

Recommendation for Space Data System Practices

REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

RECOMMENDED PRACTICE

CCSDS 311.0-M-2

Magenta Book
December 2024

Recommendation for Space Data System Practices

REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

RECOMMENDED PRACTICE

CCSDS 311.0-M-2

Magenta Book
December 2024

AUTHORITY

Issue:	Recommended Practice, Issue 2
Date:	December 2024
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the email address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
Email: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommendations** and are not in themselves considered binding on any Agency.

CCSDS Recommendations take two forms: **Recommended Standards** that are prescriptive and are the formal vehicles by which CCSDS Agencies create the standards that specify how elements of their space mission support infrastructure shall operate and interoperate with others; and **Recommended Practices** that are more descriptive in nature and are intended to provide general guidance about how to approach a particular problem associated with space mission support. This **Recommended Practice** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommended Practice** is entirely voluntary and does not imply a commitment by any Agency or organization to implement its recommendations in a prescriptive sense.

No later than five years from its date of issuance, this **Recommended Practice** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Practice** is issued, existing CCSDS-related member Practices and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such Practices or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new Practices and implementations towards the later version of the Recommended Practice.

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Practice is therefore subject to CCSDS document management and change control procedures, which are defined in the *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

<http://www.ccsds.org/>

Questions relating to the contents or status of this document should be sent to the CCSDS Secretariat at the email address indicated on page i.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (BELSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- China Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- Egyptian Space Agency (EgSA)/Egypt.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Hellenic Space Agency (HSA)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Netherlands Space Office (NSO)/The Netherlands.
- Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 311.0-M-1	Reference Architecture for Space Data Systems, Recommended Practice, Issue 1	September 2008	Original issue, superseded
CCSDS 311.0-M-2	Reference Architecture for Space Data Systems, Recommended Practice, Issue 2	December 2024	Current issue: defines extended approach specifically adapted for the space domain and aligned with best current practices in the fields of system and software architecture and modeling.

NOTE – Changes from the original issue are too numerous to permit meaningful use of change bars.

CONTENTS

<u>Section</u>	<u>Page</u>
1 INTRODUCTION	1-1
1.1 SCOPE	1-1
1.2 PURPOSE	1-1
1.3 APPLICABILITY	1-2
1.4 RATIONALE	1-2
1.5 DOCUMENT STRUCTURE	1-4
1.6 REFERENCES	1-4
2 OVERVIEW	2-1
2.1 GENERAL	2-1
2.2 RELATING ARCHITECTURE AND SYSTEMS ENGINEERING	2-1
2.3 OVERVIEW OF RASDSV2 VIEWPOINTS	2-2
2.4 SELECTING RASDSV2 VIEWPOINTS FOR A GIVEN USE	2-7
2.5 SECURITY VIEWS IN RASDSV2	2-8
3 RASDSV2 MODELLING CONCEPTS	3-1
3.1 GENERAL	3-1
3.2 DEFINITIONS	3-1
3.3 GRAPHICAL REPRESENTATIONS	3-7
3.4 CHARACTERISTICS OF OBJECTS	3-7
4 ENTERPRISE VIEWPOINT	4-1
4.1 OVERVIEW	4-1
4.2 CONCERNS	4-1
4.3 CONCEPTS FOR ENTERPRISE VIEWPOINT	4-1
4.4 CHARACTERISTICS OF ENTERPRISE OBJECTS	4-2
4.5 TERMS FOR THE ENTERPRISE VIEWPOINT	4-5
4.6 EXAMPLE ENTERPRISE OBJECT TYPES	4-7
4.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH ENTERPRISE VIEWPOINT	4-7
4.8 ENTERPRISE ARCHITECTURE AND SYSTEMS ARCHITECTURE	4-10
4.9 SECURITY TOPICS IN THE ENTERPRISE VIEWPOINT	4-12
5 FUNCTIONAL VIEWPOINT	5-1
5.1 OVERVIEW	5-1
5.2 CONCERNS	5-1

CONTENTS (continued)

