the SysML stereotype, Block, to a class:

Class <<Block>> MyBigBlock

This gives MyBigBlock only the SysML::Block features.

A.7.4 Apply UPDM Level 1 and SysML Profiles

In this scenario, System Engineers are working in a a SysML environment and applying the SysML stereotypes to model elements. UPDM Level 1 enables the System Engineers to apply the ten UPDM stereotypes (Table 6.1) in conjunction with the SysML artifacts transparently. When applying the UPDM Level 1 and the SysML profile, all of UML will be available to the modeler and all of the SysML stereotypes will also be available. This means the modelers can also apply other SysML stereotypes like Parametrics and Allocaiton to the SysML and UPDM and UML elements.

For example, the modeler creates a class and applies the Level 1 stereotype

<<System>>: Class <<System>> MySystem

This gives MySystem the UPDM::System stereotype features and the SysML::Block stereotype features because in Level-1, UPDM::System also specialize SysML::Block.

The modeler may also want to create elements that are not stereotyped with the UPDM stereotypes:

Class <<Block>> MyBigBlock This gives MyBigBlock only the SysML::Block features.

Note – SysML Strict Mode is not UPDM compliant.

Annex B
Domain Metamodel

(non-normative)

In the early design phases of UPDM, it was difficult to define a profile without a precise understanding of the semantics of DoDAF and MODAF in UML and SysML contexts. The Domain Metamodel (DMM) was created and maintained as a discussion vehicle as Profiles, Constraints, Proof of Concepts, Examples and Documentation were generated using an iterative process. Consideration was also given to other related modeling activities including the DoDAF 1.5 release and OMG SOA Profile RFP.

As the UPDM Profile matured and contained richer semantics, it was almost equivalent to the DMM. There are still model elements in the DMM are not in the Profile.

The DMM approach is a significant modeling work product to discuss DoDAF and MODAF from a non-UML profile perspective. It is a valuable asset for the OMG or government to maintain (i.e. DoDAF 2.0 basis).

B.1 Domain Metamodel Package Diagram

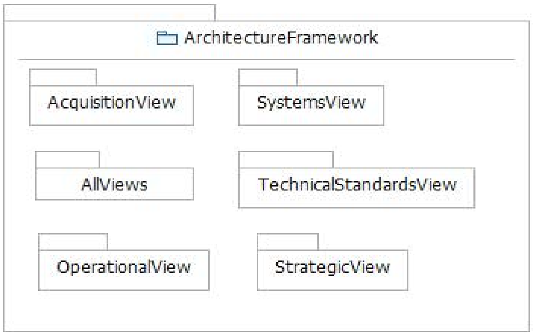


Figure B.1 - ArchitectureFramework Diagram

B.2 AcquisitionView Package

B.2.1 Package Diagrams

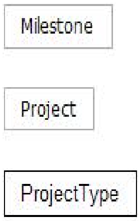


Figure B.2 - AcquisitionView Main Diagram

B.2.2 Milestone

B.2.2.1 Extension

B.2.2.2 Generalizations

B.2.2.3 Description

(20170/1) (A) A DECISION POINT THAT SEPARATES THE PHASES OF A DIRECTED, FUNDED EFFORT THAT IS DESIGNED TO PROVIDE A NEW OR IMPROVED MATERIAL CAPABILITY IN RESPONSE TO A VALIDATED NEED. - MODAF.

An event in a Project by which progress is measured.

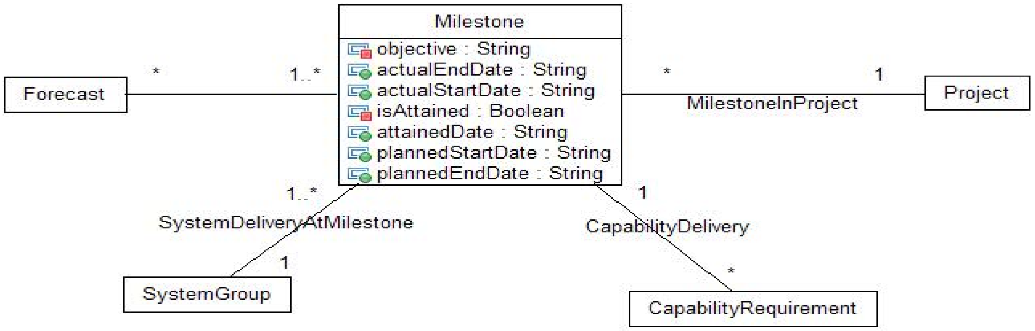


Figure B.3 - Milestone

B.2.2.4 Attributes

- actualEndDate : String [1]
- actualStartDate : String [1]
- attainedDate : String [1]

- isAttained : Boolean [1]
- objective : String [1]
- plannedEndDate : String [1]
- plannedStartDate : String [1]

B.2.2.5 Associations

- capabilityrequirement : CapabilityRequirement [*]
- project : Project [1]
- system group : SystemGroup [1]
- technology forecast : Forecast [*]

B.2.3 Project

B.2.3.1 Extension

B.2.3.2 Generalizations

B.2.3.3 Description

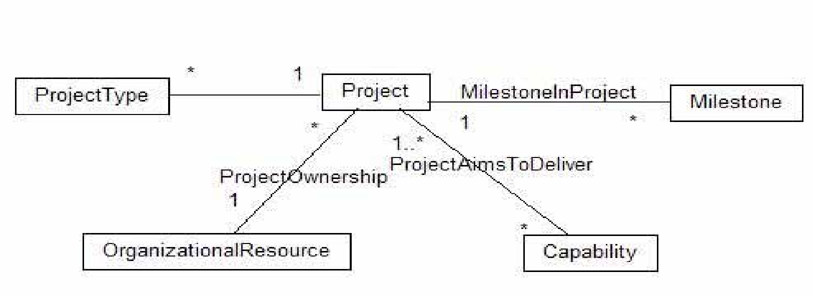


Figure B.4 - Project

B.2.3.4 Attributes

B.2.3.5 Associations

- capability : Capability [*]
- milestone : Milestone [*]
- organizationalresource : OrganizationalResource [1]
- projecttype : ProjectType [*]

B.2.4 ProjectType

B.2.4.1 Extension

B.2.4.2 Generalizations

- Project (from AcquisitionView)

B.2.4.3 Description

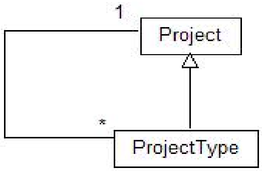


Figure B.5 - ProjectType

B.2.4.4 Attributes

B.2.4.5 Associations

- project : Project [1]

B.3 AllViews Package

B.3.1 Package Diagrams

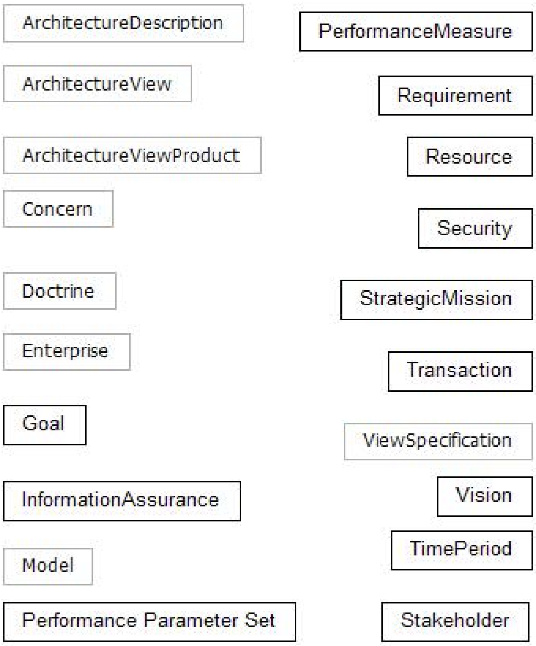


Figure B.6 - AllViews Main Diagram

B.3.2 ArchitectureDescription

B.3.2.1 Extension

B.3.2.2 Generalizations

B.3.2.3 Description

ArchitectureDescription: A specification of a system of systems at a technical level which also provides the business context for the system of systems.

B.3.2.4 Attributes

B.3.2.5 Associations

- architectureviewproduct : ArchitectureViewProduct [*]
- enterprise : Enterprise [1]
- model : Model [*]

B.3.3 ArchitectureView

B.3.3.1 Extension

B.3.3.2 Generalizations

B.3.3.3 Description

Purpose and Viewpoint explains the need for the architecture, what it should demonstrate, the types of analyses (e.g., Activity-Based Costing) that will be applied to it, who is expected to perform the analyses, what decisions are expected to be made on the basis of an analysis, who is expected to make those decisions, and what actions are expected to result. The viewpoint from which the architecture is developed is identified (e.g., planner or decision maker).

DoDAF defines Views. For example one of the Operational Views is the OV-6c. The OV-6c is a representation of a whole system from the perspective of a set of concerns. OV-6c defines the interactions of OperationalNodes. The OV-6c is a collection of UML elements and diagrams taken together. (it might be called a Model, but really would be just one part of a complete model.)

The Viewpoint that defines that View is the specification of OV-6c. The actual Viewpoint is the DoDAF Specification for OV-6c. If we look at the collection of Operational Views, they comprise another Viewpoint (the Operational View) that itself is defined in the DoDAF Specification.

Finally if we take the Operational View, the System View etc. as a Viewpoint that is also defined in DoDAF. So the Viewpoints are defined in the DoDAF Specification while the Views are the actual realization in the model. Thus, the Architecture Description, the UPDM Model, is a set of all the Views.

The collection mechanism for all of these levels of Views is the UML Package stereotyped as an ArchitectureView. Each of the “sub” viewpoints (the OV-6c, SV-5, etc.) defined in DoDAF has a corresponding stereotyped ArchitectureView that is a specialization of the root ArchitectureView and named as specified by DoDAF.

They are modular, so the Views (the packages) can be composed/assembled from sub packages that are stereotyped (recursively).

The submission also provides for direct reference to the Viewpoint Specification, the set of conventions for constructing, interpreting and analyzing a view, through an external reference mechanism that can link any element in the model to a named and typed reference identified by a URI.

In order to facilitate traceability, the submission includes the concepts of Concerns and Stakeholders, a specialization of Role. The Views, i.e., the ArchitectureViews can be associated with Concerns and Stakeholders within the UPDM model, and these concerns and Stakeholders can also be associated with their corresponding concepts in the DoDAF and other Documents using the External Reference mechanism.

This is a very direct and lightweight representation of the 1471 Viewpoint/View specification. It relies on the DoDAF specification to define Viewpoints and the corresponding Views being the realization of the Viewpoints using UML packages.. It is also extensible since each of the ArchitectureViews can be assigned a CustomType instead of one of the predefined DoDAF Views. The specifications - Viewpoints - for these Custom Views are referenced by the External Reference mechanism.

Viewpoint A set of conventions for constructing, interpreting and analyzing a view.

Views are well-formed:

- Each view corresponds to exactly one viewpoint

- A viewpoint is a pattern for constructing views
- Viewpoints define the rules on views
- No fixed set of viewpoints:
- IEEE 1471 is "agnostic" about where viewpoints come from
- Concerns drive viewpoint selection:
- Each concern is addressed by an architectural view
- Viewpoints are first-class:
- Each viewpoint used in an AD is "declared" before use

Each viewpoint is specified by:

- Viewpoint name
- The stakeholders addressed by the viewpoint
- The stakeholder concerns to be addressed by the viewpoint
- The viewpoint language, modeling techniques, or analytical methods used
- The source, if any, of the viewpoint (e.g., author, literature citation)
- A viewpoint may also include:
- Any consistency or completeness checks associated with the underlying method to be applied to models within the view
- Any evaluation or analysis techniques to be applied to models within the view
- Any heuristics, patterns, or other guidelines which aid in the synthesis of an associated view or its models

B.3.3.4 Attributes

- isStandardized : Boolean [1]
- source : String [1]

B.3.3.5 Associations

- architectureviewproduct : ArchitectureViewProduct [*]
- concern : Concern [1..*]
- stakeholder : Stakeholder [*]
- viewpointName : UPDMViewpoints [1]
- viewspecification : ViewSpecification [*]

B.3.4 ArchitectureViewProduct

B.3.4.1 Extension

B.3.4.2 Generalizations

B.3.4.3 Description

IEEE 1471: A specification of a way to present an aspect of the architecture. Views are defined with one or more purposes in mind - e.g., showing the logical topology of the enterprise, describing a process model, defining a Data model, etc. The architectural elements

Multiple views:

- An Architecture Description consists of one or more views.
- A view is a representation of a whole system from the perspective of a set of concerns.
- Views are themselves modular:
- A view may contain one or more architectural models, allowing a view to utilize multiple notations.
- Inter-view consistency:
- An AD documents any known inconsistencies among the views it contains.

A connected and coherent set of Architectural Elements which conform to a View.

B.3.4.4 Attributes

B.3.4.5 Associations

B.3.5 Doctrine

B.3.5.1 Extension

B.3.5.2 Generalizations

B.3.5.3 •

B.3.5.4 Description

Doctrine is represented as guard conditions, which are associated with events.

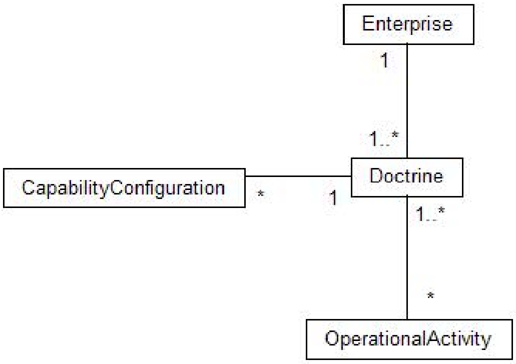


Figure B.7 - Doctrine

B.3.5.5 Attributes

B.3.5.6 Associations

- capabilityconfiguration : CapabilityConfiguration [*]
- enterprise : Enterprise [1]
- operationalactivity : OperationalActivity [*]

B.3.6 Enterprise

B.3.6.1 Extension

B.3.6.2 Generalizations

B.3.6.3 Description

An endeavor of any size involving people, Organizations and supporting systems.

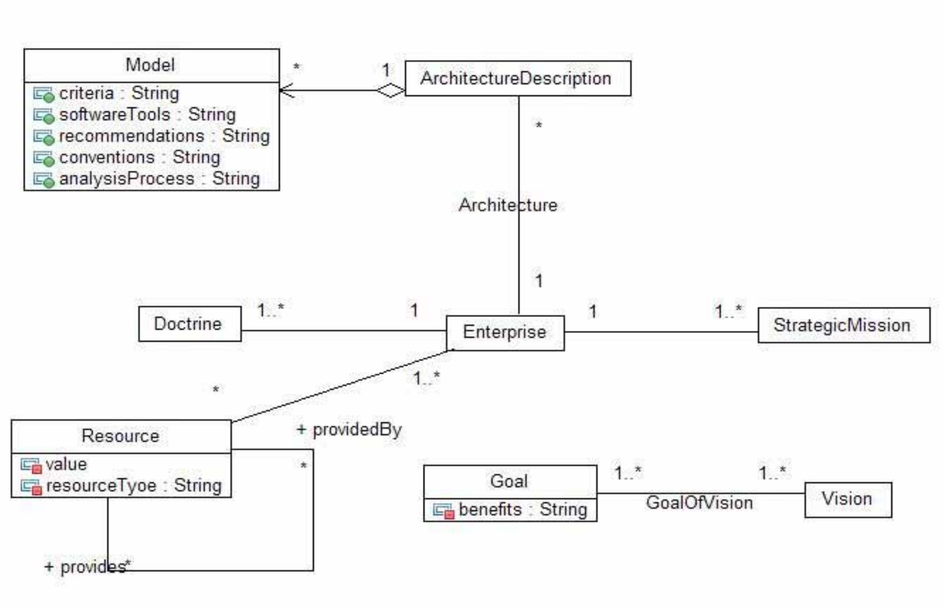


Figure B.8 - Enterprise

B.3.6.4 Attributes

B.3.6.5 Associations

- architecturedescription : ArchitectureDescription [*]
- doctrine : Doctrine [1..*]
- mission : StrategicMission [1..*]
- resource : Resource [*]

B.3.7 Goal

B.3.7.1 Extension

B.3.7.2 Generalizations

B.3.7.3 Description

A specific, required objective of the enterprise that the architecture represents.

Note: Benefits of achieving the goal are presented as a list of textual items

Some writers distinguish between goals (inexactly formulated aims that lack specificity) and objectives (aims formulated exactly and quantitatively as to time-frames and magnitude of effect). For example, a gambler might have the ambiguous goal: “I want to get lucky tonight.” Converting this into an objective, it might become: “I want to make $100 at the blackjack table by 8 o'clock tonight.” Not all authors make this distinction, preferring to use the two terms interchangeably.

B.3.7.4 Attributes

- benefits : String [1..*]

B.3.7.5 Associations

- deliveredBy : OperationalCapabilityRealization [1..*]
- vision : Vision [1..*]

B.3.8 InformationAssurance

B.3.8.1 Extension

B.3.8.2 Generalizations

B.3.8.3 Description

Information Assurance

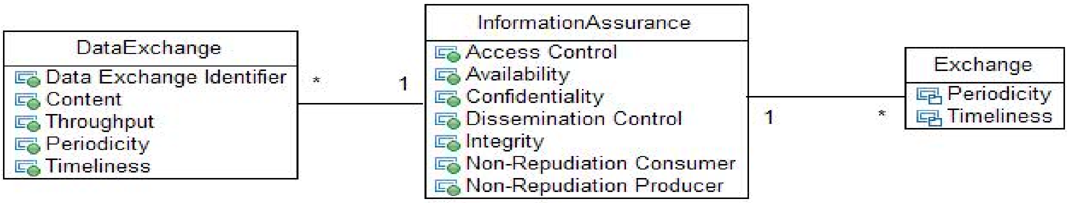


Figure B.9 - InformationAssurance

B.3.8.4 Attributes

- Access Control
- Availability
- Confidentiality
- Dissemination Control
- Integrity
- Non-Repudiation Consumer
- Non-Repudiation Producer

B.3.8.5 Associations

- data element : DataExchange [*]
- informationflow : Exchange [*]

B.3.9 Model Class

B.3.9.1 Extension

B.3.9.2 Generalizations

B.3.9.3 Description

DoDAF: A semantically complete abstraction of a system.

MODAF: An abstract or conceptual object or set of objects used in the representation of a real world entity or concept.

B.3.9.4 Attributes

- analysisProcess : String [1]
- conventions : String [1]
- criteria : String [1]
- recommendations : String [1]
- softwareTools : String [1]

B.3.9.5 Associations

B.3.10 Performance Parameter Set

B.3.10.1 Extension

B.3.10.2 Generalizations

B.3.10.3 Description

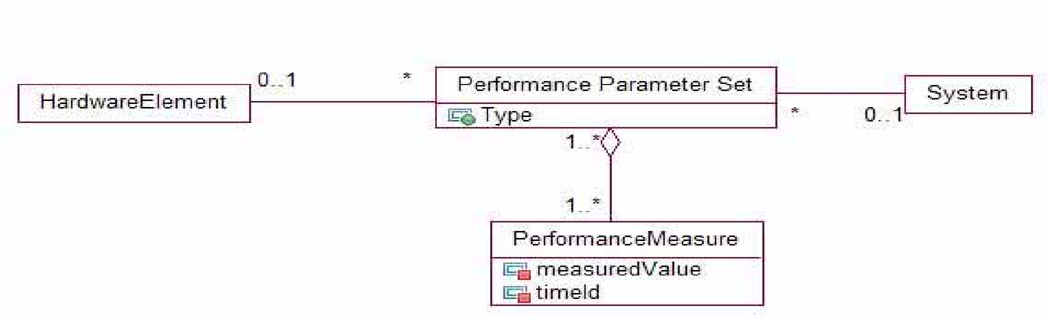


Figure B.10 - Performance Parameter Set

B.3.10.4 Attributes

B.3.10.5 Associations

- performance measures : PerformanceMeasure [1..*]

- system : System [0..1]
- system hardware : HardwareElement [0..1]

B.3.11 PerformanceMeasure

B.3.11.1 Extension

B.3.11.2 Generalizations

B.3.11.3 Description

Performance Parameters

The taxonomy minimally consists of names, descriptions, units of measure, and conditions that may be applicable to performance parameters.

B.3.11.4 Attributes

B.3.11.5 Associations

- performance parameter set : Performance Parameter Set [1..*]

B.3.12 Requirement Class

B.3.12.1 Extension

B.3.12.2 Generalizations

B.3.12.3 Description

B.3.12.4 Attributes

B.3.12.5 Associations

- capabilityrequirement : CapabilityRequirement [1]

B.3.13 Resource

B.3.13.1 Extension

B.3.13.2 Generalizations

B.3.13.3 Description

A PhysicalAsset or OrganizationalResource that can contribute towards fulfilling a Capability.

Systems Nodes that represent facilities, platforms, units, and locations.

The taxonomy minimally consists of names descriptions breakdowns into constituent parts of the node and categorizations of types of facilities, platforms, units, and locations.

B.3.13.4 Attributes

- resourceType : String [1]

B.3.13.5 Associations

- capabilityconfiguration : CapabilityConfiguration [*]
- enterprise : Enterprise [1..*]
- impact : Effect [*]
- organizationalrole : OperationalCapabilityRole [*]
- providedBy : Resource [*]
- providedCompetence : Competence [*]
- provides : Resource [*]
- resourceForCapability : Capability [1..*]

B.3.14 Security Class

B.3.14.1 Extension

B.3.14.2 Generalizations

B.3.14.3 Description

The defense domain requires sophisticated security models and this is a part of the domain that is evolving rapidly with publish/subscribe and service oriented architectures. The structure here for security can apply to any number of UPDM elements and is not constrained to the associations emphasized here. The associations emphasized here enable the delivery of existing DoDAF and MODAF work products.

A security domain has one or more resources. Each resource may have zero or more vulnerabilities. The risk of vulnerability is the effect it will cause to a resource if a loss occurs because of a threat that is manifested. The threats exploit one or more vulnerability. Safeguards mitigate vulnerabilities.

Mitigation responds proactively to protect assets from known vulnerabilities and threats.

B.3.14.4 Attributes

- classification : String [1]
  The classification for this item. An implementation tuned to a specific organization might make this an enumeration to reflect a specific security taxonomy.
- classificationCaveat: String [1]
  Special circumstances for release of information.
- protectionEndDate: String [1]
  Date when the protection on this information lapsed. If blank, the information must still comply by the security standards.
- protectionName: String [1]
  A name for the security protection

- resourceType : String [1]
- protectionType : String [1]
  The nature of the protection on the information or data.
- releasability : String [1]
  A description of how the information can be released.

B.3.14.5 Associations

B.3.15 Stakeholder Class

B.3.15.1 Extension

B.3.15.2 Generalizations

- OrganizationalResource (from OperationalView)

B.3.15.3 Description

Someone who has an interest in an Enterprise.

IEEE-1471: An individual, team or Organization (or classes thereof) with interests in, or concerns relative to, a system.

Some Typical Stakeholders - stakeholders are really roles:

- Client
- Acquirer
- Owner
- User
- Operator
- Architect
- System Engineer
- Developer
- Designer
- Builder
- Maintainer
- Service Provider
- Vendor
- Subcontractor
- Planner

B.3.15.4 Attributes

B.3.15.5 Associations

- concern : Concern [1..*]
- viewpoint : ArchitectureView [1]

B.3.16 StrategicMission

B.3.16.1 Extension

B.3.16.2 Generalizations

B.3.16.3 Description

Summarize goals and objectives into a mission statement and / or a vision statement:

> * A definition of Mission in a dictionary: purpose, reason for being; also, an inner calling to pursue an activity or perform a service.

A Mission statement may define the purpose or broader goal for being in existence or in the business. It serves as an ongoing guide without time frame. The mission can remain the same for decades if crafted well. Vision is more specific in terms of objective and future state. Vision is related to some form of achievement if successful.

Mission and Values go hand in hand. To make the mission statement effective, it needs to be aligned with the prevailing culture of its stakeholders, organization, market and political sphere. A lofty mission statement means nothing if it is not in congruence with the values practiced by the organization. A statement of values provides guiding principles when ethical issues related to realizing the Vision, and undertaking the Mission, arise.

A mission statement provides a path to realize the vision in line with its values. These statements have a direct bearing on the bottom line and success of the organization.

Military Mission: An objective together with the purpose of the intended action. (Extension of DDDS 1(A))

Note: Multiple tasks accomplish a mission. (Space and Naval Warfare Systems Command)

B.3.16.4 Attributes

B.3.16.5 Associations

- architecture scope : Enterprise [1]
- tasks : OperationalCapabilityRealization [1..*]

B.3.17 TimePeriod

B.3.17.1 Extension

B.3.17.2 Generalizations

B.3.17.3 Description

A period of time, defined by start and end dates - sometimes termed an "Epoch" in the MOD. Time periods may overlap

B.3.17.4 Attributes

- end : String [1]
- start : String [1]

B.3.17.5 Associations

- capabilityrequirement : CapabilityRequirement [1]
- vision : Vision [1]

B.3.18 Transaction

B.3.18.1 Extension

B.3.18.2 Generalizations

B.3.18.3 Description

A construct to link the information exchange and data flow used by the architecture. If the architecture extends to deal with Transactions that involve items other than information, this description of Transaction would also apply to those exchanges. Typically, a Transaction will have an association with both an InformationExchange or a DataFlow, but that is not made a constraint to allow users to allow Transactions to be used for other flows.

Semantics

The Transaction connects the InformationExchange and DataFlow with a number of attributes describing the exchange.

When an InformationExchange stereotype is created, it should include a static variable of type Transaction. When a DataExchange stereotype is created, it should include a static variable of type Transaction. When a Transaction instance specification is created, and when the user assigns the InformationExchange and DataExchange to the stereotype properties, the default value of the respective statics should be set to the Transaction Instance Specification.

Transaction applies to DataExchange or InformationExchange

B.3.18.4 Attributes

- criticality: String [1]
  Criticality of the transaction

- transaction Type: String [1]
  User defined type
- interoperability Level: String [1]
  Interoperability evaluation measurement - e.g., NCOIC's NCAT
- startTime : String [1]
  Time transaction starts
- completionTime: String [1]
  Time transaction completed.
- triggering Event: String [1]
  A description of how the information can be released.

B.3.18.5 Associations

- data flow : DataExchange [*]
- informationflow : Exchange [*]

B.3.19 ViewSpecification

B.3.19.1 Extension

B.3.19.2 Generalizations

B.3.19.3 Description

B.3.19.4 Attributes

B.3.19.5 Associations

B.3.20 Vision

B.3.20.1 Extension

B.3.20.2 Generalizations

B.3.20.3 Description

A high-level textual description of the "aims" of the enterprise. The overall aims of an Enterprise over a given period of time. Organizations sometimes summarize goals and objectives into a mission statement and / or a vision statement:

- A Definition of Vision in a dictionary: 'An image of the future we seek to create'.

A vision statement describes in graphic terms where the goal-setters want to see themselves in the future. It may describe how they see events unfolding over 10 or 20 years if everything goes exactly as hoped.

The Vision describes a future identity and the Mission describes how it will be achieved. Vision is more specific in terms of objective and future state. Vision is related to some form of achievement if successful.

For example, “We help transport goods and people efficiently and cost effectively without damaging environment” is a mission statement. Ford’s brief but powerful slogan “Quality is Job 1”could count as a mission statement. “We will be one amongst the top three transporters of goods and people in North America by 2010” is a vision statement. It is very concrete and unambiguous goal.

The vision statement can galvanize the people to achieve defined objectives, even if they are stretch objectives, provided the vision is SMART (Specific, Measurable, Achievable, Realistic and Timebound). A mission statement provides a path to realize the vision in line with its values. These statements have a direct bearing on the bottom line and success of the organization.

Features of an effective vision statement may include:

- Clarity and lack of ambiguity
- Paint a vivid and clear picture, not ambiguous
- Describing a bright future (hope)
- Memorable and engaging expression
- Realistic aspirations, achievable
- Alignment with organizational values and culture, Rational
- Time bound if it talks of achieving any goal or objective

In order to become really effective, an organizational vision statement must (the theory states) become assimilated into the organization's culture. Leaders have the responsibility of communicating the vision regularly, creating narratives that illustrate the vision, acting as role-models by embodying the vision, creating short-term objectives compatible with the vision, and encouraging others to craft their own personal vision compatible with the organization's overall vision

B.3.20.4 Attributes

B.3.20.5 Associations

- capability : Capability [*]
- goal : Goal [1..*]
- timeperiod : TimePeriod [1]

B.4 OperationalView Package

B.4.1 Package Diagrams

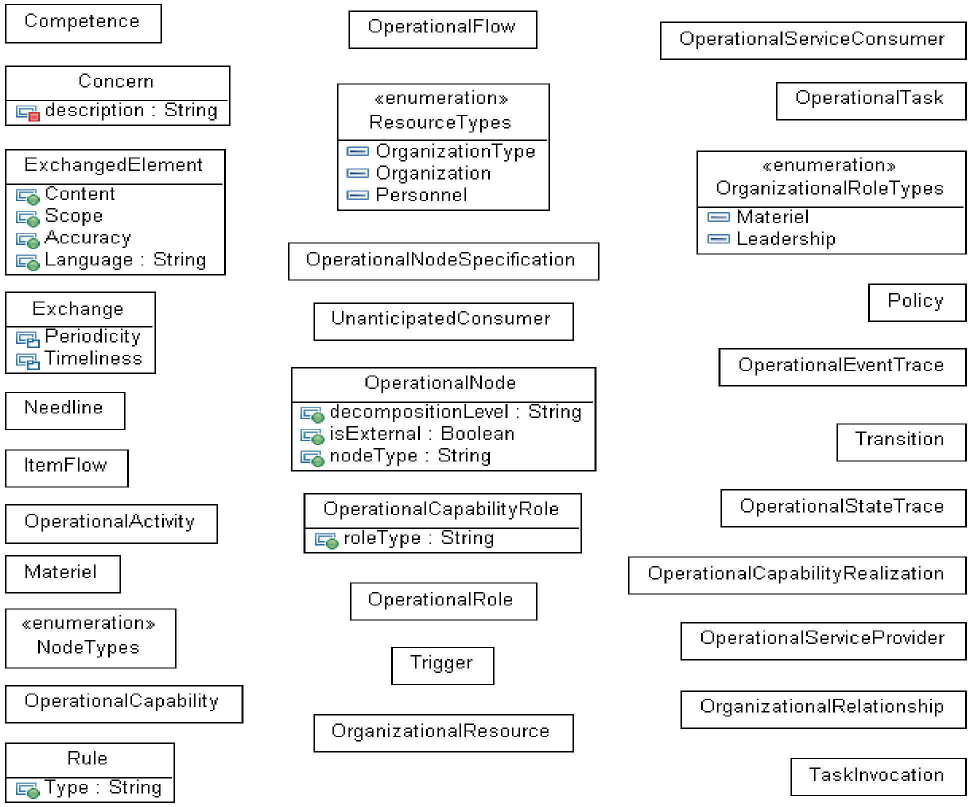


Figure B.11 - OperationalView Package Main Diagram

B.4.2 Competence

B.4.2.1 Extension

B.4.2.2 Generalizations

B.4.2.3 Description

B.4.2.4 Attributes

B.4.2.5 Associations

- organizationalresource : Resource [*]
- requiredByRole : OperationalCapabilityRole [*]

B.4.3 Concern

B.4.3.1 Extension

B.4.3.2 Generalizations

B.4.3.3 Description

An interest in a subject held by one or more Stakeholders

B.4.3.4 Attributes

- description : String [1]

B.4.3.5 Associations

- architectureview : ArchitectureView [1..*]
- stakeholder : Stakeholder [1..*]

B.4.4 Exchange

B.4.4.1 Extension

B.4.4.2 Generalizations

B.4.4.3 Description

The collection of information elements and their performance attributes such as timeliness, quality, and quantity values. (UPDM)

A specification of the information that is to be exchanged an InformationExchange must have a unique identifier. Note: additional information about the requirements for the InformationExchange may be provided by the requirementText attribute Information.

B.4.4.4 Attributes

- periodicity : String [1]
- timeliness : String [1]
- exchangedId: String[1]
- securityFeatures: Security[1]

B.4.4.5 Associations

- causedEffect : Effect [*]
- information assurance : InformationAssurance [1]
- information element : ExchangedElement [1..*]
- needline : Needline [1]

- operationalactivity : OperationalFlow [*]
- operationalactivityevent : TaskInvocation [*]
- systemInterface : SystemInterface [1..*]
- transaction : Transaction [0..1]

B.4.5 ExchangedElement

B.4.5.1 Extension

B.4.5.2 Generalizations

B.4.5.3 Description

Information Elements consists of names of information elements exchanged, descriptions, decomposition into constituent parts and subtypes, and mapping to system data elements exchanged.

Information that is passed from one operational node to another and associated with an information element and such performance attributes as timeliness, quality, and quantity values.

A formalized representation of information subject to an operational process

B.4.5.4 Attributes

- Language : String [1]
- Accuracy: String[1]
- Content: String[1]
- Scope: String[1]

B.4.5.5 Associations

- operationalinformationflow : ItemFlow [1..*]

B.4.6 ItemFlow

B.4.6.1 Extension

B.4.6.2 Generalizations

B.4.6.3 Description

OperationalInformationFlow is the flow of objects, InformationElements, within OperationalActivities.

A flow of information, energy or materiel from one activity to another

B.4.6.4 Attributes

B.4.6.5 Associations

- informationelement : ExchangedElement [1]

B.4.7 Materiel

B.4.7.1 Extension

B.4.7.2 Generalizations

B.4.7.3 Description

Materiel is tied to the elements in OV-5, where mechanisms may be used to represent systems that support operational activities. In addition, further materiel detail may be related to the activities via SV-5, by relating those activities to the system functions that are executed by systems that automate them (wholly or partially). Each operational thread or scenario (represented by an OV-6c) is associated with a certain capability, since a capability is defined in terms of the activities and their attributes depicted in OV-6c. Consequently, an SV-5 may also be used to relate a capability to the systems that support it, by labeling a set of activities with its associated capability (defined in OV-6c), and by labeling the system functions with the systems that execute them (defined in SV-1, SV-2, and SV-4)

B.4.7.4 Attributes

B.4.7.5 Associations

- operational activity : OperationalActivity [*]
- operationaltask : OperationalTask [*]

B.4.8 Needline

B.4.8.1 Extension

B.4.8.2 Generalizations

B.4.8.3 Description

A requirement that is the logical expression of the need to transfer information among nodes. Needlines are DIRECTIONAL.

Needlines and Information Exchanges: A Needline documents the requirement to exchange information between nodes. The Needline does not indicate how the information transfer is implemented. For example, if information is produced at node A, is simply routed through node B, and is used at node C, then node B would not be shown on the OV-2 diagram the Needline would go from node A to node C. OV-2 is not a communications link or communications network diagram. The system implementation (or what systems nodes or systems are used to execute the transfer) is shown in the Systems Interface Description (SV-1).

Furthermore, the Needline systems equivalent is the interface line depicted in SV-1. The actual implementation of an interface may take more than one form and is documented in a Systems Communications Description (SV-2). Therefore, a single Needline shown in the OV may translate into multiple interfaces in SV-1 and multiple physical links in SV-2.

B.4.8.4 Attributes

B.4.8.5 Associations

- consumingOpNode : OperationalNode [1]
- informationFlow : Exchange [1..*]
- providingOpNode : OperationalNode [1]
- systeminterface : SystemInterface [1..*]

B.4.9 OperationalActivity

B.4.9.1 Extension

B.4.9.2 Generalizations

B.4.9.3 Description

Operational Activities consists of names, descriptions, and decomposition into the constituent parts that comprise a process-activity.

Operational Activities are defined and standardized by the Joint Staff are in the form of Mission Essential Tasks [CJCSM 3500.04C, 2002].

Operational Activities are also specified (and sometimes standardized) in the form of process activities arising from process modeling. It is sometimes convenient to merge these sets, either as activities or tasks.

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 (e.g., 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 Operational Activity Model). It is used to portray operational actions not hardware/software system functions.

Operational Activities: The operational activities (from the OV-5 Operational Activity Model) performed by a given node may be listed on the graphic, if space permits. OV-2, in effect, turns OV-5 inside out, focusing first-order on the operational nodes and second-order on the activities. OV-5, on the other hand, places first-order attention on operational activities and only second-order attention on nodes, which can be shown as annotations on the activities.

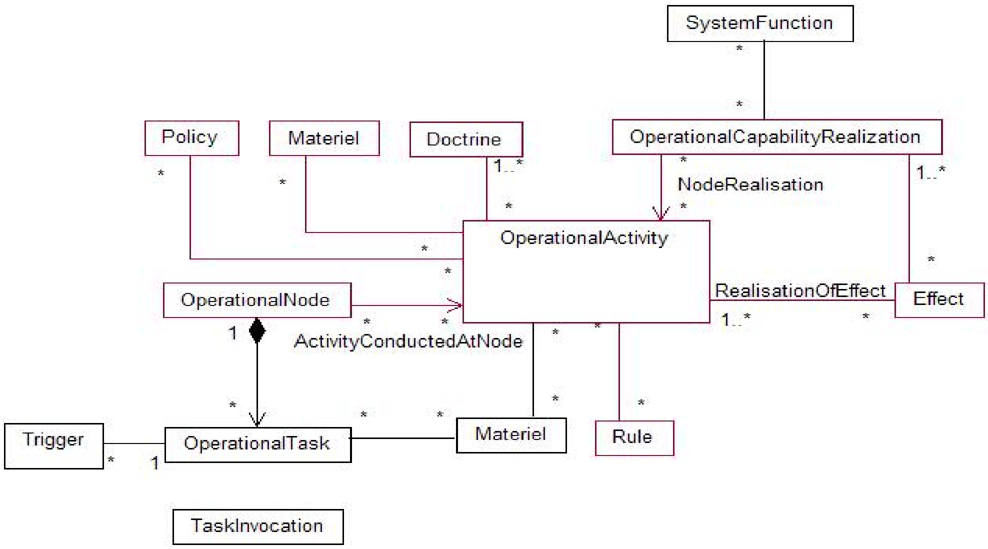


Figure B.12 - OperationalActivity

B.4.9.4 Attributes

B.4.9.5 Associations

- activity constraints : Rule [*]
- doctrine : Doctrine [1..*]
- effect : Effect [*]
- materiel : Materiel [*]
- operationalactivityflow : OperationalActivity [*]
- operationalactivityflow : OperationalFlow [*]
- operationalactivityflow2 : OperationalFlow [*]
- operationalinformationflow : ItemFlow [*]
- policy : Policy [*]
- system function : OperationalActivityRealization [*]

B.4.10 OperationalCapability

B.4.10.1 Extension

B.4.10.2 Generalizations

B.4.10.3 Description

B.4.10.4 Attributes

B.4.10.5 Associations

- capability : Capability [1]
- operationalcapabilityrealization : OperationalCapabilityRealization [1..*]

B.4.11 OperationalCapabilityRealization

B.4.11.1 Extension

B.4.11.2 Generalizations

B.4.11.3 Description

Enduring task is also an 'OperationalCapability' -- Source: MODAF

EnduringTask: A type of behavior recognized by an enterprise as being essential to achieving its goals - i.e. a strategic specification of what the enterprise does.

Note: This is equivalent to a task in an essential task list (JETL)

B.4.11.4 Attributes

B.4.11.5 Associations

- achievesGoal : Goal [1..*]
- capability : Capability [1]
- capabilityrequirement : CapabilityRequirement [*]
- effect : Effect [*]
- mission : StrategicMission [1]
- operational thread : OperationalActivity [*]
- operationalactivityevent : OperationalEventTrace [*]
- operationalcapability : OperationalCapability [1]
- operationalstatetrace : OperationalStateTrace [*]
- systemfunction : SystemFunction [*]

B.4.12 OperationalCapabilityRole

B.4.12.1 Extension

B.4.12.2 Generalizations

- OperationalNode (from OperationalView)

B.4.12.3 Description

An OperationalRole defines an association between the concept represented by an OperationalNode and the OrganizationalResource (OperationalActors) that actually perform the operations of the OperationalNodes to accomplish the node's capabilities. An OperationalRole's operations are the collection of the OperationalNode's OperationalActivities that are performed in that role.

An OperationalCapabilityRole defines an association between the concept represented by an OperationalNode and the OrganizationalResource (OperationalActors) that actually perform the operations of the OperationalNodes to accomplish the node's capabilities. An OperationalCapabilityRole's operations are the OperationalTasks that are performed in that role.

An OperationalNode is an abstract representation of capabilities that are achieved through the OperationalActivities.

The OperationalCapabilityRole is an explicit representation of that Role that enables modeling the assignment of the OrganizationalResources who will play that role.

Defines a role in an Organization which is fulfilled by a post or by a sub-ordinate organization. The role may or may not be fulfilled (i.e., roleProvider is null) - e.g., it may be a vacant post.

B.4.12.4 Attributes

- roleType : String [1]

B.4.12.5 Associations

- requiredCompetence : Competence [*]
- resource : Resource [*]

B.4.13 OperationalEventTrace

B.4.13.1 Extension

B.4.13.2 Generalizations

B.4.13.3 Description

Several modeling techniques may be used to refine and extend the architecture's OV to adequately describe the dynamic behavior and timing performance characteristics of an architecture, such as logical languages such as LDL [Naqvi, 1989] Harel Statecharts [Harel, 1987 a, b], Petri-nets [Kristensen, 1998], IDEF3 diagrams [IDEF3, 1995], and UML statechart and sequence diagrams [OMG, 2001]. OV-6 includes three such models. They are:

- Operational Rules Model (OV-6a)
- Operational State Transition Description (OV-6b)
- Operational Event-Trace Description (OV-6c)

The Operational State Transition Description 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.

B.4.13.4 Attributes

B.4.13.5 Associations

- operActivityEvent : TaskInvocation [1..*]

B.4.14 OperationalFlow

B.4.14.1 Extension

B.4.14.2 Generalizations

B.4.14.3 Description

The flow of control from one action to another that specifies the actions within an OperationalActivity. The Actions invoke either specific OperationalTasks or other OperationalActivities. An OperationalFlow has a source Action (FROM) and a target Action (TO). An OperationalControlFlow specifies partially or fully one InformationExchange.

B.4.14.4 Attributes

B.4.14.5 Associations

- from : OperationalActivity [1]
- informationexchange : Exchange [*]
- to : OperationalActivity [1]

B.4.15 OperationalNode

B.4.15.1 Extension

B.4.15.2 Generalizations

B.4.15.3 Description

Operational Nodes that represent Organizations, Organization Types, and Occupational Specialties.

The taxonomy minimally consists of names, descriptions, and breakdowns into the parts of the organization, organization type, or human role.

DoDAF1: An Operational Node is an element of the operational architecture that produces, consumes, or processes information. What constitutes an operational node can vary among architectures, including, but not limited to, representing an operational/human role (e.g., Air Operations Commander), an organization (e.g., Office of the Secretary of Defense) or organization type, i.e., a logical or functional grouping (e.g., Logistics Node, Intelligence Node), and so on. The notion of operational node will also vary depending on the level of detail addressed by the architecture effort.

UPDM: A node that performs a role or mission. A conceptual architectural element that produces, consumes or processes information. For example, an OrganizationOperationalNode may own systems that process information. In this case, the Organizational Operational Node would be realized by an Organization.

An Operational Node can be a composite structure that is decomposed into finer grained operational nodes as the architecture is decomposed. UPDM provides a refinement of Operational Node by including specific subclasses that implement the Composite/Component pattern to clarify modeling the decomposition of Operational Nodes.

In addition, the concept of an actor or resource that realizes an Operational Node has been made explicit. To enrich and further clarify the differences among Operational Node, Actor and Role, UPDM includes an explicit Role and an association: "anActor playsRole aRole ". This Operational Node complex is further extended to include a hierarchy of OperationalNodes and their corresponding actor types and role types that they play.

The relationship between architecture data elements across the SV to the Operational View (OV) can be exemplified as systems are procured and fielded to support organizations and their operations. SV-1 links together the OV and SV by depicting the assignments of systems and systems nodes (and their associated interfaces) to the operational nodes (and their associated Needlines) described in OV-2. OV-2 depicts the operational nodes representing organizations, organization types, and/or human roles, while SV-1 depicts the systems nodes that house operational nodes (e.g., platforms, units, facilities, and locations) and the corresponding systems resident at these systems nodes and which support the operational 5-2 nodes.

Operations on OperationalNodes are stereotyped as OperationalActivities for all operations that are to be considered as InformationExchange messages.

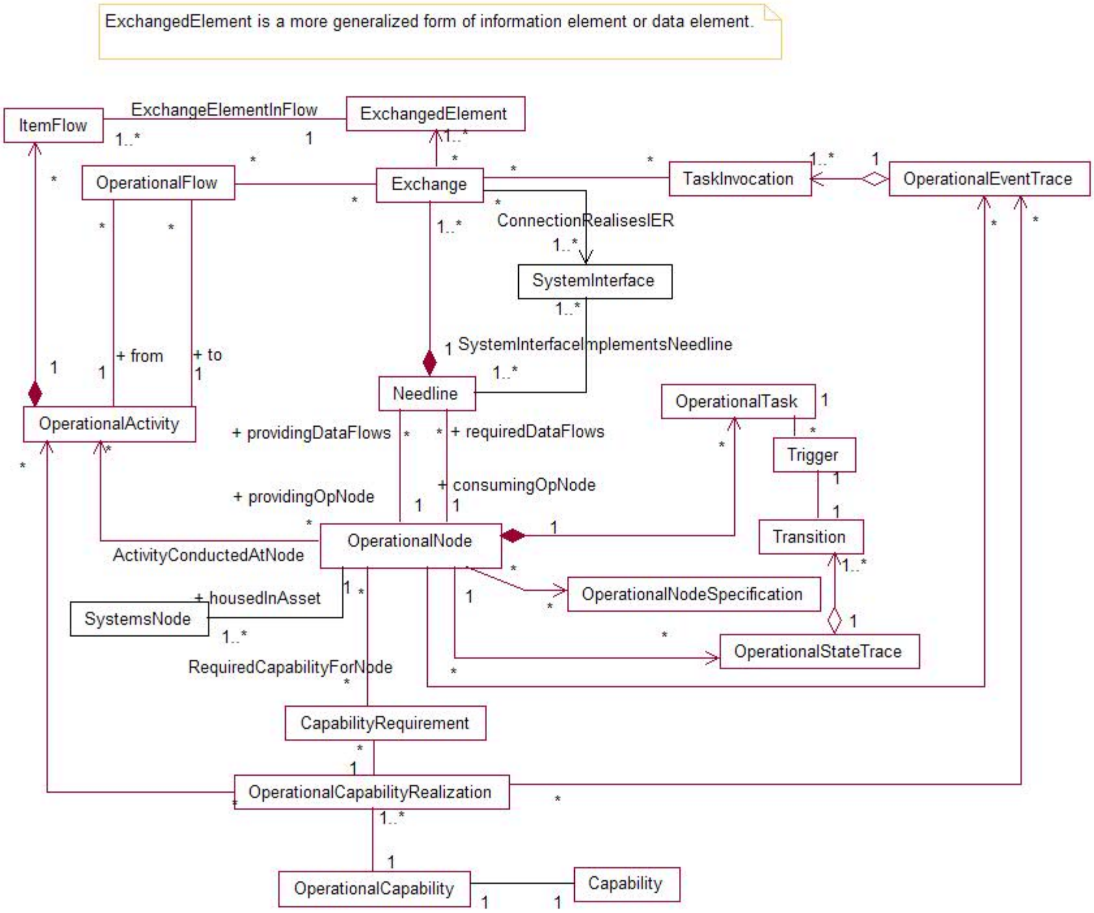


Figure B.13 - OperationalNode

B.4.15.4 Attributes

- decompositionLevel : String [1]
- isExternal : Boolean [1]
- nodeType : String [1]

B.4.15.5 Associations

- capabilityrequirement : CapabilityRequirement [*]
- eventtrace : OperationalEventTrace [*]
- housedInAsset : SystemsNode [1..*]
- implementedNodeInterfaces : OperationalNodeSpecification [*]
- operationalActivity : OperationalActivity [*]
- operationalactivityevent3 : OperationalTask [*]

- operationalstatetrace : OperationalStateTrace [*]
- providingDataFlows : Needline [*]
- requiredDataFlows : Needline [*]

B.4.16 OperationalNodeSpecification

B.4.16.1 Extension

B.4.16.2 Generalizations

B.4.16.3 Description

B.4.16.4 Attributes

B.4.16.5 Associations

B.4.17 OperationalRole

B.4.17.1 Extension

B.4.17.2 Generalizations

- OperationalCapabilityRole (from OperationalView)

B.4.17.3 Description

There are two critical constructs that can be used in a Net-Centric OV-2 diagram to illustrate Service Providers, Service Consumers, and Unanticipated Users along with the need to provide or consume information. The constructs are (a) the operational roles which are applied to operational nodes and (b) Needlines which identify the need to exchange information.