<u>Section</u>	<u>Page</u>
5.3 CONCEPTS FOR FUNCTIONAL VIEWPOINT	5-1
5.4 CHARACTERISTICS OF FUNCTIONAL OBJECTS	5-2
5.5 TERMS FOR FUNCTIONAL VIEWPOINT	5-5
5.6 EXAMPLE FUNCTIONAL OBJECT TYPES.....	5-5
5.7 EXAMPLES OF A SPACE SYSTEM DESCRIBED FROM THE FUNCTIONAL VIEWPOINT	5-7
5.8 EXAMPLE OF SPACE SYSTEM WITH INFORMATION MANAGEMENT INFRASTRUCTURE.....	5-9
5.9 SECURITY TOPICS IN THE FUNCTIONAL VIEWPOINT	5-10
6 PHYSICAL VIEWPOINT	6-1
6.1 OVERVIEW	6-1
6.2 CONCERNS	6-2
6.3 CONCEPTS FOR THE PHYSICAL VIEWPOINT	6-2
6.4 CHARACTERISTICS OF PHYSICAL OBJECTS.....	6-3
6.5 TERMS FOR PHYSICAL VIEWPOINT	6-7
6.6 TYPICAL OBJECTS IN THE PHYSICAL VIEWPOINT	6-8
6.7 EXAMPLES OF A SPACE SYSTEM DESCRIBED WITH PHYSICAL VIEWS.....	6-9
6.8 SECURITY TOPICS IN THE PHYSICAL VIEWPOINT	6-10
7 CONNECTIVITY VIEWPOINT—DERIVED	7-1
7.1 OVERVIEW	7-1
7.2 CONCERNS	7-1
7.3 CONCEPTS FOR THE CONNECTIVITY VIEWPOINT	7-2
7.4 CHARACTERISTICS OF CONNECTIVITY OBJECTS.....	7-2
7.5 TERMS FOR THE CONNECTIVITY VIEWPOINT	7-7
7.6 TYPICAL CONNECTIVITY OBJECTS	7-8
7.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH CONNECTIVITY VIEWS	7-10
7.8 SECURITY TOPICS IN THE CONNECTIVITY VIEWPOINT.....	7-14
8 STRUCTURAL VIEWPOINT—DERIVED	8-1
8.1 OVERVIEW	8-1
8.2 CONCERNS	8-1
8.3 CONCEPTS FOR THE STRUCTURAL VIEWPOINT.....	8-1
8.4 CHARACTERISTICS OF STRUCTURAL OBJECTS	8-2
8.5 TERMS FOR STRUCTURAL VIEWPOINT	8-4

CONTENTS (continued)

<u>Section</u>	<u>Page</u>
8.5 TYPICAL STRUCTURAL OBJECT TYPES	8-5
8.6 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH STRUCTURAL VIEW	8-6
8.7 SECURITY TOPICS IN THE STRUCTURAL VIEWPOINT	8-8
9 COMMUNICATIONS VIEWPOINT	9-1
9.1 OVERVIEW.....	9-1
9.2 CONCERNS.....	9-1
9.3 CONCEPTS FOR COMMUNICATIONS VIEWPOINT	9-2
9.4 CHARACTERISTICS OF COMMUNICATIONS OBJECTS	9-3
9.5 KEY OBJECTS AND RELATIONSHIPS	9-4
9.6 TERMS FOR THE COMMUNICATIONS VIEWPOINT.....	9-6
9.7 TYPICAL PROTOCOL ENTITIES	9-8
9.8 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH COMMUNICATIONS VIEWPOINT.....	9-10
9.9 SECURITY TOPICS IN THE COMMUNICATIONS VIEWPOINT	9-15
10 INFORMATION VIEWPOINT	10-1
10.1 OVERVIEW.....	10-1
10.2 CONCERNS.....	10-1
10.3 CONCEPTS FOR THE INFORMATION VIEWPOINT	10-1
10.4 CHARACTERISTICS OF INFORMATION OBJECTS	10-3
10.5 TERMS FOR THE INFORMATION VIEWPOINT	10-5
10.6 INFORMATION OBJECT TRANSFORMATIONS	10-6
10.7 EXAMPLE OF OBJECTS FOR THE INFORMATION VIEWPOINT	10-8
10.8 SECURITY TOPICS IN THE INFORMATION VIEWPOINT	10-10
11 SERVICE VIEWPOINT	11-1
11.1 OVERVIEW.....	11-1
11.2 CONCERNS.....	11-1
11.3 CONCEPTS FOR THE SERVICES VIEWPOINT	11-1
11.4 CHARACTERISTICS OF SERVICE OBJECTS	11-2
11.5 TERMS FOR SERVICE VIEWPOINT	11-5
11.6 TYPICAL SERVICE TYPES	11-6
11.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH SERVICES VIEWS	11-7
11.8 SECURITY TOPICS IN THE SERVICES VIEWPOINT.....	11-9