The use of operational role applied to operational nodes stresses the responsibility the node plays in interacting with external organizations and human roles.

B.4.17.4 Attributes

B.4.17.5 Associations

B.4.18 OperationalServiceConsumer

B.4.18.1 Extension

B.4.18.2 Generalizations

- OperationalRole (from OperationalView)

B.4.18.3 Description

The operational role of Service Consumer helps to illustrate the set of expected users of information. Needlines associated with Nodes that have a Service Consumer role indicates that information is required by that consumer. The operational role of Service Consumer may be applied to both the internal system or to external systems that require needed functionality.

Any classifier (class, component, and so on) may act as the consumer of a service, and that includes another service. While this stereotype is most definitely optional, it may help you identify elements of a model - that are not services themselves -- as clients of services. On the other hand, it may just be overhead and you don't need to use it.

B.4.18.4 Attributes

B.4.18.5 Associations

- consumed : Service [*]

B.4.19 OperationalServiceProvider

B.4.19.1 Extension

B.4.19.2 Generalizations

- OperationalRole (from OperationalView)

B.4.19.3 Description

The operational role of Service Provider helps to illustrate the set of known sources of information. Needlines associated with Nodes that have a Service Provider role indicates that information is provided by that source. The operational role of Service Provider may be applied to both the internal system or to external systems that provide needed information.

The service provider is a software element that provides one or more services. A service provider has a property that captures information about its location, although the meaning of this is implementation-dependent. The class acting as the service provider may not expose any attributes or operations directly: only public ports may be provided (stereotyped as service), and these are typed by service specifications.

B.4.19.4 Attributes

B.4.19.5 Associations

- provided : Service [*]

B.4.20 OperationalStateTrace

B.4.20.1 Extension

B.4.20.2 Generalizations

B.4.20.3 Description

OperationalStateTrace is a state machine model of operational node behavior.

B.4.20.4 Attributes

B.4.20.5 Associations

- operationalcapabilityrealization : OperationalCapabilityRealization [1]
- trigger : Transition [1..*]

B.4.21 OperationalTask

B.4.21.1 Extension

B.4.21.2 Generalizations

B.4.21.3 Description

OperationalTask is an operation on OperationalNodes that provide the means to invoke OperationalActivities.

B.4.21.4 Attributes

B.4.21.5 Associations

- materiel : Materiel [*]
- triggering event : Trigger [*]

B.4.22 OrganizationalRelationship

B.4.22.1 Extension

B.4.22.2 Generalizations

B.4.22.3 Description

An assertion that two instances of OrganizationType typically are related.

B.4.22.4 Attributes

B.4.22.5 Associations

- organization : OrganizationalResource [2..2]

B.4.23 OrganizationalResource

B.4.23.1 Extension

B.4.23.2 Generalizations

- Resource (from AllViews)

B.4.23.3 Description

An OrganizationalResource is the resource that realizes an Operational Node. Specific specializations of OrganizationalResource reflect the specialized OperationalNodes: Human, Organization, and Organization Type.

B.4.23.4 Attributes

B.4.23.5 Associations

- capabilityconfiguration : CapabilityConfiguration [*]
- organizationrelationship : OrganizationalRelationship [*]
- physicalasset : SystemsNode [1..*]
- project : Project [*]

B.4.24 Policy

B.4.24.1 Extension

B.4.24.2 Generalizations

B.4.24.3 Description

Any policy or doctrine references that provide further explanation of the operational activity

B.4.24.4 Attributes

B.4.24.5 Associations

- operationalactivity : OperationalActivity [*]

B.4.25 Rule

B.4.25.1 Extension

B.4.25.2 Generalizations

B.4.25.3 Description

A constraint on the behavior of an element in the enterprise architecture.

B.4.25.4 Attributes

- Type : String [1]

B.4.25.5 Associations

- operationalactivity : OperationalActivity [*]
- system : System [*]

B.4.26 TaskInvocation

B.4.26.1 Extension

B.4.26.2 Generalizations

B.4.26.3 Description

The event that invokes either an OperationalTask or a SystemTask in the EventTrace.

B.4.26.4 Attributes

B.4.26.5 Associations

- informationexchange : Exchange [*]

B.4.27 Transition

B.4.27.1 Extension

B.4.27.2 Generalizations

B.4.27.3 Description

B.4.27.4 Attributes

B.4.27.5 Associations

- trigger : Trigger [1]

B.4.28 Trigger

B.4.28.1 Extension

B.4.28.2 Generalizations

B.4.28.3 Description

Triggers/Events

The taxonomy minimally consists of names, descriptions, and breakdown into constituent parts of the event or trigger and categorization of types of events or triggers.

Several modeling techniques may be used to refine and extend the architecture's OV to adequately describe the dynamic behavior and timing performance characteristics of an architecture, such as logical languages such as LDL [Naqvi, 1989] Harel Statecharts [Harel, 1987 a, b], Petri-nets [Kristensen, 1998], IDEF3 diagrams [IDEF3, 1995], and UML state chart and sequence diagrams [OMG, 2001]. OV-6 includes three such models. They are:

- Operational Rules Model (OV-6a)
- Operational State Transition Description (OV-6b)
- Operational Event-Trace Description (OV-6c)

B.4.28.4 Attributes

B.4.28.5 Associations

- operationalactivity : OperationalTask [1]
- system : System [*]
- transition : Transition [1]

B.4.29 UnanticipatedConsumer

B.4.29.1 Extension

B.4.29.2 Generalizations

- ServiceConsumer (from SystemsView)

B.4.29.3 Description

B.4.29.4 Attributes

B.4.29.5 Associations

B.5 StrategicView Package

B.5.1 Package Diagrams

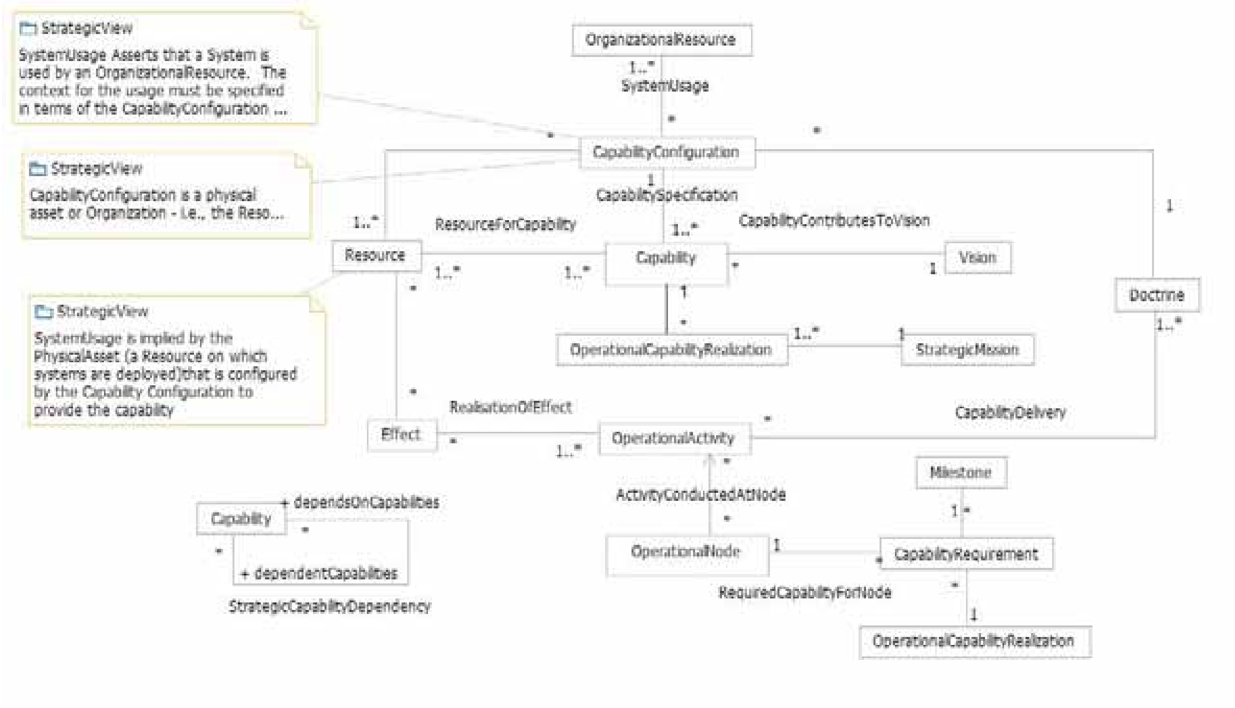

StrategicView
SystemUsage Asserts that a System is used by an OrganizationalResource. The context for the usage must be specified in terms of the CapabilityConfiguration ...
StrategicView
CapabilityConfiguration is a physical asset or Organization - i.e., the Reso...
StrategicView
SystemUsage is implied by the PhysicalAsset (a Resource on which systems are deployed)that is configured by the Capability Configuration to provide the capability
OrganizationalResource
1..*
SystemUsage
*
*
CapabilityConfiguration
*
1
CapabilitySpecification
1..*
ResourceForCapability
CapabilityContributesToVision
1
1..*
Resource
1..*
1..*
Capability
*
1
Vision
Doctrine
1..*
*
1
*
OperationalCapabilityRealization
1..*
1
StrategicMission
*
RealisationOfEffect
CapabilityDelivery
Effect
*
1..*
OperationalActivity
*
*
Milestone
Capability
+ dependsOnCapabilities
ActivityConductedAtNode
1
*
*
*
+ dependentCapabilities
OperationalNode
1
*
CapabilityRequirement
StrategicCapabilityDependency
RequiredCapabilityForNode
*
1
OperationalCapabilityRealization

Figure B.14 - StrategicView Main Diagram

B.5.2 Capability

B.5.2.1 Extension

B.5.2.2 Generalizations

B.5.2.3 Description

A high level specification of the enterprise's ability

OV-5 is a key product for describing capabilities and relating capabilities to mission accomplishment. The DoD Dictionary of Military Terms [DoD JP 1-02, 2001] defines a capability as “the ability to execute a specified course of action. (A capability may or may not be accompanied by an intention.)” A capability can be defined by one or more sequences of activities, referred to as operational threads or scenarios. A capability may be further described in terms of the attributes required to accomplish the set of activities (such as the sequence and timing of operational activities or materiel that enable the capability), in order to achieve a given mission objective. Capability-related attributes may be associated with specific activities or with the information flow between activities, or both. When represented by a set of operational activities, a capability can also be linked to an operational node in an OV-2.

May be defined in terms of the attributes required to accomplish a set of activities in order to achieve a given mission objective; capability-related attributes may be associated with specific activities.

Name Name of capability

Source: MODAF

(T1 Capability)

A type of behavior recognized by an enterprise as being essential to achieving its goals - i.e., a strategic specification of what the enterprise does.

Note: This is equivalent to a task in an essential task list (JETL)

Source: DoDAF Vol II

The ability to execute a specified course of action. (JP 1-02) It is defined by an operational user and expressed in broad operational terms in the format of an initial capabilities document or a DOTMLPF change recommendation. In the case of materiel proposals, the definition will progressively evolve to DOTMLPF performance attributes identified in the CDD and CPD. (CJCSI 3170.01C) (A capability may or may not be accompanied by an intention.)" A capability can be defined by one or more sequences of activities, referred to as operational threads or scenarios. A capability may be further described in terms of the attributes required to accomplish the set of activities (such as the sequence and timing of operational activities or materiel that enable the capability), in order to achieve a given mission objective. Capability-related attributes may be associated with specific activities or with the information flow between activities, or both.

When represented by a set of operational activities, a capability can also be linked to an operational node in an OV-2.

B.5.2.4 Attributes

- isFielded : Boolean [1]

B.5.2.5 Associations

- capabilityconfiguration : CapabilityConfiguration [1]
- dependentCapabilities : Capability [*]
- dependsOnCapabilities : Capability [*]
- operationalcapability : OperationalCapability [1]
- operationalcapabilityrealization : OperationalCapabilityRealization [*]
- project : Project [1..*]
- resource : Resource [1..*]
- vision : Vision [1]

B.5.3 CapabilityConfiguration

B.5.3.1 Extension

B.5.3.2 Generalizations

B.5.3.3 Description

A CapabilityConfiguration is a combination of organizational aspects with their competencies and equipment, i.e., Resources that combine to provide a Capability. A CapabilityConfiguration is a physical asset or organization configured to provide a capability, and must be guided by Doctrine. (Source MODAF).>

B.5.3.4 Attributes

B.5.3.5 Associations

- capability : Capability [1..*]
- doctrine : Doctrine [1]
- organizationalresource : OrganizationalResource [1..*]
- resource : Resource [1..*]

B.5.4 CapabilityRequirement

B.5.4.1 Extension

B.5.4.2 Generalizations

B.5.4.3 Description

A time-dependent requirement for a Capability.

Note 1: The purpose of this being time-dependent is to provide a class to which required metrics for a Capability may be assigned for a given period of time, represented as MeasurableProperties (via the capabilityMetrics attribute).

Note 2: that some capabilities might be called "capability functions" - MODAF does not distinguish between capability functions and capabilities, other than by virtue of their position in a hierarchy, defined by CapabilityComposition

Discussion:

A Capability (A high level specification of the enterprise's ability) can exist without a context of Nodes or Systems. It is simply a requirement of the ability to achieve an effect.

When Nodes and or systems are conceived to realize the capability, then we get the OperationalCapability that has the Node and time dependent requirements.

Thus we define the CapabilityRequirement in terms of the OperationalCapability -i.e., the Enduring Task.

This is a distinction not made by MODAF.

B.5.4.4 Attributes

B.5.4.5 Associations

- capability : OperationalCapabilityRealization [1]
- milestone : Milestone [1]
- node : OperationalNode [1]
- requirement : Requirement [1]
- timeperiod : TimePeriod [1]

B.5.5 Effect Class

B.5.5.1 Extension

B.5.5.2 Generalizations

B.5.5.3 Description

An action that brings about change in the behavior or state of something.

B.5.5.4 Attributes

B.5.5.5 Associations

- causingOperation : Exchange [1]
- impactedResource : Resource [*]
- operationalactivity : OperationalActivity [1..*]
- operationalcapability : OperationalCapabilityRealization [1..*]

B.6 SystemsView Package

B.6.1 Package Diagrams

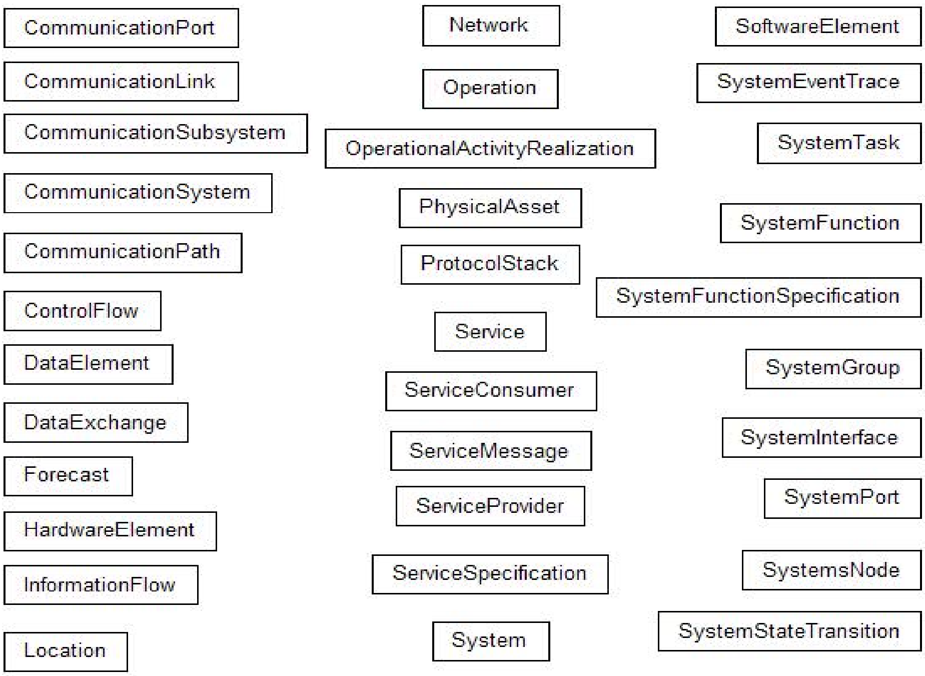


Figure B.15 - SystemsView Main Diagram

B.6.2 Action

B.6.2.1 Extension

B.6.2.2 Generalizations

B.6.2.3 Description

A step in the description of an activity.

B.6.2.4 Attributes

B.6.2.5 Associations

- dataexchange2 : ControlFlow [*]
- systemcontrolflow : ControlFlow [*]
- systemfunction : SystemFunction [1]
- systemtask : SystemTask [1]

B.6.3 CommunicationLink

B.6.3.1 Extension

B.6.3.2 Generalizations

B.6.3.3 Description

A communications link is a single physical connection from one system (or node) to another. A communications path is a (connected) sequence of communications systems and communications links originating from one system (or node) and terminating at another systems (or node).

B.6.3.4 Attributes

B.6.3.5 Associations

- communicationport : CommunicationPort [0..1]
- communicationport2 : CommunicationPort [0..1]

B.6.4 CommunicationPath

B.6.4.1 Extension

B.6.4.2 Generalizations

B.6.4.3 Description

Source: T1/DoDAF

A communications path is a (connected) sequence of communications systems and communications links originating from one system (or node) and terminating at another systems (or node). A communications network may contain multiple paths between the same pair of systems.

B.6.4.4 Attributes

B.6.4.5 Associations

- communicationsystem : CommunicationSystem [1..*]
- comNet : Network [1]
- consumer : System [1]
- producer : System [1]
- systeminterface : SystemInterface [1]

B.6.5 CommunicationPort

B.6.5.1 Extension

B.6.5.2 Generalizations

B.6.5.3 Description

CommunicationPort - Asserts that a connection exists between two ports belonging to parts in a system composite structure model, a.k.a CommunicationLink

B.6.5.4 Attributes

B.6.5.5 Associations

- communicationlink : CommunicationLink [*]
- communicationlink2 : CommunicationLink [*]
- protocolstack : ProtocolStack [1]

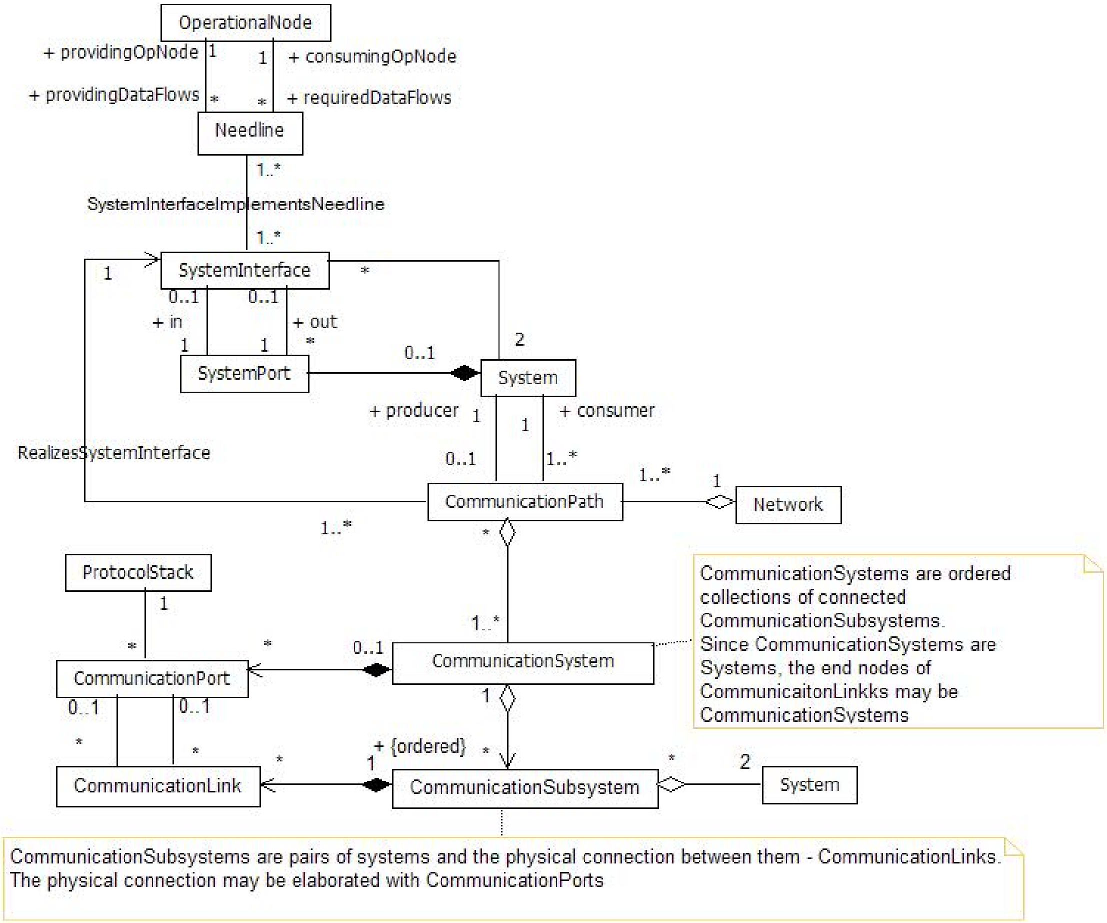
B.6.6 CommunicationSubsystem

B.6.6.1 Extension

B.6.6.2 Generalizations

B.6.6.3 Description

A link - a single physical connection - and the two systems at either end that make up one hop of a CommunicationPath. .



CommunicationSystems are ordered collections of connected CommunicationSubsystems. Since CommunicationSystems are Systems, the end nodes of CommunicaitonLinkks may be CommunicationSystems

CommunicationSubsystems are pairs of systems and the physical connection between them - CommunicationLinks. The physical connection may be elaborated with CommunicationPorts

Figure B.16 - CommunicationSystemDiagram

B.6.6.4 Attributes

B.6.6.5 Associations

- communicationlink : CommunicationLink [*]
- system : System [2..2]

B.6.7 CommunicationSystem

B.6.7.1 Extension

B.6.7.2 Generalizations

- System (from SystemsView)

B.6.7.3 Description

Source: T1/DoDAF

Communications systems (e.g., switches, routers, and communications satellites) are systems whose primary function is to control the transfer and movement of system data as opposed to performing mission application processing.

Source: UST/MODAF

A CommunicationsSystem is a System whose primary function is to control the transfer and movement of system data as opposed to performing mission application processing. Examples include switches, routers, and communications satellites.

B.6.7.4 Attributes

B.6.7.5 Associations

- {ordered} : CommunicationSubsystem [*]
- communicationpath : CommunicationPath [*]
- systemlink : CommunicationPort [*]

B.6.8 ControlFlow

B.6.8.1 Extension

B.6.8.2 Generalizations

B.6.8.3 Description

The flow from one action to another in the description of a system function

B.6.8.4 Attributes

B.6.8.5 Associations

- action : Action [1]
- requestingSystemFunction : Action [1]

B.6.9 DataElement

B.6.9.1 Extension

B.6.9.2 Generalizations

B.6.9.3 Description

B.6.9.4 Attributes

- Accuracy : String [1]

B.6.9.5 Associations

B.6.10 DataExchange

B.6.10.1 Extension

B.6.10.2 Generalizations

B.6.10.3 Description

Provides details of system data elements being exchanged between systems and the attributes of that exchange.

- Interface Identifier
- Data Exchange Identifier
- Data Description
- Data Element Name and Identifier
- Content
- Format Type
- Media Type
- Accuracy
- Units of Measurement
- Data Standard
- Producer
- Sending System Name and Identifier
- Sending System Function Name and Identifier
- Consumer
- Receiving System Name and Identifier
- Receiving System Function Name and Identifier
- Nature of Transaction
- Transaction Type
- Triggering Event
- Interoperability Level Achieved
- Criticality
- Performance Attributes
- Periodicity
- Timeliness
- Throughput
- Size
- Information Assurance
- Access Control
- Availability

- Confidentiality
- Dissemination Control
- Integrity
- Non-Repudiation Producer
- Non-Repudiation Consumer
- Security
- Protection (Type Name, Duration, Date)
- Classification
- Classification Caveat
- Releasability
- Security Standard

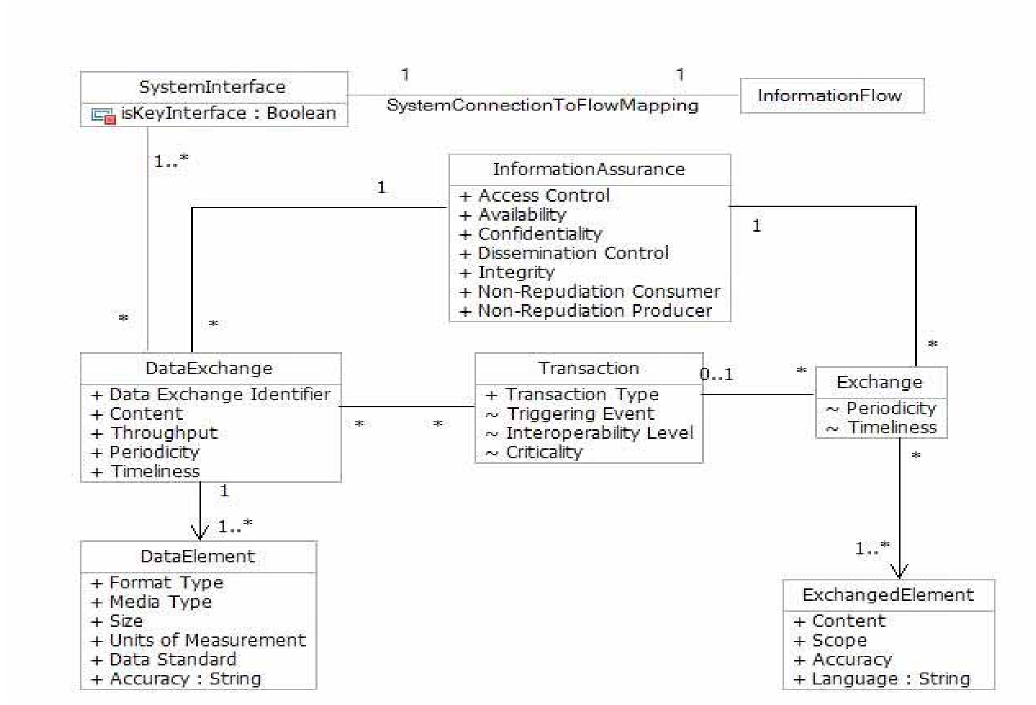


Figure B.17 - DataExchange

B.6.10.4 Attributes

B.6.10.5 Associations

- action : ControlFlow [*]
- date element : DataElement [1..*]
- information assurance : InformationAssurance [1]
- system interface : SystemInterface [1..*]
- systeminformationflow : InformationFlow [*]

- systemtask : SystemTask [*]
- transaction : Transaction [*]

B.6.11 Forecast Class

B.6.11.1 Extension

B.6.11.2 Generalizations

B.6.11.3 Description

Describes the actual or predicted status of a System at a Project Milestone - i.e., a point in the lifecycle of the system.

A statement about the future state of one or more types of system or standard.

Note: this is an EffectivityConstrainedItem, i.e., the forecast is effective for a given period

B.6.11.4 Attributes

B.6.11.5 Associations

- milestone : Milestone [1..*]
- standard : Standard [*]
- system group : SystemGroup [1..*]
- techstandardprofile : TechStandardProfile [*]
- time period : TimePeriod [1]

B.6.12 HardwareElement

B.6.12.1 Extension

B.6.12.2 Generalizations

B.6.12.3 Description

B.6.12.4 Attributes

B.6.12.5 Associations

- performance parameter set : Performance Parameter Set [*]
- systemgroup : SystemGroup [1]
- systemsNode : SystemsNode [1]

B.6.13 InformationFlow

B.6.13.1 Extension

B.6.13.2 Generalizations

B.6.13.3 Description

An ObjectFlow between SystemActivities.

B.6.13.4 Attributes

B.6.13.5 Associations

- dataelement : DataElement [1..*]
- systeminterface : SystemInterface [1]

B.6.14 Location

B.6.14.1 Extension

B.6.14.2 Generalizations

B.6.14.3 Description

Where appropriate, system or physical nodes that constitute the location of an operational node may augment the description of an OperationalNode. These are often taken as recommendations or boundaries for further SV details.

B.6.14.4 Attributes

B.6.14.5 Associations

- system node : SystemsNode [*]

B.6.15 Network

B.6.15.1 Extension

B.6.15.2 Generalizations

- System (from SystemsView)

B.6.15.3 Description

Source: T1/DoDAF

Network - A communications network may contain multiple paths between the same pair of systems. The term interface used in SV-1 and referenced in SV-2 represents an abstraction of one or more communications path(s)

Source: UST/MODAF

A Network is a System that may contain multiple communication paths between the same pair of Systems. This is typically realized by the Network owning a set of connected SystemUsages that are described in an SV-2 internal block diagram.

B.6.15.4 Attributes

B.6.15.5 Associations

- communication path : CommunicationPath [1..*]
- system node : SystemsNode [1..*]

B.6.16 Operation

B.6.16.1 Extension

B.6.16.2 Generalizations

B.6.16.3 Description

The operations that are available to the consumer of the Service.

B.6.16.4 Attributes

B.6.16.5 Associations

- serviceMessages : ServiceMessage [*]

B.6.17 OperationalActivityRealization

B.6.17.1 Extension

B.6.17.2 Generalizations

B.6.17.3 Description

A collaboration of systems that cooperate to achieve a high level system activity. By linking an OperationalActivityRealization to an OperationalActivity, the subactivites and steps that define the behavior of OperationalActivityRealization can be shown to realize the OperationalActivity to which it is linked.

B.6.17.4 Attributes

B.6.17.5 Associations

- operational activity : OperationalActivity [*]
- system : System [*]
- systemeventtrace : SystemEventTrace [1]
- systemfunction : SystemFunction [*]

B.6.18 PhysicalAsset

B.6.18.1 Extension

B.6.18.2 Generalizations

B.6.18.3 Description

AKA SystemsNode, the collection of artifacts and resources that constitute a deployment of one or more systems to affect an OpeationalNode.

B.6.18.4 Attributes

B.6.18.5 Associations

B.6.19 ProtocolStack

B.6.19.1 Extension

B.6.19.2 Generalizations

B.6.19.3 Description

B.6.19.4 Attributes

B.6.19.5 Associations

- systemport : CommunicationPort [*]

B.6.20 Service

B.6.20.1 Extension

B.6.20.2 Generalizations

B.6.20.3 Description

The service model element provides the end-point for service interaction (in web service terminology), whereas the definition of these interactions is a part of the service specification stereotype. In your model, a service not only identifies the provided interface, but may also identify required interfaces (such as callback interfaces).

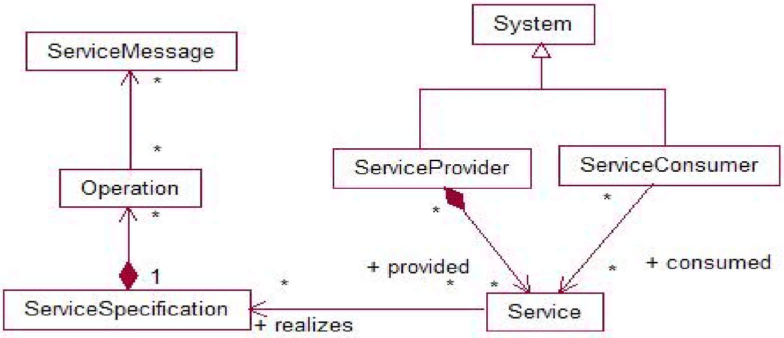


Figure B.18 - Service

B.6.20.4 Attributes

B.6.20.5 Associations

- realizes : ServiceSpecification [*]

B.6.21 ServiceConsumer

B.6.21.1 Extension

B.6.21.2 Generalizations

- System (from SystemsView)

B.6.21.3 Description

Any classifier (class, component, and so on) may act as the consumer of a service, and that includes another service. While this stereotype is most definitely optional, it may help you identify elements of a model -- that are not services themselves -- as clients of services. On the other hand, it may just be overhead and you don't need to use it.

B.6.21.4 Attributes

B.6.21.5 Associations

- consumed : Service [*]
- required NodeInterfaces : OperationalNodeSpecification [1..*]

B.6.22 ServiceMessage

B.6.22.1 Extension

B.6.22.2 Generalizations

B.6.22.3 Description

A serviceMessage is a container for actual data that has meaning to the service and the consumer. A serviceMessage may not have operations, but it may have properties and associations to other classes (likely of some domain model). A message stereotype has a property to denote its assumed encoding form (for example, SOAP-literal, SOAP-rpc, ASN.1, and so on).

B.6.22.4 Attributes

B.6.22.5 Associations

B.6.23 ServiceProvider

B.6.23.1 Extension

B.6.23.2 Generalizations

- System (from SystemsView)

B.6.23.3 Description

The service provider is a software element that provides one or more services. A service provider has a property that captures information about its location, although the meaning of this is implementation-dependent. The class acting as the service provider may not expose any attributes or operations directly: only public ports may be provided (stereotyped as service), and these are typed by service specifications.

B.6.23.4 Attributes

B.6.23.5 Associations

- provided : Service [*]
- providedNodeInterfaces : OperationalNodeSpecification [1..*]

B.6.24 ServiceSpecification

B.6.24.1 Extension

B.6.24.2 Generalizations

- SystemFunctionSpecification (from SystemsView)

B.6.24.3 Description

The external identity of a service by which it can be requested for consumption.

B.6.24.4 Attributes

B.6.24.5 Associations

- operations : Operation [*]

B.6.25 SoftwareElement

B.6.25.1 Extension

B.6.25.2 Generalizations

B.6.25.3 Description

B.6.25.4 Attributes

B.6.25.5 Associations

- system : System [*]
- systemGroup : SystemGroup [1]
- systemsNode : SystemsNode [1]

B.6.26 System

B.6.26.1 Extension

B.6.26.2 Generalizations

B.6.26.3 Description

Systems consisting of family of systems (FoS), system of systems (SoS), networks of systems, individual systems, and items (e.g., equipment hardware and software).

The taxonomy minimally consists of names, descriptions, breakdowns into the constituent parts of the system and categorization of types of systems. Typing may also address variations across time and systems node installation.

Any organized assembly of resources and procedures united and regulated by interaction or interdependence to accomplish a set of specific functions.

(UPDM)

SV-1 links together the OV and SV by depicting the assignments of systems and systems nodes (and their associated interfaces) to the operational nodes (and their associated Needlines) described in OV-2. OV-2 depicts the operational nodes representing organizations, organization types, and/or human roles, while SV-1 depicts the systems nodes that house operational nodes (e.g., platforms, units, facilities, and locations) and the corresponding systems resident at these systems nodes and which support the operational 5-2 nodes.

A coherent combination of physical artefacts, energy and information, assembled for a purpose. NOTE: from v0.96 of this model, the PhysicalAsset stereotype was introduced

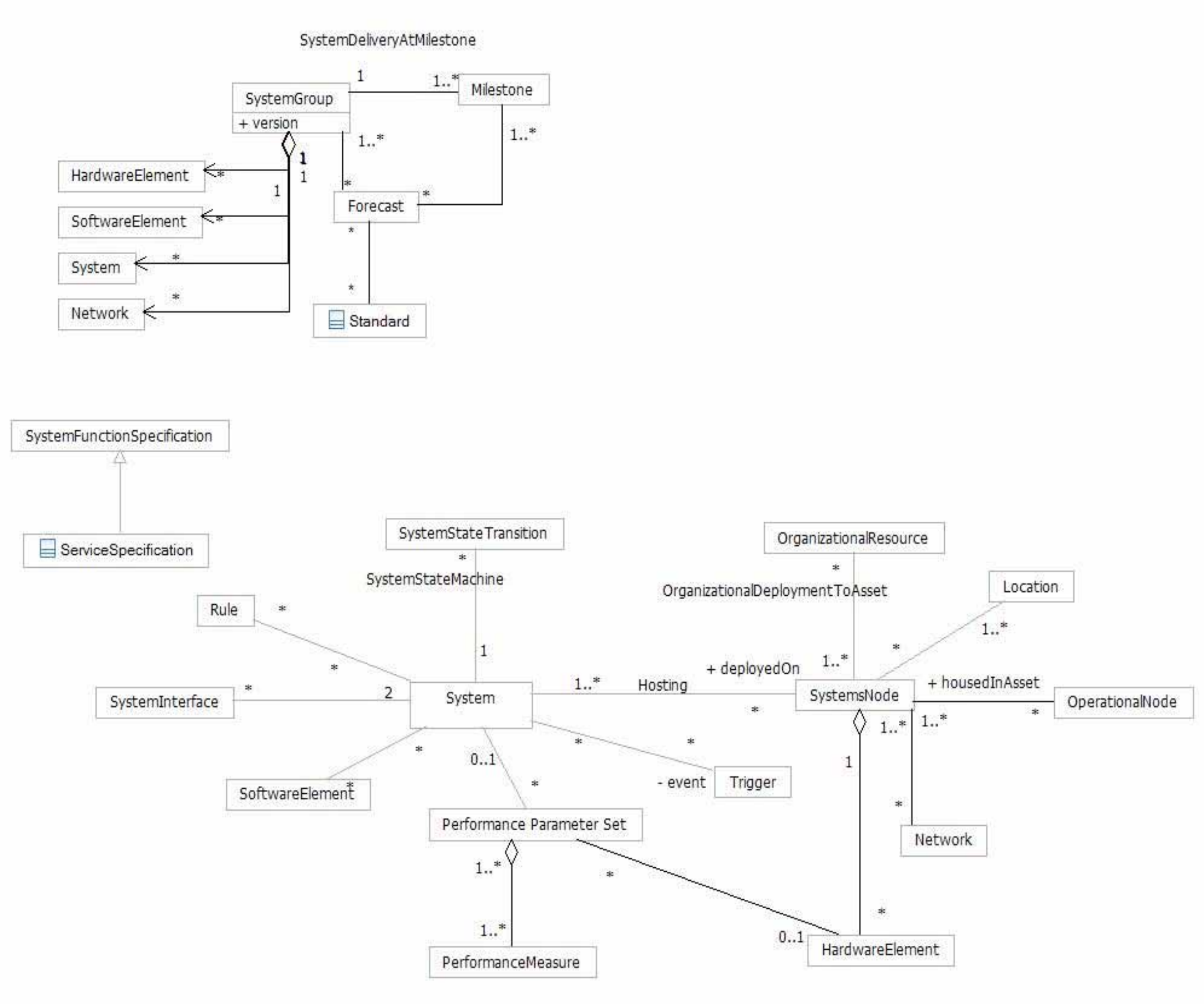


Figure B.19 - SystemGroup and System

B.6.26.4 Attributes

B.6.26.5 Associations

- communicationlink : CommunicationSubsystem [*]
- deployedOn : SystemsNode [*]
- event : Trigger [*]
- fromConnectionPath : CommunicationPath [1..*]
- operationalactivityrealization : OperationalActivityRealization [*]
- performance parameter set : Performance Parameter Set [*]

- rule : Rule [*]
- state transition : SystemStateTransition [*]
- system interface : SystemInterface [*]
- systemactivityfunction : SystemTask [*]
- systemeventtrace : SystemEventTrace [*]
- systemfunction : SystemFunction [*]
- systemPort : SystemPort [*]
- systemSoftware : SoftwareElement [*]
- toConnectionPath : CommunicationPath [0..1]

B.6.27 SystemEventTrace

B.6.27.1 Extension

B.6.27.2 Generalizations

B.6.27.3 Description

The ordered sequence of events that trigger exchanges among systems.

B.6.27.4 Attributes

B.6.27.5 Associations

- operationalactivityrealization : OperationalActivityRealization [*]
- system : System [*]
- systemtask : SystemTask [*]

B.6.28 SystemFunction

B.6.28.1 Extension

B.6.28.2 Generalizations

B.6.28.3 Description

The Activities that a System performs.

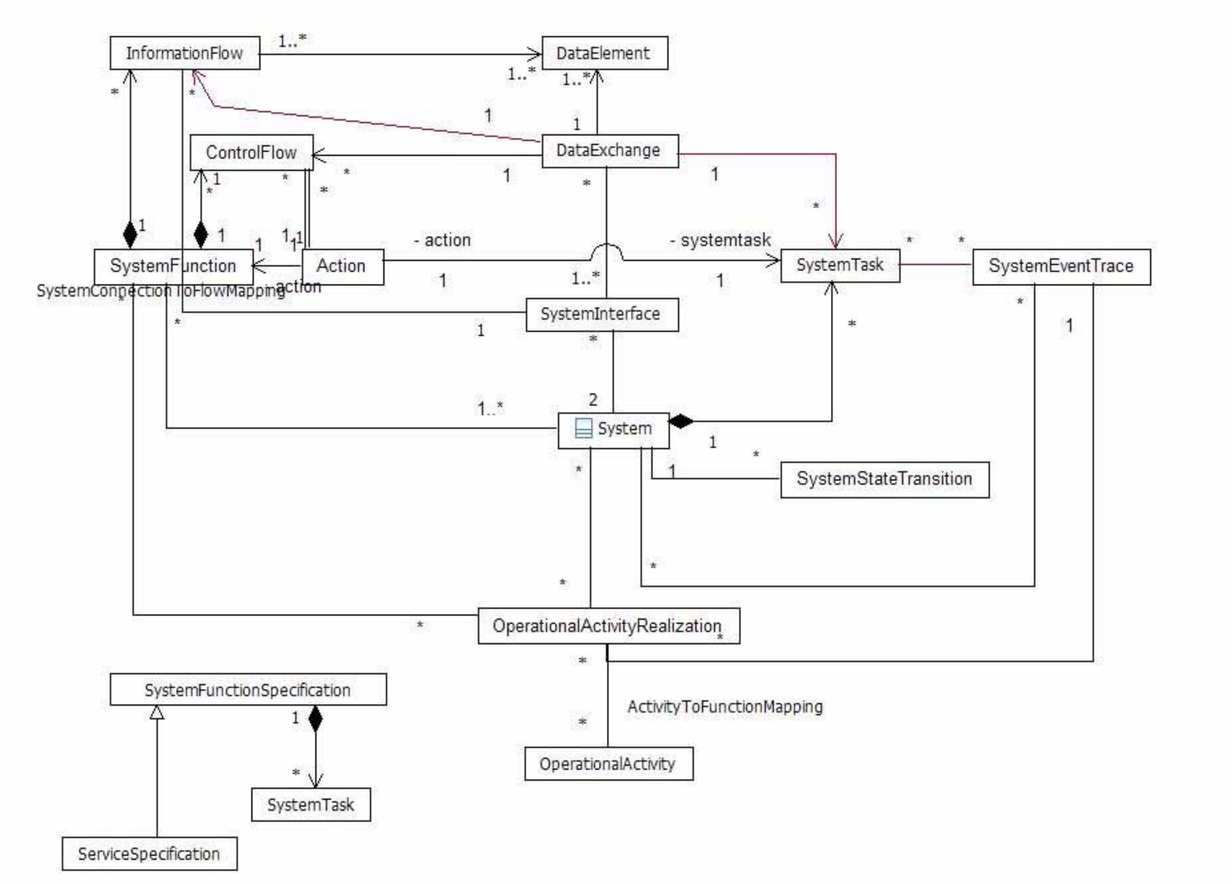


Figure B.20 - SystemFunction

B.6.28.4 Attributes

B.6.28.5 Associations

- dataexchange : InformationFlow [*]
- operationalactivityrealization : OperationalActivityRealization [*]
- operationalcapabilityrealization : OperationalCapabilityRealization [*]
- system : System [1..*]
- systemcontrolflow : ControlFlow [*]

B.6.29 SystemFunctionSpecification Class

B.6.29.1 Extension

B.6.29.2 Generalizations

B.6.29.3 Description

System Functions

The taxonomy minimally consists of names, descriptions, and decomposition into the constituent parts that comprise a system function.

A data transform that supports the automation of activities or information elements exchange. (DoDAF)

System functions are not limited to internal system functions and can include Human Computer Interface (HCI) and Graphical User Interface (GUI) functions or functions that consume or produce system data from/to system functions that belong to external systems. The external system data sources and/or sinks can be used to represent the human that interacts with the system or external systems. The system data flows between the external system data source/sink (representing the human or system) and the HCI, GUI, or interface function can be used to represent human-system interactions, or system-system interfaces. Standards that apply to system functions, such as HCI and GUI standards, are also specified in this product.

The relationship between operational activities and system functions can also be expected to be many-to- many (i.e., one operational activity may be supported by multiple system functions, and one system function may support multiple operational activities). Figure 5-22 provides a notional example of SV-5.

First, the activities are related to the system functions that automate them (wholly or partially). Then a set of activities are associated with a capability (as defined in OV-6c). By labeling the system functions with the systems that execute them (as defined in SV-1, SV-2, and SV-4),

System Function of a System Supports Operational Activity for a Capability.

System Function Implements Operational Activity*

B.6.29.4 Attributes

B.6.29.5 Associations

- systemactivityfunction : SystemTask [*]

B.6.30 SystemGroup

B.6.30.1 Extension

B.6.30.2 Generalizations

B.6.30.3 Description

Product Definition. The Systems Evolution Description 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.

Product Purpose. SV-8, when linked together with other evolution products such as SV-9 and TV-2, provides a clear definition of how the architecture and its systems are expected to evolve over time. In this manner, the product can be used as an architecture evolution project plan or transition plan, much of the desired purpose can be achieved by defining sets of UML products for different periods of system evolution, thus allowing a comparison between time periods to highlight the evolutionary changes. Textual descriptions of the evolution steps should be included.

System Group*

Name*	Name/identifier for a set of systems, subsystems, or system hardware/software items.
Description*	Textual description of system group
System*	See SV-1 Definition Table
System Hardware/Software Item	See SV-1 Definition Table
Communications System	See SV-2 Definition Table
Communications Link	See SV-2 Definition Table
Communications Network	See SV-2 Definition Table
Time Period* Identifier*	Identifier for the time period
Date*	A specific date in time

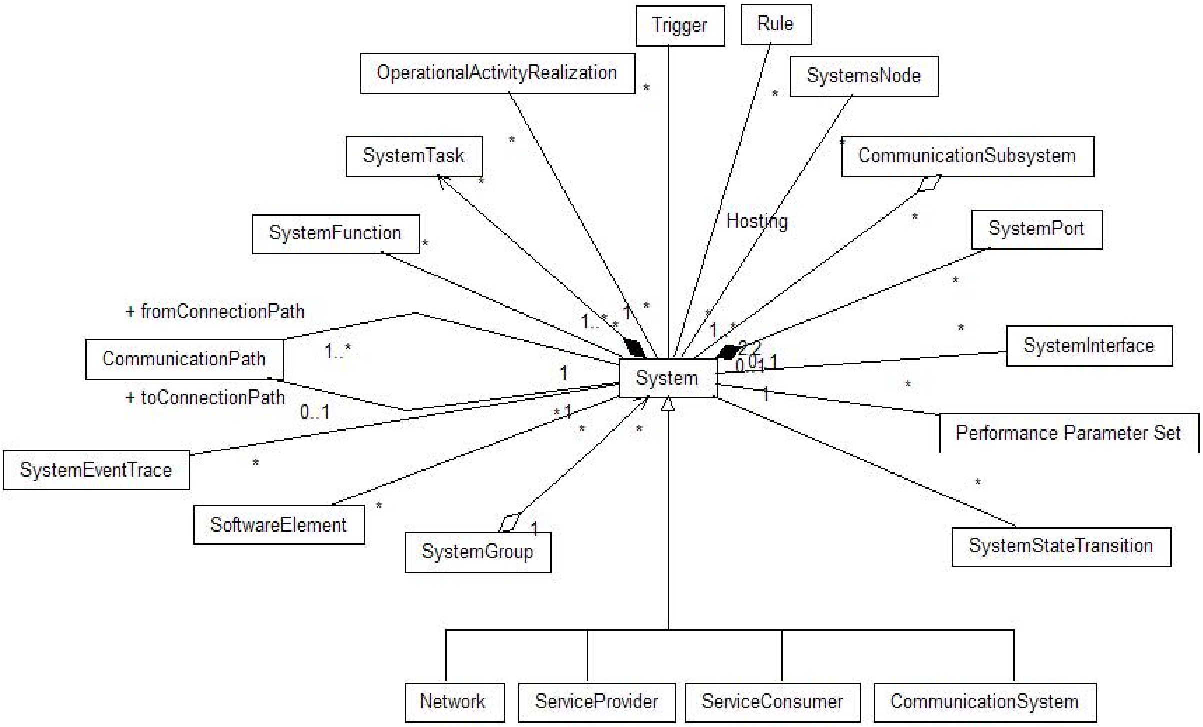


Figure B.21 - SystemGroup

B.6.30.4 Attributes

B.6.30.5 Associations

- communication network : Network [*]
- milestone : Milestone [1..*]
- system : System [*]
- system software : SoftwareElement [*]
- systemHardware : HardwareElement [*]
- ‘technology forecast : Forecast [*]

B.6.31 SystemInterface

B.6.31.1 Extension

B.6.31.2 Generalizations

B.6.31.3 Description

Source: T1/DoDAF

An interface, as depicted in SV-1, is a simplified, abstract representation of one or more communications paths between systems nodes or between systems (including communications systems) and is usually depicted graphically as a straight line. SV-1 depicts all interfaces that are of interest for the architecture purpose.