CONTENTS (continued)

<u>Section</u>	<u>Page</u>
12 OPERATIONS VIEWPOINT	12-1
12.1 OVERVIEW	12-1
12.2 CONCERNS FOR THE OPERATIONS VIEWPOINT	12-1
12.3 CONCEPTS FOR THE OPERATIONS VIEWPOINT	12-1
12.4 CHARACTERISTICS OF OPERATIONS OBJECT	12-2
12.5 TERMS FOR THE OPERATIONS VIEWPOINT	12-5
12.6 EXAMPLES OF OPERATIONS VIEWS	12-6
12.7 SECURITY TOPICS IN THE OPERATIONS VIEWPOINT	12-8
13 DERIVING OTHER VIEWS FROM THE BASIC VIEWPOINTS	13-1
ANNEX A NOTES ON USE OF RASDSV2 TO DO SYSTEMS	
ARCHITECTURE (INFORMATIVE)	A-1
ANNEX B FORMAL METHODS AND TOOLS (INFORMATIVE)	B-1
ANNEX C RASDSV2 AND SYSML EXAMPLES (INFORMATIVE)	C-1
ANNEX D ARCHITECTURE, MODELS, METAMODELS,	
AND FRAMEWORKS (INFORMATIVE)	D-1
ANNEX E GLOSSARY AND ACRONYMS (INFORMATIVE)	E-1
<u>Figure</u>	
2-1 Systems Engineering ‘Vee’	2-1
2-2 Systems Architecture vs Engineering	2-2
2-3 RASDSv2 Viewpoints, Concerns, and Objects	2-3
2-4 RASDSv2 Ontology: Concept Model, Objects, and Relationships	2-6
3-1 Icons Used in This Document	3-7
3-2 Unified Representation of Objects	3-8
3-2 Representation of Objects	3-8
3-3 Relationship Types among Objects (derived from UML)	3-9
3-4 Example of a RASDSv2 Viewpoint Ontology (Functional)	3-9
4-1 Attributes of Enterprise Objects	4-3
4-2 Ontology of Enterprise Objects	4-4
4-3 Representation of Enterprise Objects	4-5
4-4 Simple Example of a Single Mission Enterprise View	4-8
4-5 Example of an Enterprise View (Mars Exploration Federation)	4-9
4-6 Example of Multi-Agency Enterprise Launch View (Mission Z)	4-10
4-7 Enterprise Architecture Ontology (Adapted from TOGAF)	4-11
4-8 Enterprise and Technical Architecture Ontology Relationships	4-11
5-1 Overview of Functional Object	5-2

CONTENTS (continued)