An SV-1 interface is the systems representation of an OV-2 Needline. A single Needline shown in the OV may translate into multiple system interfaces.

The actual implementation of an interface may take more than one form (e.g., multiple physical links). Details of the physical links and communications networks that implement the interfaces are documented in SV-2.

Characteristics of the interface are described in Systems-Systems Matrix (SV-3). System functions and system data flows are documented in a Systems Functionality Description (SV-4), and the system data carried by an interface are documented in the Systems Data Exchange Matrix (SV-6).

waveform, bandwidth, radio frequency, and packet or waveform encryption methods

B.6.31.4 Attributes

- isKeyInterface : Boolean [1]

B.6.31.5 Associations

- data flow : DataExchange [*]
- in : SystemPort [1]
- needline : Needline [1..*]
- out : SystemPort [1]

- system : System [2..2]
- systemfunctionflow : InformationFlow [1]

B.6.32 SystemPort

B.6.32.1 Extension

B.6.32.2 Generalizations

- HardwareElement (from SystemsView)

B.6.32.3 Description

Mechanism to allow for connecting two systems.

Represents the system hardware that implements the connection of two systems

An interface (logical or physical) provided by a System. SystemPort may implement a PortType, though there is no requirement for SystemPorts to be typed

B.6.32.4 Attributes

B.6.32.5 Associations

- system : System [0..1]
- systemlink : SystemInterface [0..1]
- systemlink2 : SystemInterface [0..1]

B.6.33 SystemsNode

B.6.33.1 Extension

B.6.33.2 Generalizations

- Resource (from AllViews)

B.6.33.3 Description

Source: T1/DoDAF

Systems Nodes that represent facilities, platforms, units, and locations. The taxonomy minimally consists of names, descriptions, breakdowns into constituent parts of the node, and categorizations of types of facilities, platforms, units, and locations.

A node with the identification and allocation of resources (e.g., platforms, units, facilities, and locations) required to implement specific roles and missions. (UPDM)

A class of physical object that can host systems and/or people.

A SystemContext is a construct that is used in those cases when there is no enclosing composite SystemsNode)i.e., PhysicalAsset). The (somewhat artificial) top level construct of SystemContext then acts as the composite.

SV-1 Product Definition. The Systems Interface Description depicts systems nodes and the systems resident at these nodes to support organizations/human roles represented by operational nodes of the Operational Node Connectivity Description (OV-2). SV-1 also identifies the interfaces between systems and systems nodes.

B.6.33.4 Attributes

B.6.33.5 Associations

- communication network : Network [*]
- hardware : HardwareElement [*]
- location : Location [1..*]
- operationalNodes : OperationalNode [*]
- organizationalResource : OrganizationalResource [*]
- system : System [1..*]

B.6.34 SystemStateTransition

B.6.34.1 Extension

B.6.34.2 Generalizations

B.6.34.3 Description

Three types of models may be used to adequately describe the dynamic behavior and performance characteristics of a SV. These three models are:

- Systems Rules Model (SV-10a)
- Systems State Transition Description (SV-10b)
- Systems Event-Trace Description (SV-10c)

The State Transition Description 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.

B.6.34.4 Attributes

B.6.34.5 Associations

- system : System [1]

B.6.35 SystemTask Class

B.6.35.1 Extension

B.6.35.2 Generalizations

B.6.35.3 Description

SystemActivityFunction (or Tasks). The taxonomy minimally consists of names, descriptions, and decomposition into the constituent parts that comprise a process-activity.

SystemActivityFunction defined and standardized by the Joint Staff are in the form of Mission Essential Tasks [CJCSM 3500.04C, 2002].

Operational Activities are also specified (and sometimes standardized) in the form of process activities arising from process modeling. It is sometimes convenient to merge these sets, either as activities or tasks.

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 (e.g., 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 Operational Activity Model). It is used to portray operational actions not hardware/software system functions.

SystemActivityFunction. The system operations (from the OV-5 Operational Activity Model) performed by a given system.

B.6.35.4 Attributes

B.6.35.5 Associations

- systemeventtrace : SystemEventTrace [*]

Annex C
Usage Example

(non-normative)

The UPDM Proof of Concept (POC) has been updated significantly in the last two years. The most notable changes were in the August 2006 and February 2007 time frames as a direct results of submission team merging activities. The POC and associated Example were changed accordingly during 2005 and 2006. The Example was not updated for this final submission merge activity requested by the OMG Architecture Board. The Example is presented here as the basis for expected future publications and tutorials on UPDM modeling.

C.1 Overview

The following set of DoDAF views show how a domain architecture would be represented using the UPDM. The text and diagrams were collected and represented in the IBM Rational DoDAF Modeling plug-in for IBM Rational Software Architect version 6.0.1.1. The document itself was largely generated directly from an model in the tool.

Note: Formatting anomalies in the example below reflect the actual output from the prototype extensions to the IBM Rational DoDAF Modeling plug-in for RSA. Sections such as the Table of Contents do not refer to actual pagination in the submission, but in the relative pagination of the embedded report. The output from the tool was only formatted for visibility and clarity.

C.2 Example

The remainder of this section is a report generated by the POC based on the 1st and 2nd UPDM submissions. Please disregard what appears to be formatting errors as they are part of the report, not this specification.

C.2.1 DoDA F Report

Table of Contents

1. Report Source

This report was created from Model: UPDM Example (JFACC planning).

Part1: All Views

2. AV-1: Overview and Summary Information

2.1 Architecture Project Identification

2.1.1 Name

Joint Force Air Component Commander (JFACC) Planning System

2.1.2 Project Description

The JFACC Planning System describes the mission planning system used by the JFACC to coordinate air operations mission tasking across the Joint Task Force Theater of Operations. The DoD military information contained in this architecture has been extracted from open source information, primarily from Joint Publication 3-01 "Joint Doctrine for Countering Air and Missile Threats," 19 October 1999.

2.1.3 Architect Name

UPDM Example Team

2.1.4 Developing Organization

UML Profile for DoDAF/MODAF (UPDM) Proposal Team ONE

2.1.5 Assumptions and Constraints

Caveat: The architecture data in this example is intended for demonstration purposes only. The authors make no claim regarding its accuracy or completeness.

2.1.6 Approval Authority

Object Management Group Architecture Board

2.1.7 Date Completed

November 11, 2006

2.1.8 Level of Effort and Projected and Actual Costs to Develop the Architecture

N/A

2.2 Scope: Architectural Views and Products Identification

The architecture scope is defined as sufficient to meet OMG requirements for UPDM Profile submission.

2.2.1 Architecture Name

JFACC Planning System Architecture

2.2.2 Architecture Description

2.2.3 Views and Products Developed

2.2.3.1 Views

The architecture is a description of the structure and behavior of the JFACC Planning System. The architecture is manifested as an integrated database from which all views prescribed in DoDAF and MODAF instructions can be generated

2.2.3.2 Products

The architecture data file can be exported in XMI format. Architecture reports of selected views could be generated from the data file.

2.2.4 Time Frames Addressed

This architecture describes the JFACC Planning System as described in the current version of Joint Publication 3-01.

2.2.5 Organizations Involved

UPDM Team One.

2.3 Purpose and Viewpoint

2.3.1 Purpose, Analysis, Questions to be answered by Analysis of the Architecture

The JFACC Planning System architecture is intended to display the UPDM Team One profile proposal to meet OMG submission requirements.

2.3.2 From whose Viewpoint the Architecture is developed

The JFACC Planning System architecture has been developed from the viewpoint of operational planners, system owners, system designers, and product builders.

2.4 Context

The Joint Force Commander (JFC) may delegate responsibilities for joint air operations to the JFACC for the joint forces. When assigned, the JFACC plans and manages the execution of all air operations for the joint force. During planning, the JFACC must collaborate with supporting and supported commanders. These include the JFC and the area air defense commander (AADC) community of interest (COI). The AADC COI consists of AADC air defense planning nodes and air defense execution nodes. Among these nodes is the AADC. The JFACC planning systems should support this collaboration and planning.

2.4.1 Mission

2.4.2 Doctrine, Goals, and Vision

Air operations usually begin early in the conduct of joint operations, and their effects produce a desired degree of air superiority at the time and place of the JFACC's choosing. Air operations are conducted to protect the joint force and eliminate opposition. The degree of success in conducting air operations relies on proper planning within the JFACC and other organizations.

Air operations doctrine that supports the JFACC is outlined in Joint Publication 3-01, "Joint Doctrine for Countering Air and Missile Threats.

2.4.3 Rules, Criteria, and Conventions Followed

Architecture conventions are taken from the Team ONE UML 2.0 Profile for DoDAF/MODAF. Operational activities and system functions are not real, but are derived from joint publications found in the DoD Standards Registry. The following references were used to provide a limited context for this architecture. All of these references are available on the internet from the URL listed.

Joint Doctrine Joint Force Employment briefing

http://www.dtic.mil/doctrine/jrm/plans.pdf

Joint Forces Staff College, Joint Planning and Operations Course

http://www.jfsc.ndu.edu/schools_programs/jpoc/course_materials/default.asp

Doctrine for Joint Operations

http://www.dtic.mil/doctrine/jel/new_pubs/jp3_0.pdf

Joint Doctrine for countering air and missile threats

http://www.dtic.mil/doctrine/jel/new_pubs/jp3_01.pdf

Joint Communications System

http://www.dtic.mil/doctrine/jel/new_pubs/jp6_0.pdf

DoD dictionary of military terms

http://www.dtic.mil/doctrine/jel/doddict/index.html

Joint Staff officers guide, 2000

http://www.jfsc.ndu.edu/current_students/documents_policies/documents/jsogpub_1_2000.pdf

2.4.4 Tasking for Architecture Project and Linkages to Other Architectures

Tasking for this architecture can be found in: UML Profile for DoDAF/MODAF, September 16, 2005, Request for Proposal (OMG Document: c4i/05-09-12)

2.5 Tools and File Formats Used

The JFACC Planning System Architecture is maintained using IBM Rational System Architect (RSA), Version 6.0. Architecture export is accomplished in XML Metadata Interchange (XMI) format. All architecture views are maintained in RSA, except for views employing graphical images e.g., OV-1, and URLs for which links from RSA are provided.

2.6 Findings

2.6.1 Analysis Findings

No analysis has been conducted on this example.

2.6.2 Recommendations

No recommendations apply to this example.

3. AV-2: Integrated Dictionary

Table 2 - AV-2 Glossary of elements contained in the architecture model

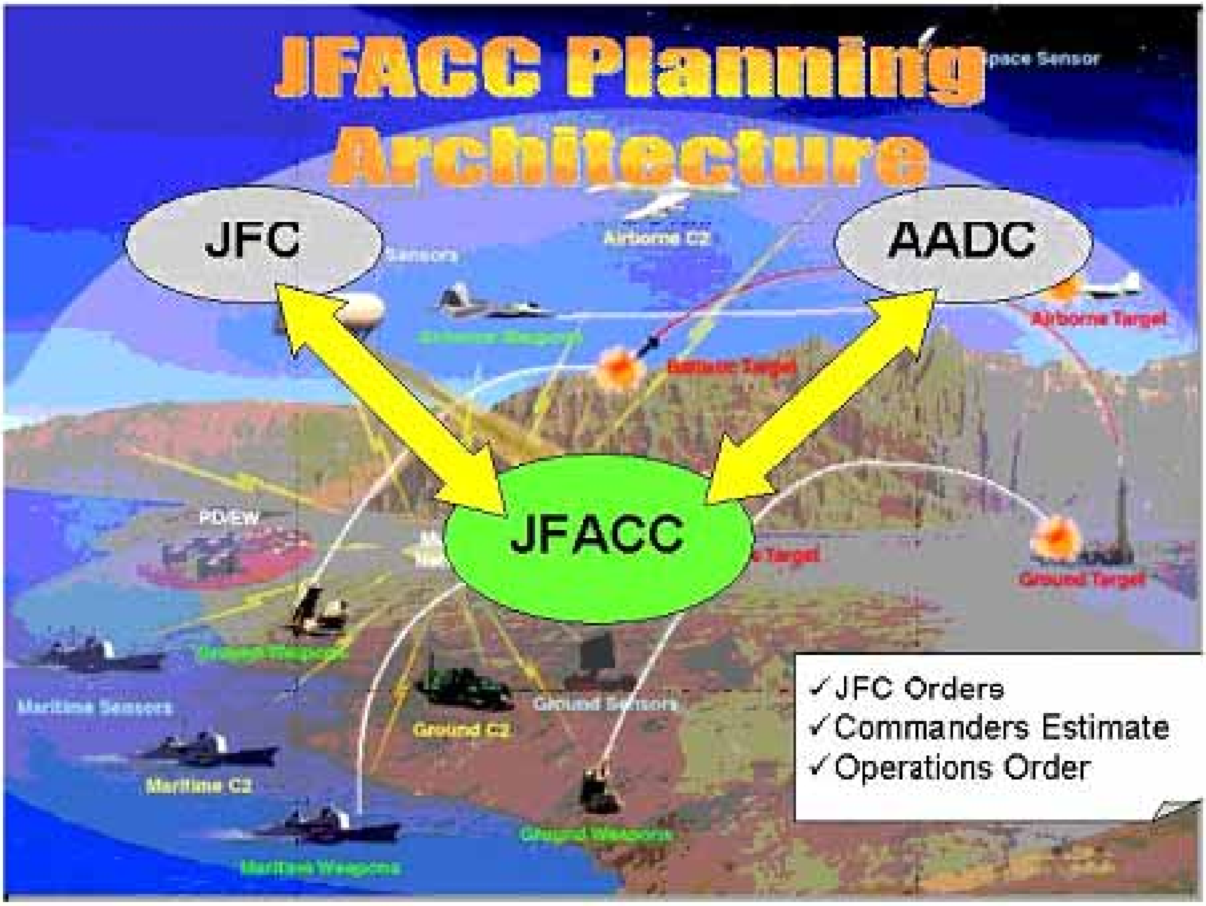
Element	Element Kind	AV-2 Definition
AADC	OperationalNode	Within a unified command, subordinate unified command, or joint task force, the commander will assign overall responsibility for air defense to a single commander. Normally, this will be the component commander with the preponderance of air defense capability and the command, control, and communications capability to plan and execute integrated air defense operations. Representation from the other components involved will be provided, as appropriate, to the area air defense commander's headquarters. Also called AADC. (DoD dictionary)
AADC:JFACC Communications	Needline	The information exchanges between AADC and JFACC
JFACC	OperationalNode	The commander within a unified command, subordinate unified command, or joint task force responsible to the establishing commander for making recommendations on the proper employment of assigned, attached, and/or made available for tasking air forces; planning and coordinating air operations; or accomplishing such operational missions as may be assigned. The joint force air component commander is given the authority necessary to accomplish missions and tasks assigned by the establishing commander. Also called JFACC. See also joint force commander. (DoD dictionary) Also called JFACC.
JFACC Communications	OperationalNode	The staff component of the JFACC responsible for communications activities (DoD dictionary)
JFACC Communications:AADC	Needline	The information exchanges between JFACC and AADC.
JFACC Communications:JFACC planning	Needline	The information exchanges between staff components of the JFACC. Specifically from the communications section to the planning section

Table 2 - AV-2 Glossary of elements contained in the architecture model

JFACC Intelligence	OperationalNode	The staff component of the JFACC responsible for intelligence activities. (Joint pub 3-01)
JFACC Operations	OperationalNode	The staff component of the JFACC responsible for operations activities.. (Joint pub 3-01)
JFACC Planning System	System	The system that performs the functions of JFACC planning. This system is comprised of the Commander's Intent Development System and the Operations Order Development System. (Imagined)
JFACC planning	OperationalNode	The staff component of the JFACC responsible for planning activities. (Joint pub 3-01)
JFACC planning:AADC	Needline	The information exchanges between the planning component of the JFACC and the AADC
JFACC planning:JFACC Communications	Needline	The information exchanges between staff components of the JFACC. Specifically from the planning section to the communications section.
JFC	OperationalNode	A general term applied to a combatant commander, sub-unified commander, or joint task force commander authorized to exercise combatant command (command authority) or operational control over a joint force. Also called JFC. (DoD dictionary)
JFC Operations	OperationalNode	The staff component of the JFC responsible for operations activities. (Joint pub 3-01)
JFC Planning	OperationalNode	The staff component of the JFC responsible for planning activities. (Joint pub 3-01)
JFC Planning:AADC	Needline	The information exchanges between the planning component of the JFC and the AADC
JFC Planning:JFACC Communications	Needline	The information exchanges between the planning component of the JFC and the communications component of the JFACC.
Planning Communication System 1	CommunicationSystem	The system that is the primary implementation of communications within the Joint Planning Network.
Planning Communication System 2	CommunicationSystem	The system that is the secondary (backup) implementation of communications within the Joint Planning Network
Planning Network	CommunicationSystem	The combination of communication terminals and adaptation interfaces the provide transport services for the JFC planning activities.

Part 2: Operational Views

4. OV-1: High Level Operation Concept Graphic


JFACC Planning Architecture

✓JFC Orders
✓Commanders Estimate
✓Operations Order

Figure 1 - Overall Operational Context of the Architecture the JFACC and associated nodes within the Joint Area of Operations.

Operational Context Description.

The Joint Force Commander (JFC) typically delegates command and control authority for air operations to a Joint Force Air Component Commander (JFACC). The JFC provides joint air operations planning guidance to the JFACC in accordance with that delegation. The JFACC, in collaboration with the JFC and the AADC, produces a commanders intent and operations orders for air operations as outlined in joint doctrine. These planning products support planning guidance derived from the originating JFC guidance. The JFACC keeps the JFC informed of his commander's estimate, and receives approvals and recommendations of intended courses of action. The JFACC also produces operations orders to provide coordination and focus of effort for associated forces. An Area Air Defense Commander (AADC) community of interest (COI), supports planning for the air defense portion of air operations of the Joint Force Area of Responsibility. The AADC COI includes the AADC, a node that coordinates and directs action of the COI. The AADC is generally assigned to the component commander best equipped to conduct integrated air defense operations. This AADC COI interacts with the JFACC in the development of commander's intent and operations order by providing recommendations.

5. OV-2: Operational Node Connectivity Description

Implementation note:

The UPDM profile enables modelers to provide sufficient information in their models to facilitate auto-generation of the OV-2 product. Start in the OV-6c interactions, find the messages that are stereotyped <<Needline>>.

The OV-2 is a generalization of the intricate relationships expressed by the messages in the OV-6c interaction(s). In order to construct the Needlines for an automated OV-2, the tool builder need only iterate over the messages, find the sending and receiving OperationalNodes.

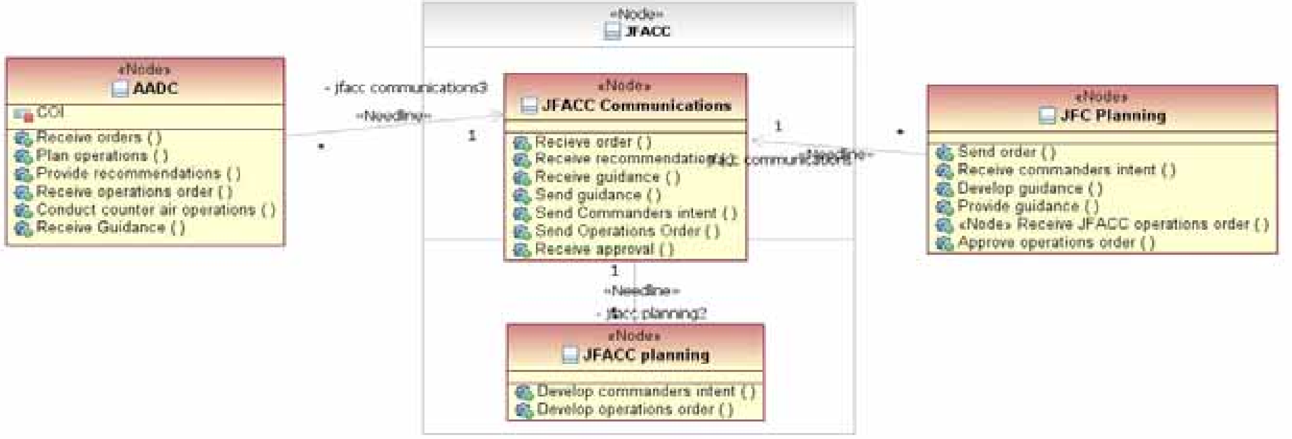


Figure 2 - OV-2 Operational node interactions and needlines based on joint planning doctrine

6. OV-3: Operational Information Exchange Matrix

Table 3 - OV-3 Matrix

Needline	Information Element	Producer Node	Consumer Node	UJTL Task	Transaction Type	Timing	QoS Rqmt	Security Classif. (highest)	Distribution Handling
AADC::JFACC Communications	AADC Recommend-ations	AADC	JFACC Communica-tions	Air Operations Planning	Point to Point	periodic 24 hr	best effort (receipt)	U	US only
JFACC Communications:: AADC	Operations Orders	JFACC Communica-tions	AADC	Air Operations Planning	Force Broadcast	periodic 24 hr	best effort	U	Coalition
JFACC Communications:: AADC	JFACC Guidance	JFACC Communica-tions	AADC	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFACC Communications:: JFACC Planning	AADC Information	JFACC Communica-tions	JFACC planning	Air Operations Planning	Point to Point	on event	best effort (receipt)	U	Coalition
JFACC Communications:: JFACC Planning	Force Orders	JFACC Communica-tions	JFACC planning	Air Operations Planning	Force Broadcast	when issued	best effort	U	Coalition
JFACC Planning:: JFACC Communications	Operations Orders	JFACC Planning	JFACC Communica-tions	Air Operations Planning	Force Broadcast	periodic	best effort	U	Coalition
JFACC Planning:: JFACC Communications	JFACC Guidance	JFACC Planning	JFACC Communica-tions	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition

Table 3 - OV-3 Matrix

JFACC Planning::AADC	JFACC Guidance	JFACC Planning	AADC	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFC Planning:: JFACC Communications	Force Orders	JFC Planning	JFACC Communications	Air Operations Planning	Force Broadcast	when issued	best effort	U	Coalition
JFACC Communications:: JFC Planning	Operations Orders	JFACC Communications	JFC Planning	Air Operations Planning	Force Broadcast	periodic 24 hr	best effort	U	Coalition
JFC Planning::JFACC Planning	Startup (initiate planning operations)	JFC Planning	JFACC Planning	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFC Planning::JFACC Planning	Planning Alert	JFC Planning	JFACC Planning	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFC Planning::JFACC Planning	Cease Planning Alert	JFC Planning	JFACC Planning	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFC Planning::JFACC Planning	Standdown (cease planning operations)	JFC Planning	JFACC Planning	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFC Planning::JFACC Planning	JFC Planning Guidance	JFC Planning	JFACC Planning	Air Operations Planning	Point to Point	when issued	reliable	U	Coalition
JFC Planning::JFACC Planning	Operations Plans	JFC Planning	JFACC Planning	Air Operations Planning	Point to Point	on event	reliable	U	US only
JFACC Planning::JFC Planning	Commander's Intent	JFACC Planning	JFC Planning	Air Operations Planning	Point to Point	on event	reliable	U	US only

OV-3 Matrix reflecting the data flows associated with needlines in the OV-2

7. OV-4: Organizational Relationship Chart

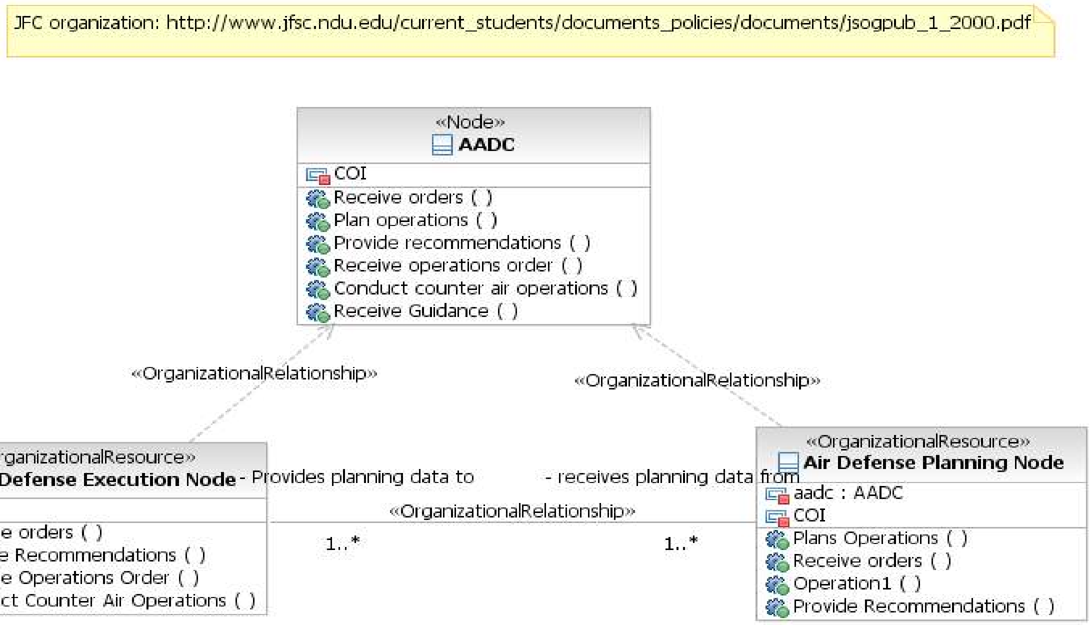

JFC organization: http://www.jfsc.ndu.edu/current_students/documents_policies/documents/jsogpub_1_2000.pdf
«Node»
AADC
COI
Receive orders ( )
Plan operations ( )
Provide recommendations ( )
Receive operations order ( )
Conduct counter air operations ( )
Receive Guidance ( )
«OrganizationalRelationship»
«OrganizationalRelationship»
«OrganizationalResource»
Air Defense Execution Node
COI
Receive orders ( )
Provide Recommendations ( )
Receive Operations Order ( )
Conduct Counter Air Operations ( )
- Provides planning data to
- receives planning data from
«OrganizationalRelationship»
1..*
1..*
«OrganizationalResource»
Air Defense Planning Node
aadc : AADC
COI
Plans Operations ( )
Receive orders ( )
Operation1 ( )
Provide Recommendations ( )

Figure 3 - URL provides overall organization and composite diagram reflects AADC COI

8. OV-5: Operational Activity Model

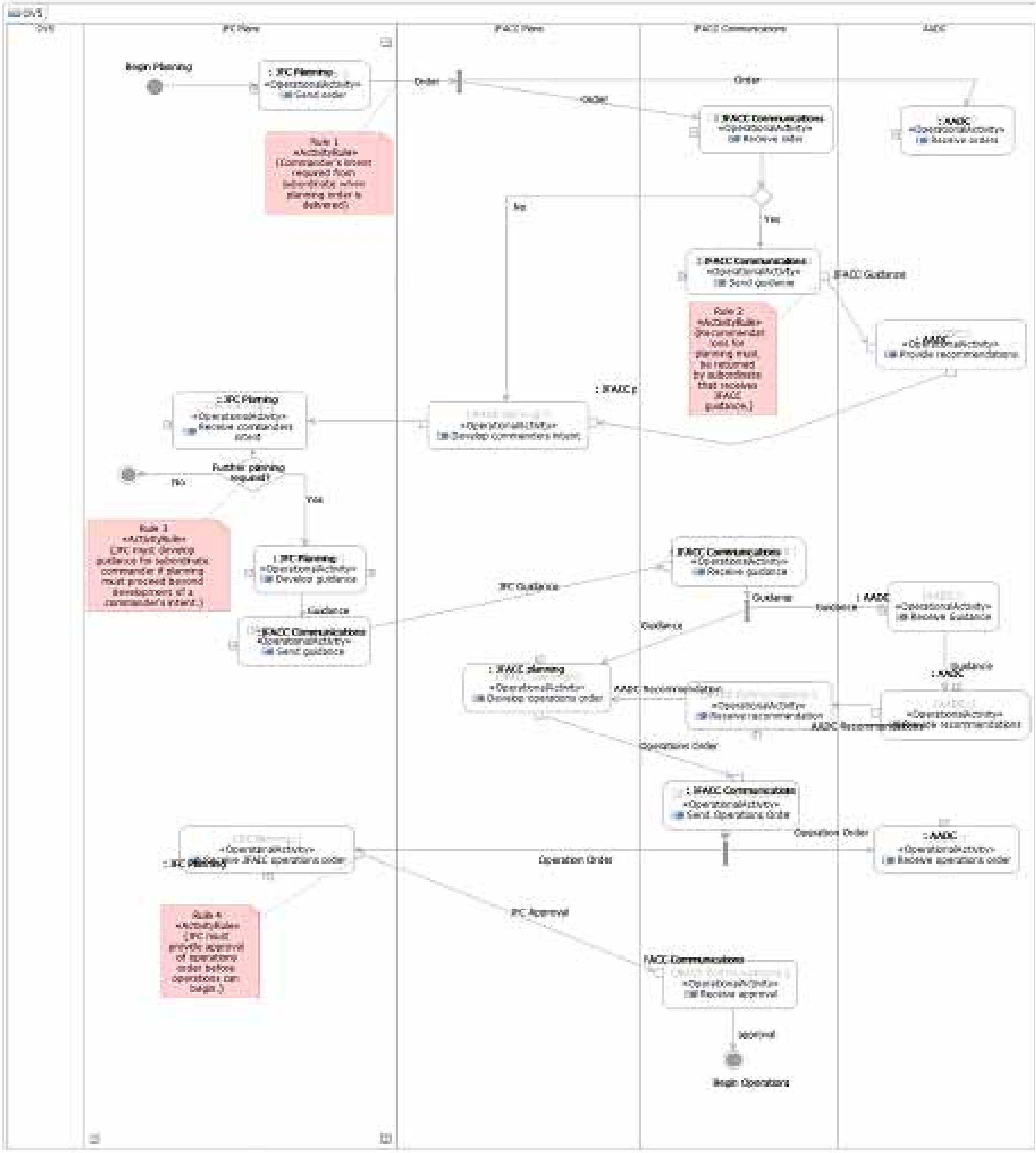


Figure 4 - Operational activities and behavior of the nodes supporting JFACC planning

9. OV-6a: Operational Rules Model

9.1 Introduction

The JFACC planning system provides system support to the joint air operations planning activities required to be accomplished by the JFACC. The operational rules provide the constraints within the architecture for the operational processes and interactions as specified by the joint planning publications.

9.1.1 Purpose

To provide enough information for an example OV-6.

9.1.2 Scope

The scope of these rules are to provide the minimum rule set to explain the UPDM profile through a domain example.

9.1.3 References

The rules were derived without reference.

9.1.4 Overview

9.2 Definitions

All definitions are contained in the AV-2.

9.2.1 <Rule 1>

Joint Doctrine requires the JFACC to develop a Commander's Intent after receipt of the JFC planning order.

9.2.2 <Rule 2>

Subordinate organizations within the AADC COI must develop and provide recommendations for air operations planning to the JFACC after receiving JFACC guidance. This is a doctrine controlled activity.

9.2.3 <Rule 3>

JFC must develop and provide guidance to the JFACC and other subordinate commander once a Commander's intent is produced in order to continue planning. This is a doctrine controlled activity.

9.2.4 <Rule 4>

JFC must provide approval of the JFACC operations order before air operations can begin. This is an organization and doctrine related activity.

10. OV-6b: Operational State Transition Description

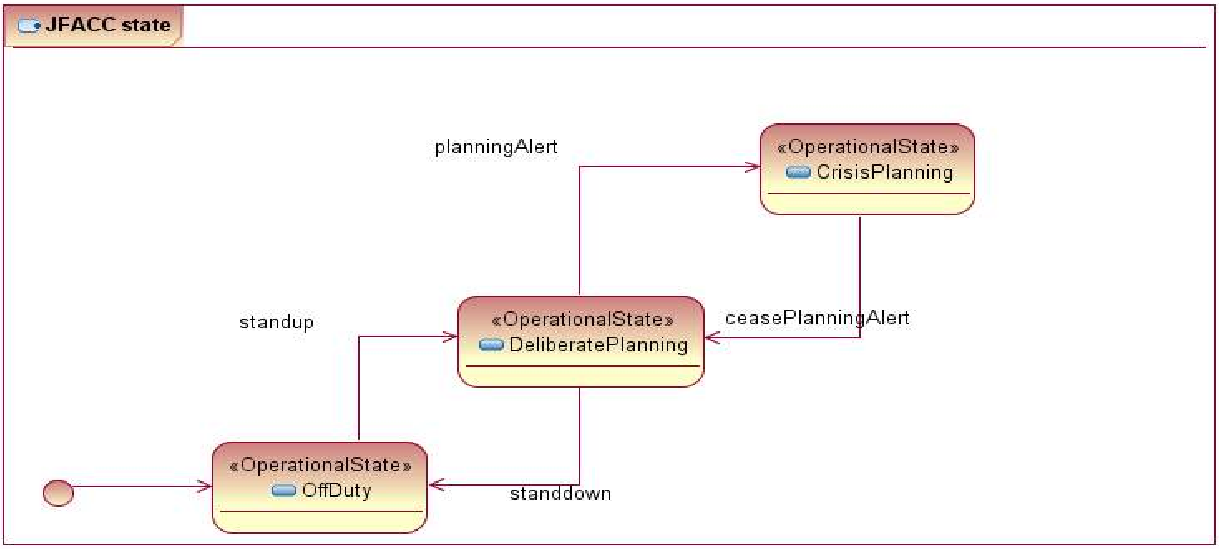


Figure 5 - JFACC operational node states for planning air operations

11. OV-6c: Operational Event-Trace Description

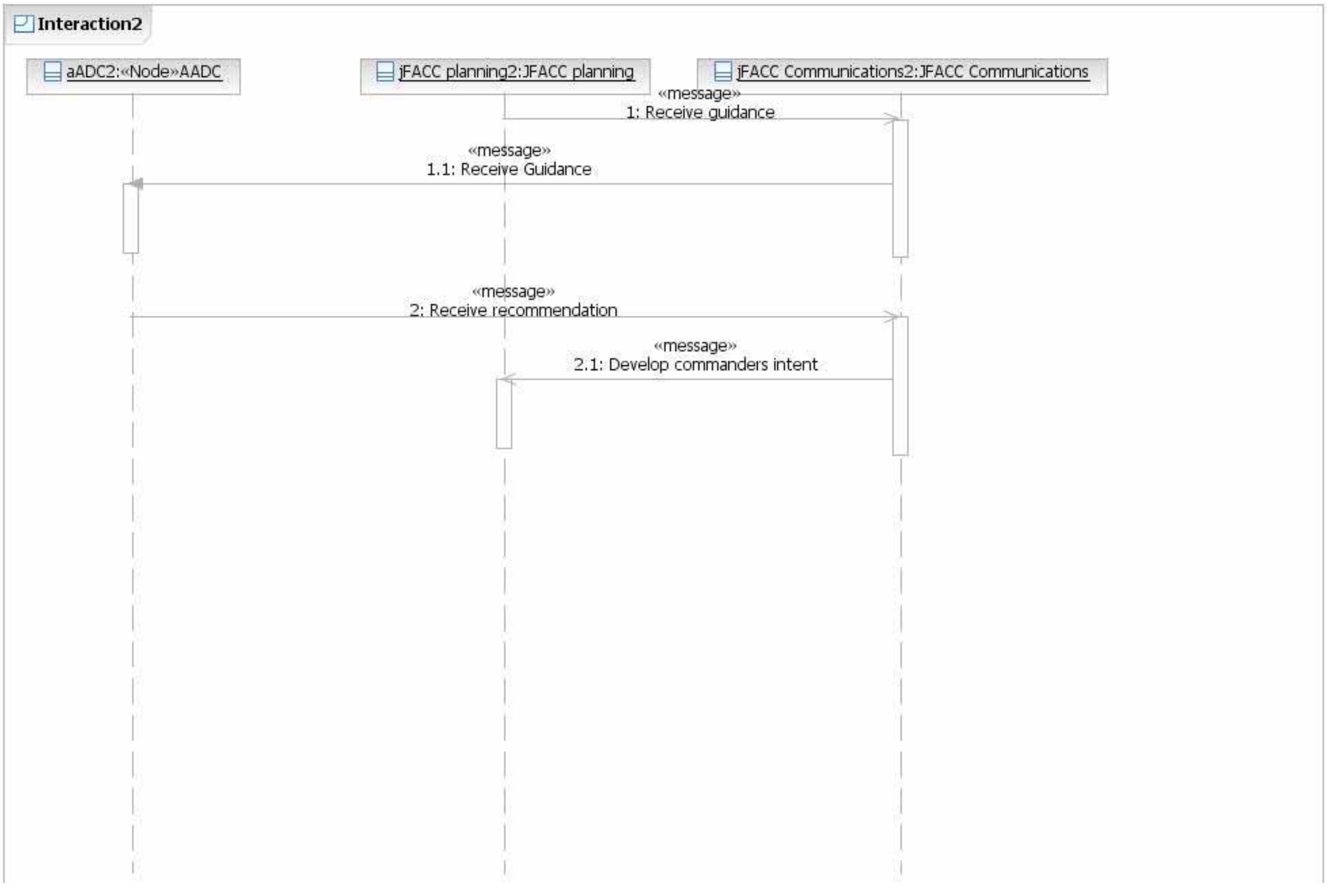

Interaction2
aADC2:«Node»AADC
jFACC planning2:JFACC planning
jFACC Communications2:JFACC Communications
«message»
1: Receive guidance
«message»
1.1: Receive Guidance
«message»
2: Receive recommendation
«message»
2.1: Develop commanders intent

Figure 6 - JFACC and AADC COI interaction to provide recommendations for commander's intent

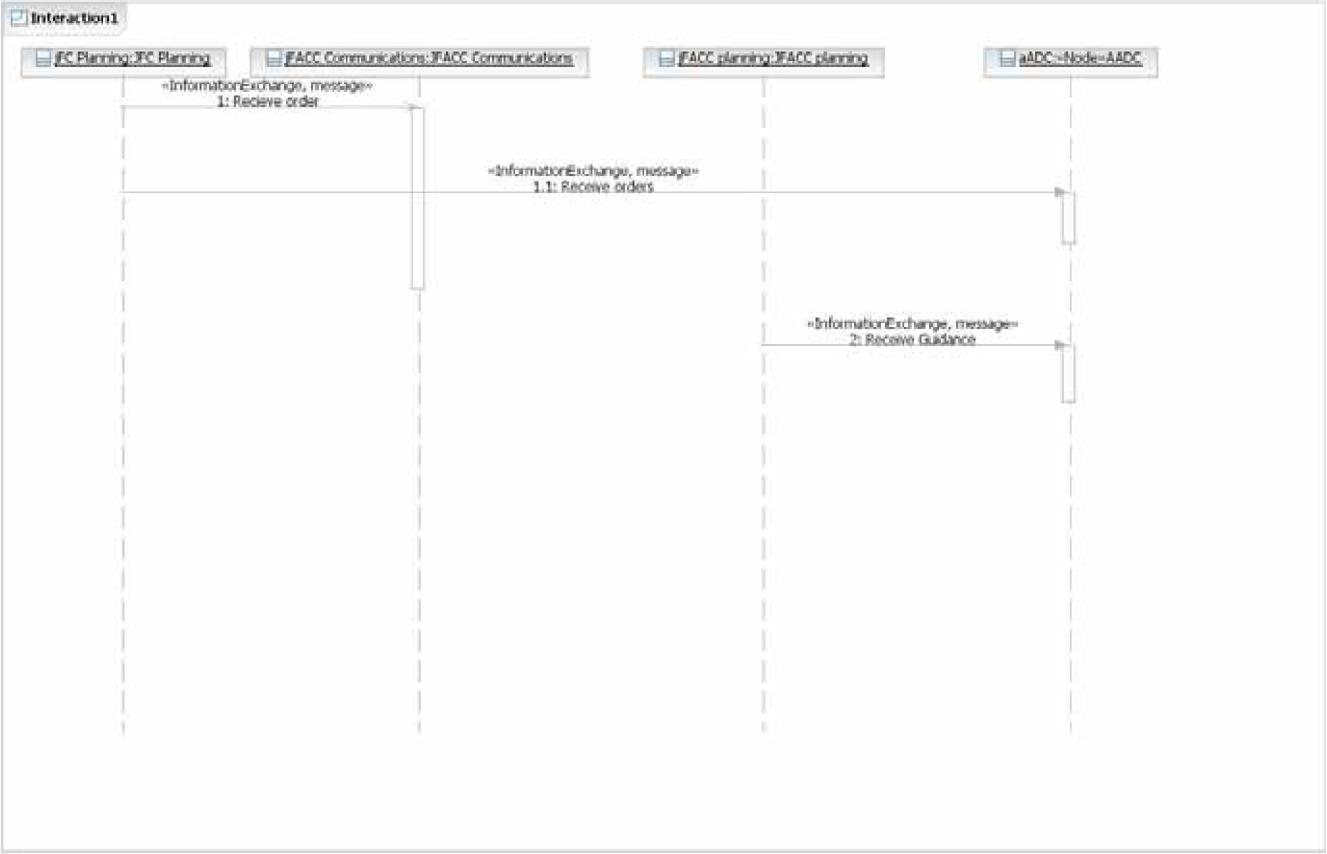

Interaction1
JFC Planning:JFC Planning
JFACC Communications:JFACC Communications
JFACC planning:JFACC planning
aADC:«Node»AADC
«InformationExchange, message»
1: Recieve order
«InformationExchange, message»
1.1: Receive orders
«InformationExchange, message»
2: Receive Guidance

Figure 7 - Interactions between all Nodes to distribute JFC operations order.

12.OV-7: Logical Data Model

Implementation note:

A starting point for auto-generating the OV-7 would be to examine the information exchanges (as represented by stereotyped Messages in the OV-6c), extract the types of the parameters of the operations mapped to the Messages, and populate the OV-7 with these types.

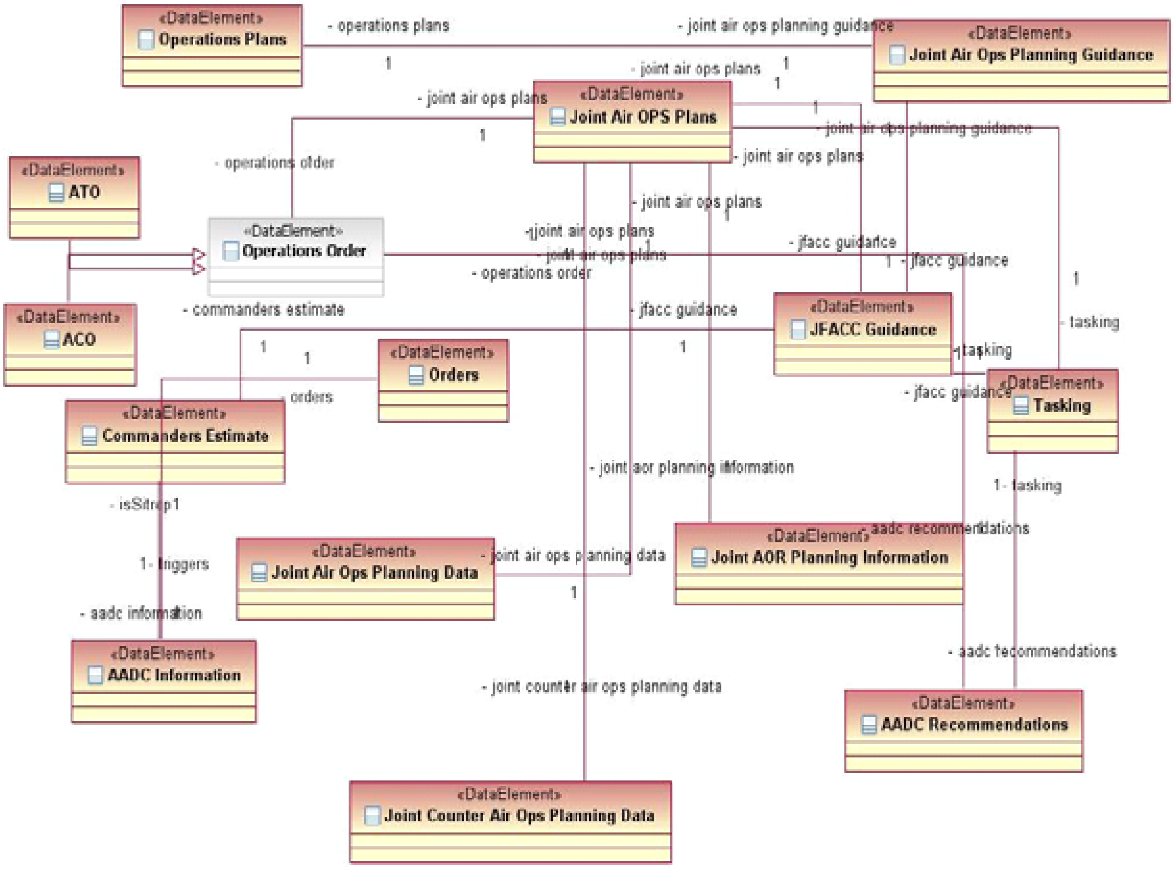

«DataElement» Operations Plans
«DataElement» Joint Air Ops Planning Guidance
«DataElement» Joint Air OPS Plans
«DataElement» ATO
«DataElement» Operations Order
«DataElement» ACO
«DataElement» JFACC Guidance
«DataElement» Orders
«DataElement» Commanders Estimate
«DataElement» Tasking
«DataElement» Joint Air Ops Planning Data
«DataElement» Joint AOR Planning Information
«DataElement» AADC Information
«DataElement» AADC Recommendations
«DataElement» Joint Counter Air Ops Planning Data
- operations plans
- joint air ops planning guidance
- joint air ops plans
- operations order
- commanders estimate
- jfacc guidance
- tasking
- orders
- isSitrep
- triggers
- aadc information
- joint air ops planning data
- joint aor planning information
- aadc recommendations
- joint counter air ops planning data
1

Figure 8 - Logical model of the data exchanged between operational Nodes during planning

Part 3: Systems Views

13. SV-1: Systems Interface Description

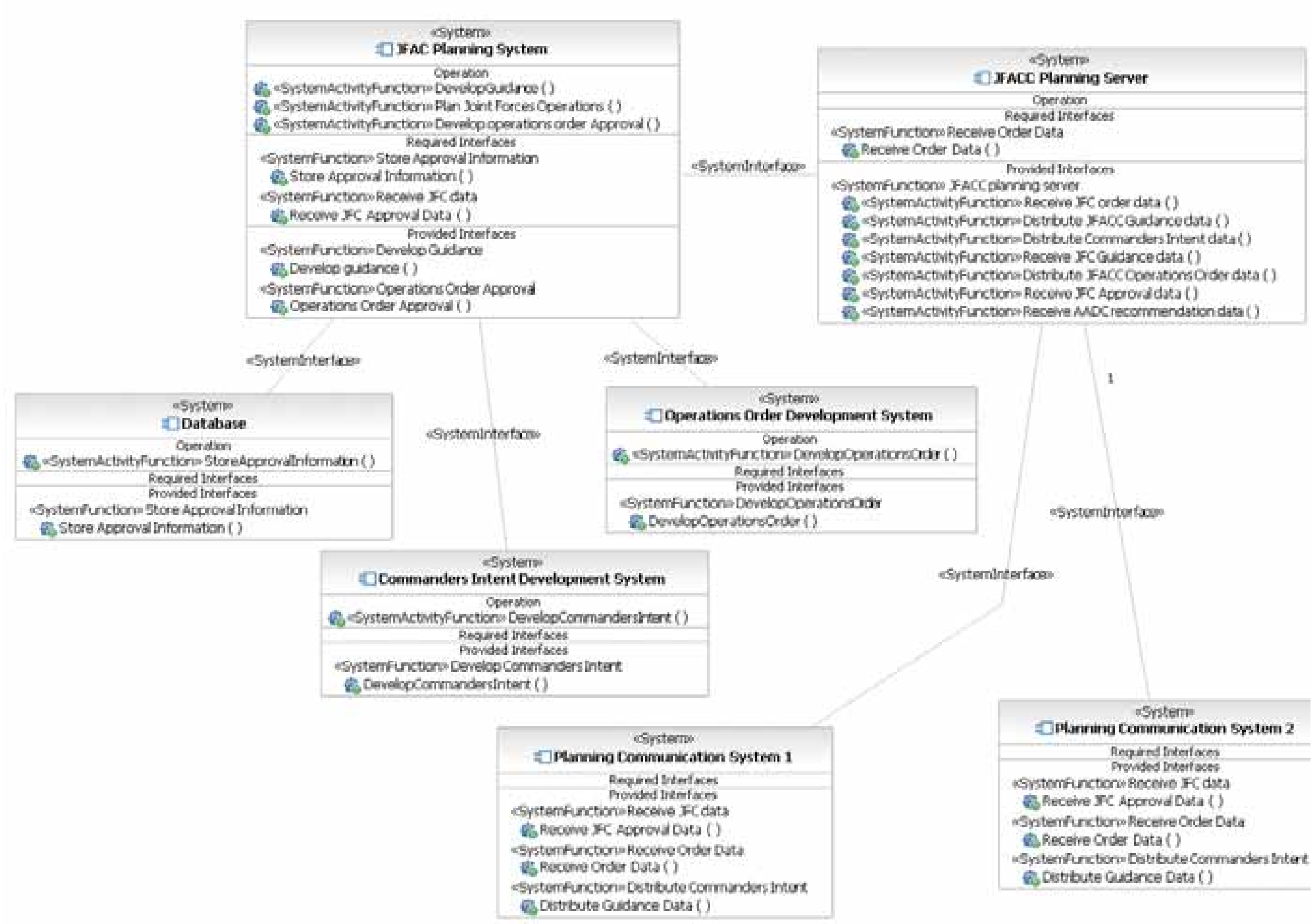

«System»
JFAC Planning System
Operation
«SystemActivityFunction» DevelopGuidance ( )
«SystemActivityFunction» Plan Joint Forces Operations ( )
«SystemActivityFunction» Develop operations order Approval ( )
Required Interfaces
«SystemFunction» Store Approval Information
Store Approval Information ( )
«SystemFunction» Receive JFC data
Receive JFC Approval Data ( )
Provided Interfaces
«SystemFunction» Develop Guidance
Develop guidance ( )
«SystemFunction» Operations Order Approval
Operations Order Approval ( )
«SystemInterface»
«System»
JFACC Planning Server
Operation
Required Interfaces
«SystemFunction» Receive Order Data
Receive Order Data ( )
Provided Interfaces
«SystemFunction» JFACC planning server
«SystemActivityFunction» Receive JFC order data ( )
«SystemActivityFunction» Distribute JFACC Guidance data ( )
«SystemActivityFunction» Distribute Commanders Intent data ( )
«SystemActivityFunction» Receive JFC Guidance data ( )
«SystemActivityFunction» Distribute JFACC Operations Order data ( )
«SystemActivityFunction» Receive JFC Approval data ( )
«SystemActivityFunction» Receive AADC recommendation data ( )
«SystemInterface»
«SystemInterface»
1
«System»
Database
Operation
«SystemActivityFunction» StoreApprovalInformation ( )
Required Interfaces
Provided Interfaces
«SystemFunction» Store Approval Information
Store Approval Information ( )
«SystemInterface»
«System»
Operations Order Development System
Operation
«SystemActivityFunction» DevelopOperationsOrder ( )
Required Interfaces
Provided Interfaces
«SystemFunction» DevelopOperationsOrder
DevelopOperationsOrder ( )
«SystemInterface»
«SystemInterface»
«System»
Commanders Intent Development System
Operation
«SystemActivityFunction» DevelopCommandersIntent ( )
Required Interfaces
Provided Interfaces
«SystemFunction» Develop Commanders Intent
DevelopCommandersIntent ( )
«System»
Planning Communication System 1
Required Interfaces
Provided Interfaces
«SystemFunction» Receive JFC data
Receive JFC Approval Data ( )
«SystemFunction» Receive Order Data
Receive Order Data ( )
«SystemFunction» Distribute Commanders Intent
Distribute Guidance Data ( )
«System»
Planning Communication System 2
Required Interfaces
Provided Interfaces
«SystemFunction» Receive JFC data
Receive JFC Approval Data ( )
«SystemFunction» Receive Order Data
Receive Order Data ( )
«SystemFunction» Distribute Commanders Intent
Distribute Guidance Data ( )

Figure 9 - JFACC Planning Systems interfaces

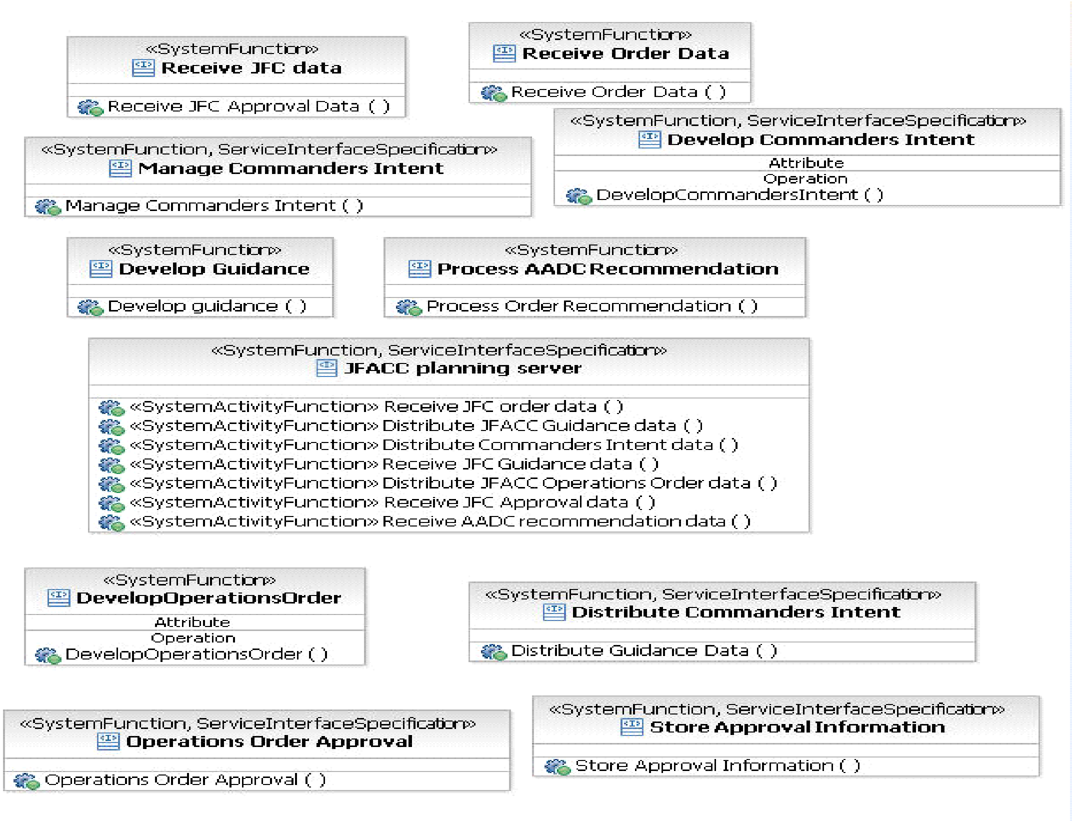

«SystemFunction»
Receive JFC data
Receive JFC Approval Data ( )
«SystemFunction»
Receive Order Data
Receive Order Data ( )
«SystemFunction, ServiceInterfaceSpecification»
Manage Commanders Intent
Manage Commanders Intent ( )
«SystemFunction, ServiceInterfaceSpecification»
Develop Commanders Intent
Attribute
Operation
DevelopCommandersIntent ( )
«SystemFunction»
Develop Guidance
Develop guidance ( )
«SystemFunction»
Process AADC Recommendation
Process Order Recommendation ( )
«SystemFunction, ServiceInterfaceSpecification»
JFACC planning server
«SystemActivityFunction» Receive JFC order data ( )
«SystemActivityFunction» Distribute JFACC Guidance data ( )
«SystemActivityFunction» Distribute Commanders Intent data ( )
«SystemActivityFunction» Receive JFC Guidance data ( )
«SystemActivityFunction» Distribute JFACC Operations Order data ( )
«SystemActivityFunction» Receive JFC Approval data ( )
«SystemActivityFunction» Receive AADC recommendation data ( )
«SystemFunction»
DevelopOperationsOrder
Attribute
Operation
DevelopOperationsOrder ( )
«SystemFunction, ServiceInterfaceSpecification»
Distribute Commanders Intent
Distribute Guidance Data ( )
«SystemFunction, ServiceInterfaceSpecification»
Operations Order Approval
Operations Order Approval ( )
«SystemFunction, ServiceInterfaceSpecification»
Store Approval Information
Store Approval Information ( )

Figure 10 - JFACC planning services

14. SV-2: Systems Communications Description

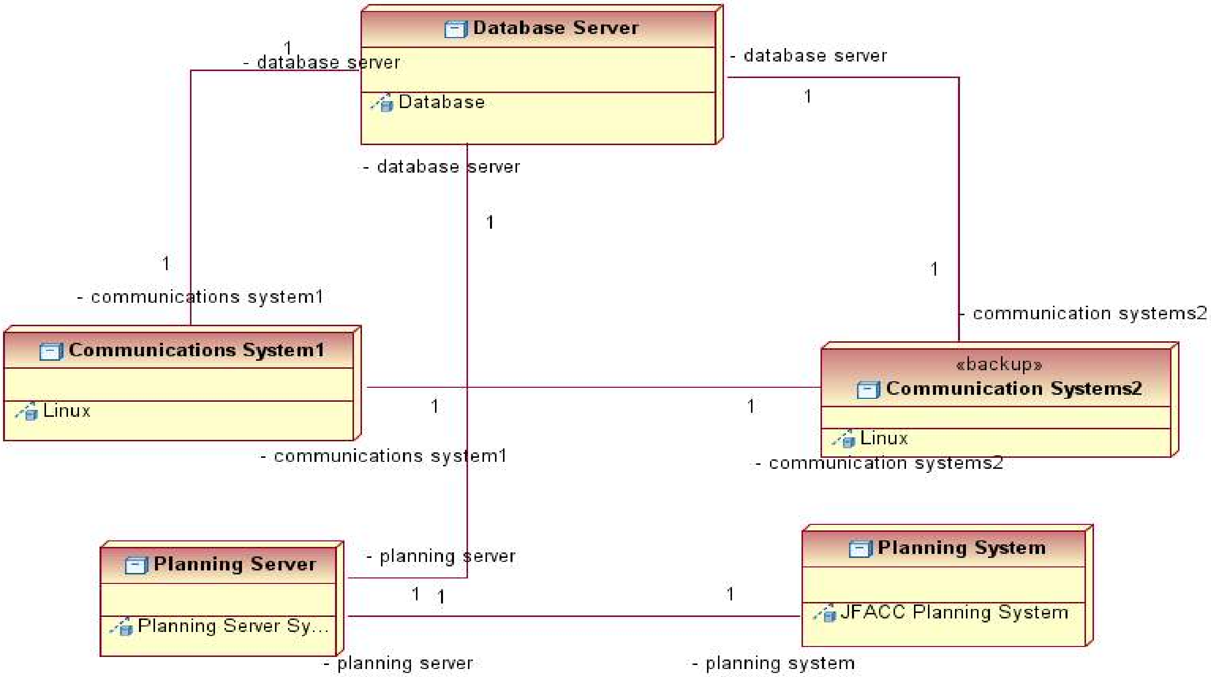


Figure 11 - Deployment diagram of the JFACC planning system

15. SV-3: Systems-Systems Matrix

This view was generated with the DoDAF plugin. This will be updated as the UPDM implementation is completed.

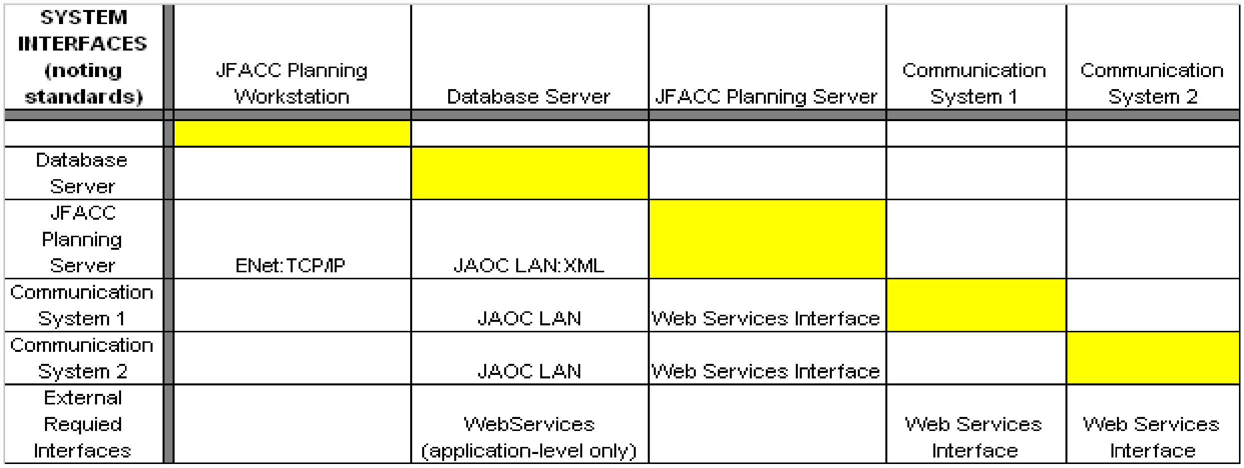
Table 4 - Systems to systems association matrix for the JFACC planning system

	AADC planning network	Commanders Intent Development System	Develop Guidance	Distribute Commanders Intent	JFACC planning server	Manage Commander's Intent	Operations Order Approval	Process AADC Recommendation	Receive JFC data	Receive Order Data	Store Approval Information
AADC planning network					X						
Commanders Intent Development System					X						
Develop Guidance							X	X			
Distribute Commanders Intent						X					

Table 4 - Systems to systems association matrix for the JFACC planning system

JFACC planning server	X	X									
JFC planning network	X				X						
Manage Commander’s Intent				X					X		
Operations Order Approval			X							X	
Process AADC Recommendation			X								
Receive JFC data						X					
Receive Order Data							X				X
Store Approval Information										X	

Table 5 - System interface matrix that also indicates service interface specifications



SYSTEM INTERFACES (noting standards)	JFACC Planning Workstation	Database Server	JFACC Planning Server	Communication System 1	Communication System 2
	🟨				
Database Server		🟨			
JFACC Planning Server	ENet:TCP/IP	JAOC LAN:XML	🟨		
Communication System 1		JAOC LAN	Web Services Interface	🟨	
Communication System 2		JAOC LAN	Web Services Interface		🟨
External Requied Interfaces		WebServices (application-level only)		Web Services Interface	Web Services Interface

16.SV-4: Systems Functionality Description

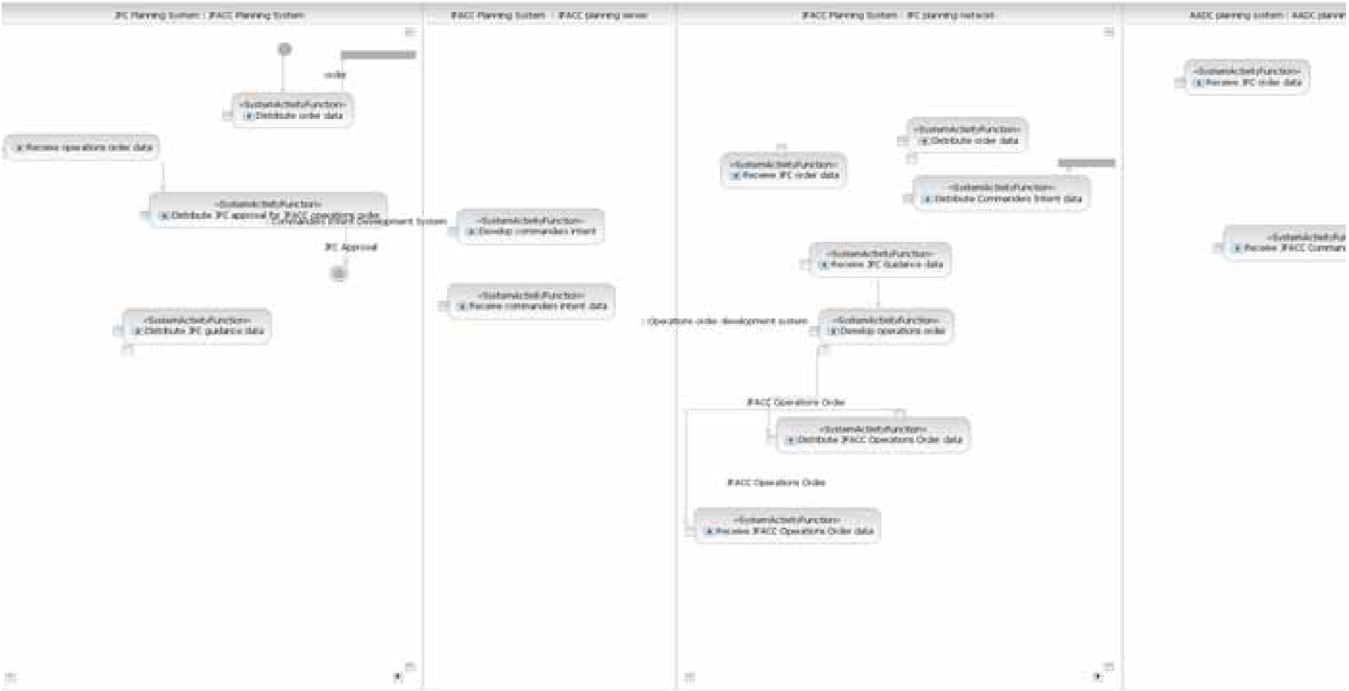

JFC Planning System : JFACC Planning System
JFACC Planning System : JFACC planning server
JFACC Planning System : JFC planning network
AADC planning system : AADC plann
order
«SystemActivityFunction»
Distribute order data
Receive operations order data
«SystemActivityFunction»
Distribute JFC approval for JFACC operations order
Commanders intent Development System
JFC Approval
«SystemActivityFunction»
Distribute JFC guidance data
«SystemActivityFunction»
Develop commanders intent
«SystemActivityFunction»
Receive commanders intent data
Operations order development system
«SystemActivityFunction»
Receive JFC order data
«SystemActivityFunction»
Distribute order data
«SystemActivityFunction»
Distribute Commanders Intent data
«SystemActivityFunction»
Receive JFC Guidance data
«SystemActivityFunction»
Develop operations order
JFACC Operations Order
«SystemActivityFunction»
Distribute JFACC Operations Order data
JFACC Operations Order
«SystemActivityFunction»
Receive JFACC Operations Order data
«SystemActivityFunction»
Receive JFC order data
«SystemActivityFu
Receive JFACC Comman

Figure 12 - JFACC planning system behavior and interaction with external systems

17.SV-5: Operational Activity to Systems Function Traceability Matrix

Table 6 - JFACC operations and function mapping

	Receive JFC Data	Manage Commanders Intent	Distribute Commanders Intent	Process AADC Recommendation	Develop Guidance	Operations Order Approval	Receive Order Data	Store Approval Information
Develop Commanders Intent	X	X	X					
Develop Operations Order				X	X	X	X	X

18.SV-6: Systems Data Exchange Matrix

This view was generated with the DoDAF plugin. This will be updated as the UPDM implementation is completed.

Table 7 - Critical data exchanges characteristics

Data Exchange Identifier	Producer	Consumer	Content	Timeliness	Throughput	Security
AADC planning network:: JFACC planning server	AADC planning network	JFACC planning server	Force Location information	1 minute	NA	U
AADC planning network:: JFC planning network	AADC planning network	JFC planning network	System status	30 seconds	NA	U

Table 7 - Critical data exchanges characteristics

Commanders Intent Development System:: JFACC planning server	Commanders Intent Development System	JFACC planning server	Planning data	1 minute	NA	U
Develop Guidance:: Operations Order Approval	Develop Guidance	Operations Order Approval	Text document	30 seconds	NA	U
JFACC planning server:: AADC planning network	JFACC planning server	AADC planning network	Text document	30 seconds	NA	U
JFACC planning server:: Commanders Intent Development System	JFACC planning server	Commanders Intent Development System	Text document	30 seconds	NA	U
JFACC planning server:: JFC planning network	JFACC planning server	JFC planning network	Text document	30 seconds	NA	U
JFACC planning server:: JFC planning network	JFACC planning server	JFC planning network	Text document	30 seconds	NA	U
JFACC planning server:: JFC planning network	JFACC planning server	JFC planning network	Text document	30 seconds	NA	U
JFC planning network:: AADC planning network	JFC planning network	AADC planning network	Force data	30 seconds	NA	U
JFC planning network:: JFACC planning server	JFC planning network	JFACC planning server	Force data	30 seconds	NA	U
Manage Commander’s Intent:: Distribute Commander’s Intent	Manage Commanders Intent	Distribute Commanders Intent	Force data	30 seconds	NA	U
Operations Order Approval:: Receive Order Data	Operations Order Approval	Receive Order Data	Force data	30 seconds	NA	U
Process AADC Recommendation::Develop Guidance	Process AADC Recommendation	Develop Guidance	Force data	30 seconds	NA	U
Receive JFC data::Manage Commanders Intent	Receive JFC data	Manage Commanders Intent	Force order	30 seconds	NA	U
Receive Order Data:: Store Approval Information	Receive Order Data	Store Approval Information	Force order	30 seconds	NA	U

19. SV-7: Systems Performance Parameters Matrix

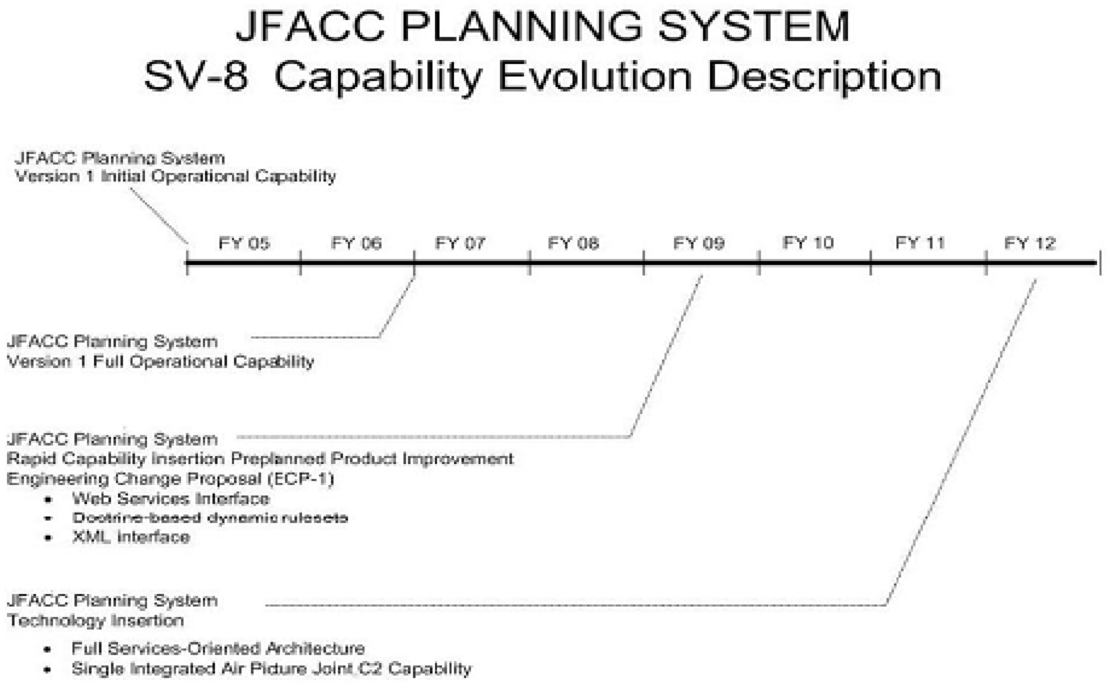
Table 8 - Performance metrics for JFACC planning system

System	Subsystem	Performance Metric	Performance Threshold / Objective			
JFACC Planning System	Planning Component Platform (processor/OS/ middleware)					
		Reliability (MTBF, hours)	1000	2000	2000/5000	5000/104
		Availability (Ao)	0.95	0.99	0.99/0.995	0.999/0.9999
		Initialization delay (sec)	5.0	2.5	2.5/1.0	1.0/0.05
		Data transfer rate (MBps)	102	103	103/104	104/105

Table 8 - Performance metrics for JFACC planning system

		Restart time (sec)	30	30	30/10	2
	Planning Application					
		Reliabilty (MTB software failure, hours)	500	2000	5000/103	104/105
		Availability (Ao)	0.95	0.99	0.995/0.999	0.9995/0.999999
		Flight plan capacity per 24 hours	2000	5000	5000/104	5000/104
	Alert Manager Application					
		Operator interaction response time (sec)	5	2.5	2.5/1.0	0.25/0.10
		Reliability (MTB software failure, hours)	500	2000	5000	104/105
		Availability (Ao)	0.95	0.99	0.995/0.999	0.9995/0.999999
	Database/Server					
		Query response time (sec)	0.5	0.2	0.2/0.1	0.05/0.01
		Data capacity (GB)	800	2000	104/105	105/106

20. SV-8: Systems Evolution Description


JFACC PLANNING SYSTEM
SV-8 Capability Evolution Description
JFACC Planning System
Version 1 Initial Operational Capability
FY 05
FY 06
FY 07
FY 08
FY 09
FY 10
FY 11
FY 12
JFACC Planning System
Version 1 Full Operational Capability
JFACC Planning System
Rapid Capability Insertion Preplanned Product Improvement
Engineering Change Proposal (ECP-1)
• Web Services Interface
• Doctrine-based dynamic rulesets
• XML interface
JFACC Planning System
Technology Insertion
• Full Services-Oriented Architecture
• Single Integrated Air Picture Joint C2 Capability

Evolution of the JFACC planning system through the next few years.

21. SV-9: Systems Technology Forecast

Table 9 - Forecast of technology that will be available for the JFACC planning system

DISR Service	TECHNOLOGY FORECASTS		
	NEAR TERM (1-2 Years)	MID TERM (2-4 Years)	LONG TERM (>4 Years)
Application Platform			
Geospatial	Guidelines 3.123		
Data Services	Web Content Accessibility Guideline 17.0		
Communications			
Data Exchange		WIDERESR	X-2010
Integration (System of Systems)			
Architecture	UML Profile for DoDAF/MODAF		DoD SOA
Model-Based System Interoperability			Integrated Object Model 2.0

22. SV-10a: Systems Rules Model

22.1 Introduction

The JFACC planning system provides system support to the joint air operations planning activities required to be accomplished by the JFACC. The systems rules provide the constraints within the architecture for the systems processes and interactions as specified by the joint planning publications.

22.1.1 Purpose

To provide an example SV-10C.

22.1.2 Scope

The scope of these rules are to provide the minimum rule set to explain the UPDM team ONE profile through a domain example.

22.1.3 References

The rules were derived without reference and are basic system interactions.

22.1.4 Overview

These rules are provided as a context for the UPDM Team ONE submission example.

22.2 Definitions

All definitions are contained in the AV-2.

22.2.1 <Rule 1>

Each node receiving the JFC order data must provide a confirmation of receipt to the JFC planning system network interface via the same network the data was received on.

22.2.2 <Rule 2>

Each node receiving the AADC recommendation data must provide a confirmation of receipt to the JFACC planning system via the same network the data was received on.

22.2.3 <Rule 3>

Each node receiving the JFACC Commanders intent data must provide a confirmation of receipt to the JFACC planning system via the same network the data was received on.

23. SV-10b: Systems State Transition Description

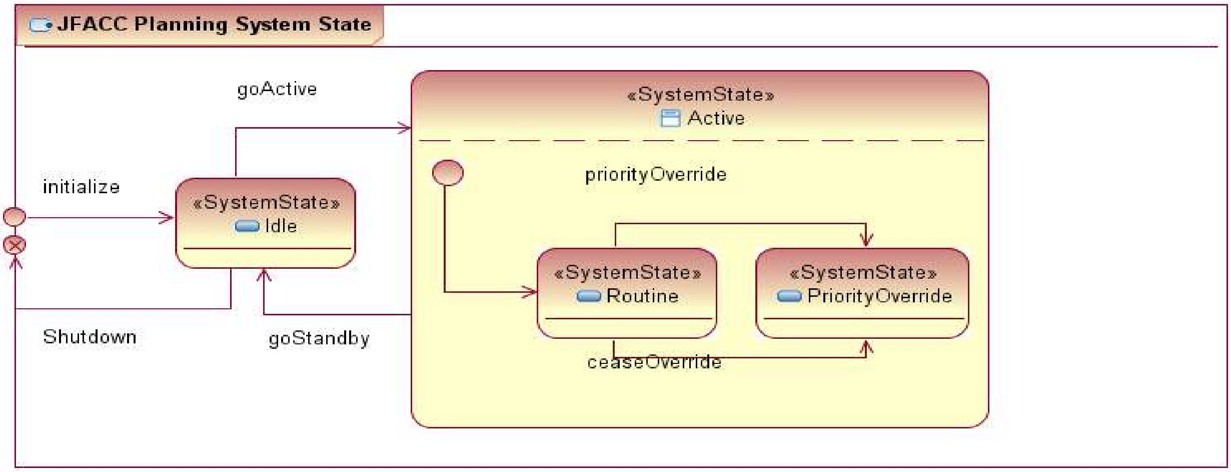


Figure 13 - System states of the JFACC planning system

24. SV-10c: Systems Event-Trace Description

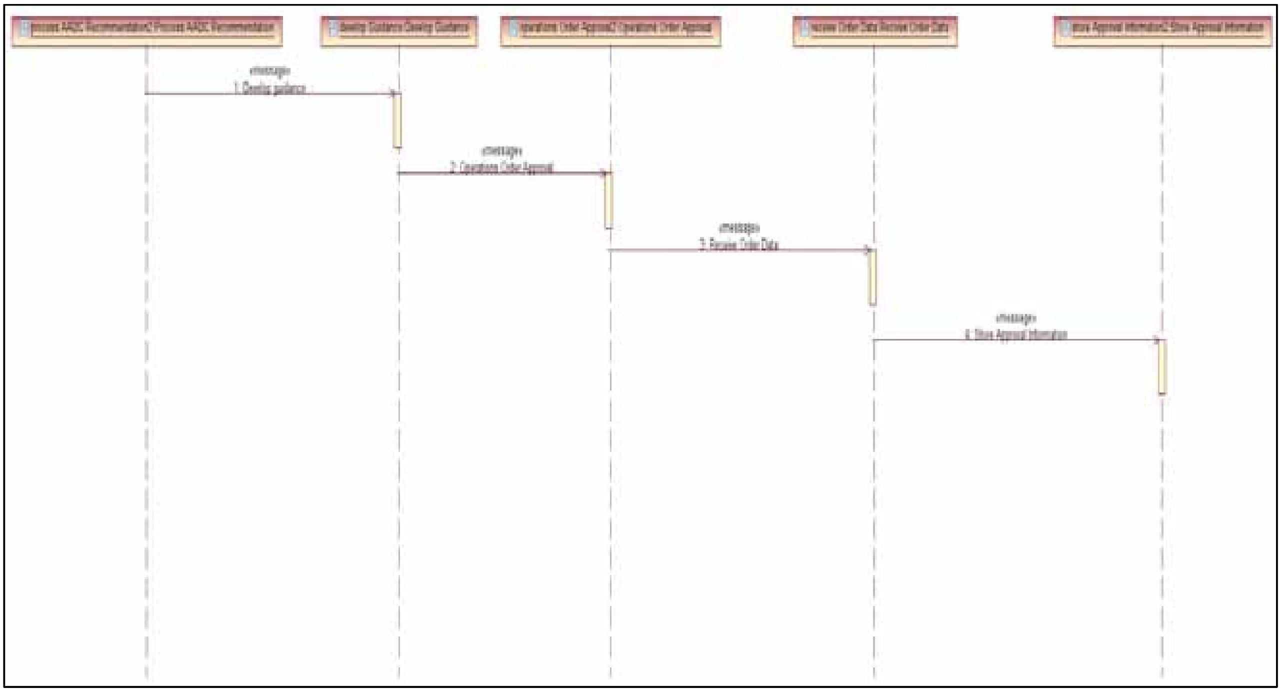

rocess AADC Recommendation2:Process AADC Recommendation
evelop Guidance:Develop Guidance
perations Order Approval2:Operations Order Approval
eview Order Data:Review Order Data
tore Approval Information2:Store Approval Information
«message»
1. Develop guidance
«message»
2. Operations Order Approval
«message»
3. Review Order Data
«message»
4. Store Approval Information

Figure 14 - Conduct counter air operations

Figure 13 - System interfactions to develop the commander's intent

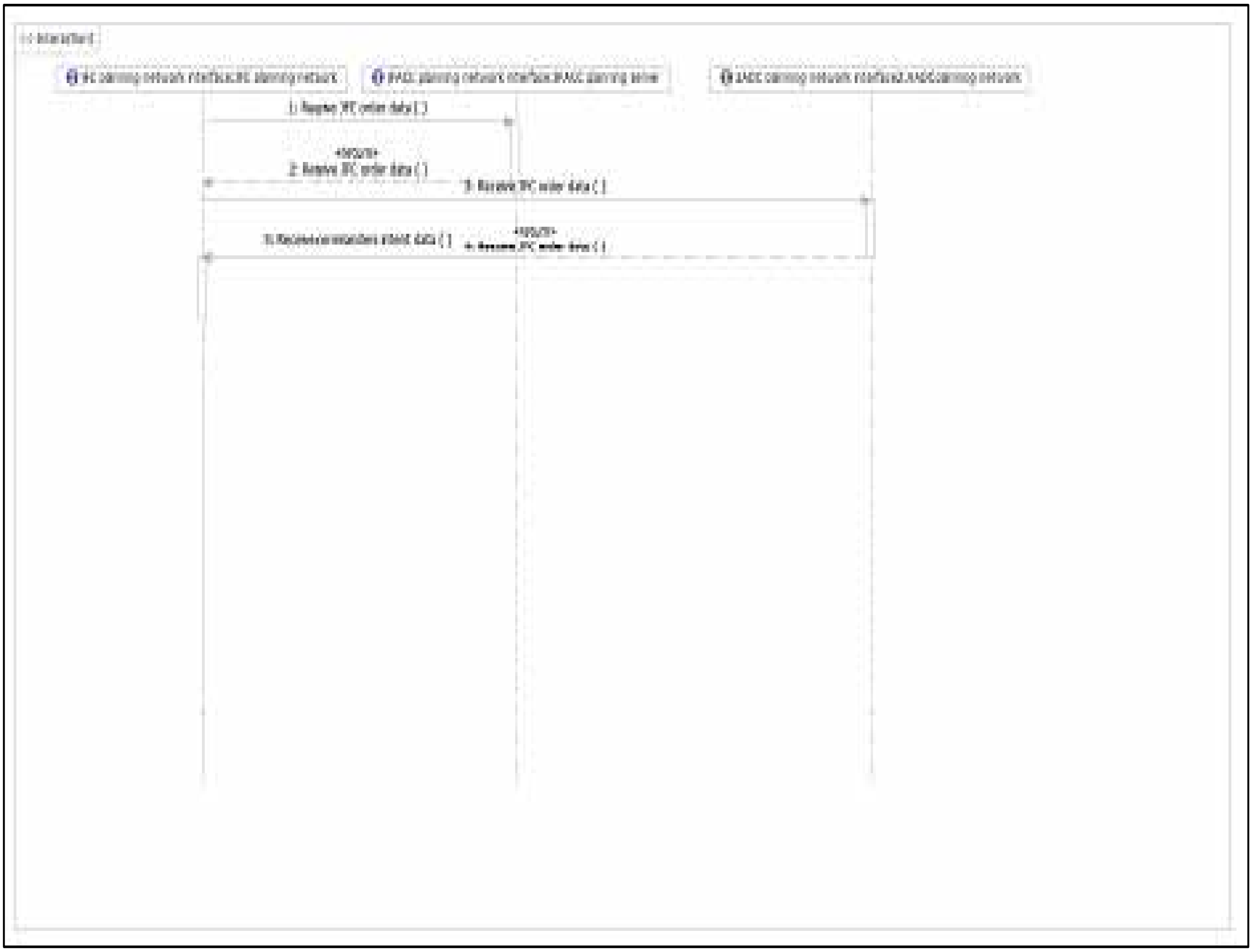

JFC planning network interface:JFC planning network
JFACC planning network interface:JFACC planning server
JAOC planning network interface2:JAOC planning network
1: Request JFC order data ( )
«return»
2: Request JFC order data ( )
3: Receive JFC order data ( )
5: Receive commanders intent data ( )
«return»
4: Receive JFC order data ( )

Figure 15 - Develop commander's intent

25. SV-11: Physical Schema

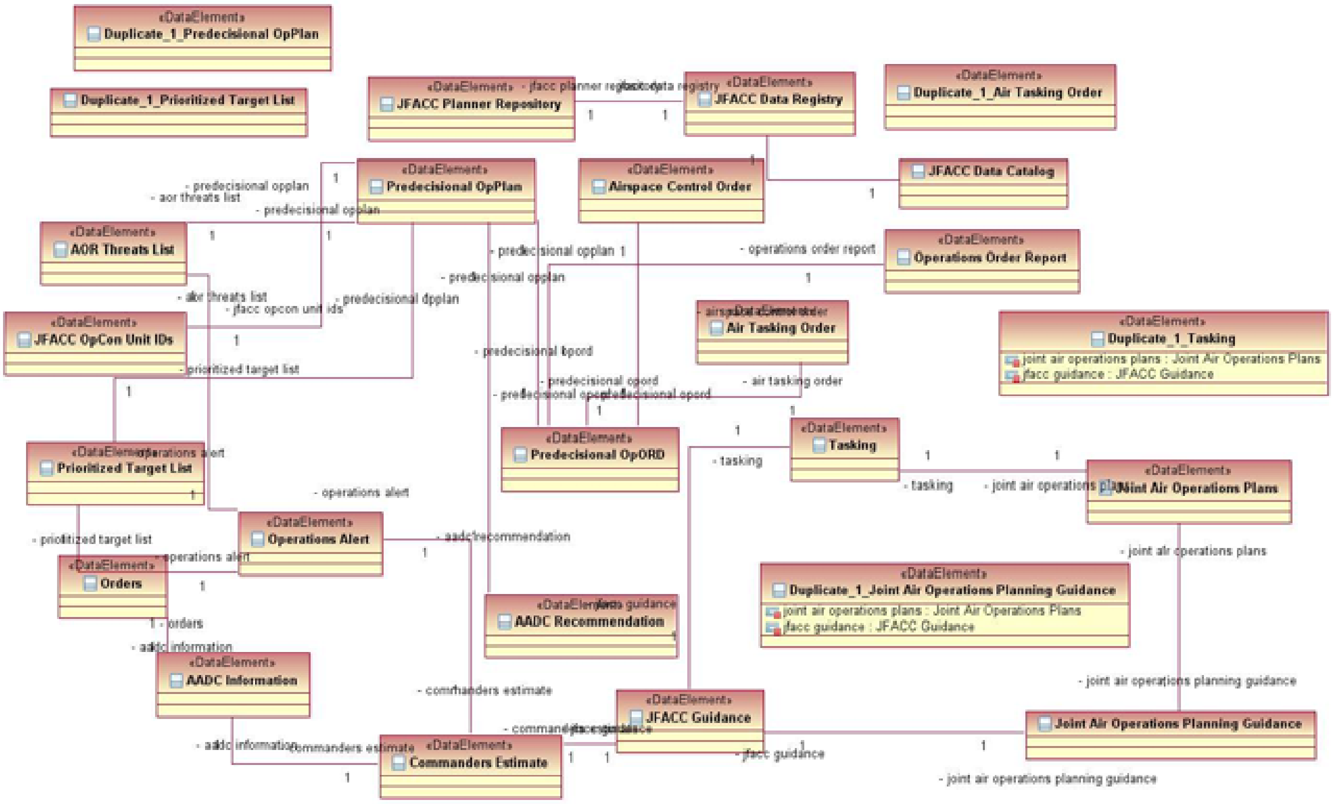

«DataElement» Duplicate_1_Predecisional OpPlan
Duplicate_1_Prioritized Target List
«DataElement» JFACC Planner Repository
- jfacc planner repository
- jfacc data registry
«DataElement» JFACC Data Registry
«DataElement» Duplicate_1_Air Tasking Order
JFACC Data Catalog
«DataElement» Predecisional OpPlan
«DataElement» Airspace Control Order
- predecisional opplan
- aor threats list
- predecisional opplan
«DataElement» AOR Threats List
«DataElement» Operations Order Report
- operations order report
- predecisional opplan
- predecisional opplan
- aor threats list
- jfacc opcon unit ids
- predecisional opplan
«DataElement» JFACC OpCon Unit IDs
- airspace control order
«DataElement» Air Tasking Order
«DataElement» Duplicate_1_Tasking
joint air operations plans : Joint Air Operations Plans
jfacc guidance : JFACC Guidance
- prioritized target list
- predecisional opord
- predecisional opord
- predecisional opord
- predecisional opord
- air tasking order
«DataElement» Prioritized Target List
- operations alert
«DataElement» Predecisional OpORD
«DataElement» Tasking
- tasking
- tasking
- joint air operations plans
«DataElement» Joint Air Operations Plans
- operations alert
- prioritized target list
«DataElement» Operations Alert
- aadc recommendation
- operations alert
«DataElement» Orders
- joint air operations plans
«DataElement» Duplicate_1_Joint Air Operations Planning Guidance
joint air operations plans : Joint Air Operations Plans
jfacc guidance : JFACC Guidance
«DataElement» AADC Recommendation
- jfacc guidance
- orders
- aadc information
«DataElement» AADC Information
- commanders estimate
- joint air operations planning guidance
«DataElement» JFACC Guidance
- commanders estimate
- jfacc guidance
Joint Air Operations Planning Guidance
- aadc information
- commanders estimate
«DataElement» Commanders Estimate
- jfacc guidance
- joint air operations planning guidance
1

Figure 16 - The logical data model for the JFACC planning system.

Part 4: Technical Standards Views

26. TV-1: Technical Standards Profile

Table 10 - Technical standards applicable to the development of JFACC planning system

Service Area	Standard Identifier	Standard Reference
Architectures	OMG UML 2.0	Unified Modeling Language: Superstructure, version 2.0, OMG Final Adopted Specification, August 2003
Architectures	Core Architecture Data Model (CADM)	Core Architecture Data Model (CADM), Baseline Release 1.02, January 27 2005 (with approved implementation date of July 01 2005).
Architectures	Model Driven Architecture	Model Driven Architecture Guide ver. 1.0.1
Data Exchange	OMG ptc/03-07-07	Data Distribution Service for Real-Time Systems Specification, Version 1.0, July 2003.
Data Exchange	RDF Vocabulary Description Language 1.0: RDF Schema	Resource Description Framework (RDF) Vocabulary Description Language 1.0: RDF Schema W3C Recommendation 10 February 2004
Data Exchange	XML 1.1:2004	Extensible Markup Language (XML) 1.1, W3C Recommendation 04 February 2004, edited in place 15 April 2004
Geospatial	WMS 1.3	OpenGIS® Web Map Service (WMS) Implementation Specification

Table 10 - Technical standards applicable to the development of JFACC planning system

Graphics Exchange	ISO/IEC 15948	Portable Network Graphics (PNG): Functional Specification Final Committee Draft (FCD), 2000.
Graphics Services	OpenGL Graphics System:2001	OpenGL Graphics System: A Specification (Version 1.3), 14 August 2001.
Imagery Exchange	MIL-STD-2500B(2)	National Imagery Transmission Format (Version 2.1) for the National Imagery Transmission Format Standard, 22 August 1997 with Notice 1, 2 October 1998, and Notice 2, 1 March 2001.
Information Assurance	CMS/XML Digital Signature Profiles v1.1	DoD Digital Signature Implementation Profiles
Information Assurance	IETF RFC 3275	Extensible Markup Language (XML) Signature Syntax and Processing
Metadata Exchange	XMI	XML Metadata Interchange, Version 1.1, ad/99-10-22, 25 October 1999.
Modeling and Simulation	IEEE 1516.1	IEEE 1516.1-2000 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification
Operating System Services	ISO/IEC 9945-2:2003	Portable Operating System Interface(POSIX) - Part 2: System Interfaces, 2003.
Policy-based C4I	IETF RFC 3377	Lightweight Directory Access Protocol (v3): Technical Specification; September 2002
Web Services	SOAP 1.2	Simple Object Access Protocol (SOAP) 1.2, W3C Recommendation 24 June 2003
Web Services	WSDL 1.1	Web Services Description Language (WSDL) 1.1, W3C Note, 15 March 2001.
Web Services	OWL	Web Ontology Language
Web Services	UDDI 3.0.2	OASIS Universal Description, Discovery, and Integration Version 3.0.2 UDDI Spec, Dated 2004-Oct-19

27. TV-2: Technical Standards Forecast

Table 11 - Forecasted standards for the JVFACC planning system

Service Area	Standard Identifier	Standard Reference	Expected Effectivity
Architecture	UML for DoDAF/MODAF	Request For Proposal OMG Document: c4i/05-09-12	June 2007
Architecture	DoD SOA 2.0	DoD Services-Oriented Architecture Enterprise Reference Model, Ver. 2.0; ASD (NII)	2010
Data Exchange	WiDEX	IETF Widget Description Exchange Service	2008
Data Services	Web Content Accessibility Guideline 2.0	W3C Call for Review: Web Content Accessibility Guideline 2.0, Apr. 2006	2007
Geospatial	GML 3.1.1	OpenGIS Geography Markup Language Encoding Specification	mid 2007

This document was generated from the model and auxiliary documents.

Annex D
Bibliography

References

An IBM Rational Approach to the Department of Defense Architecture Framework (DoDAF), January 2006, IBM Rational, ftp://ftp.software.ibm.com/software/rational/web/whitepapers/G507-1903-00_v5_LoRes.pdf

MOD Architectural Framework, Viewpoint Overview, Version 1.0, 31 August 2005, MODAF Partners, Inc., http://www.modaf.com/Documents

DoD Architecture Framework Version 1.0, Volume I: Definitions and Guidelines, 9 February 2004, DoD Architecture Framework Working Group

DoD Architecture Framework Version 1.0, Volume II: Product Descriptions 9 February 2004, DoD Architecture Framework Working Group

DoD Architecture Framework Version 1.5, Volume II: Product Descriptions, 13 September 2006. Final Draft. DoD Architecture Framework Working Group

DoD Dictionary of Military Terms, http://www.dtic.mil/doctrine/jel/doddict/index.html

Joint Doctrine Joint Force Employment Briefing, http://www.dtic.mil/doctrine/jrm/plans.pdf

Joint Forces Staff College, Joint Planning and Operations Course, http://www.jfsc.ndu.edu/schools_programs/jpoc/course_materials/default.asp

Doctrine for Joint Operations, http://www.dtic.mil/doctrine/jel/new_pubs/jp3_0.pdf

Joint Doctrine for countering air and missile threats, http://www.dtic.mil/doctrine/jel/new_pubs/jp3_01.pdf

Joint Communications System, http://www.dtic.mil/doctrine/jel/new_pubs/jp6_0.pdf

Federal Information Processing Standards Publication (FIPS PUB) 183 -- INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0), issued by the National Institute of Standards and Technology after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.