<u>Figure</u>		<u>Page</u>
5-2	Ontology of Functional Objects	5-4
5-3	Representation of Functional Objects	5-4
5-4	Simple Example of a Functional View	5-7
5-5	Example of Functional View (Functional Objects, Provided Interfaces, and Data)	5-8
5-6	Representative Functional Objects and Information Management Infrastructure Elements	5-9
6-1	Physical Object Overview	6-3
6-2	Ontology for Physical Viewpoint.....	6-5
6-3	Physical Object Representation.....	6-6
6-4	Simple Example of Physical View Connector	6-9
6-5	Physical View Example Composition.....	6-9
7-1	Overview of Connectivity Object (Node)	7-3
7-2	Ontology for Connectivity Viewpoint.....	7-5
7-3	Representation of Connectivity Objects.....	7-6
7-4	Simple Connectivity View (Nodes for Some Mission).....	7-10
7-5	Connectivity View with Node Details	7-11
7-6	Connectivity View with Allocated Engineering Objects	7-12
7-7	Functional View of Image Compression.....	7-13
7-8(a)	Connectivity View of Software Compression Approach.....	7-13
7-8(b)	Connectivity View of Hardware Only Compression Approach.....	7-13
8-1	Ontology of Structural Objects	8-4
8-2	Representation of Structural Objects	8-6
8-3	Example View of a Hinge Assembly	8-7
8-4	A ‘Cartoon’ Conduit Hose Added to Figure 6-4.....	8-7
8-5	A ‘Cartoon’ Physical Bus and Articulated Antenna Added to Figure 6-5.....	8-8
9-1	Communication Object Overview.....	9-3
9-2	Ontology of Communications Viewpoint (Protocol) Objects.....	9-5
9-3	Representation of Communication Objects.....	9-6
9-4	Simple Example of an End-to-End Communications View.....	9-10
9-5	Example of a Return Communications View Showing Abstract Protocol Stack and Allocation to Nodes	9-12
9-6	Example of SSI End-to-End Communications View Showing Nodes.....	9-13
9-7	PDU Example, Space Packet Protocol	9-14
9-8	Example State Machine Diagram—SLE RCF	9-14
9-9	Example End-to-End Network (BPSec) and Application Layer (File Secure) Security Protocol Deployment	9-16
10-1	Overview of Information Objects	10-3
10-2	Information Object Ontology	10-4
10-3	Representation of Information Objects	10-5
10-4	Example of Information View Showing the Basic Object Semantics	10-7

CONTENTS (continued)

<u>Figure</u>	<u>Page</u>
10-5 Information Object Transformations.....	10-7
10-6 Example of Functional View with Representation of Information Objects.....	10-9
10-7 Example Information Model—Spacecraft Onboard Services.....	10-9
10-8 Information Object Formal Models.....	10-10
11-1 Service Viewpoint Overview	11-3
11-2 Ontology of Service Viewpoint.....	11-3
11-3 Service Interface Representation.....	11-4
11-4 Abstract Example of a Cross Support Service: Monitor Data	11-7
11-5 Aspects of ASL-Style Service: Mission Control	11-8
12-1 Service Object	12-3
12-2 Ontology of Operations Viewpoint.....	12-3
12-3 Operations Object Representation.....	12-4
12-4 Operations Viewpoint Core Model with Temporal Aspects (‘Schedule’).....	12-6
12-5 Simple Example of a Operations View	12-7
12-6 Operations Viewpoint Mission System Tasks and Data Flows	12-7
13-1 RASDSv2 in Wider Social Context	13-1
C-1 Example SysML Profile	C-1
C-2 Example of RASDSv2 Connectivity View Showing Composition of Abstract System Objects	C-2
C-3 Example of RASDSv2 Connectivity View Showing Abstract Interfaces, Communications Links, and Data Flows among Connectivity Objects.....	C-3
C-4 Example of RASDSv2 Connectivity View Showing Communication Flows among Connectivity Objects	C-4
C-5 Example of RASDSv2 Connectivity View Showing Allocation of System Elements to Different Physical Sites.....	C-5
C-6 Example of RASDSv2 High-level Connectivity View Providing Context for Interface Modeling	C-6
C-7 Example of RASDSv2 Communications View for Component Interface Modeling	C-7
C-8 RASDSv2 Abstract Model of a Protocol Entity Showing Behavior and Interfaces	C-8
C-9 RASDSv2 State Machine Model of TCP Protocol Connection Establishment Behavior.....	C-9
C-10 RASDSv2 Model of Abstract and Realized Information Objects.....	C-10
D-1 ISO 42010 Fundamental Concepts.....	D-3
D-2 RASDSv2 Related to Metamodel Levels.....	D-4
D-3 Different Viewpoints of a Space System	D-5

CONTENTS (continued)

<u>Table</u>		<u>Page</u>
4-1	Example Enterprise Objects	4-7
5-1	Example Functional Objects	5-6
5-2	Typical Infrastructure Objects.....	5-6
6-1	Example Physical Objects	6-8
7-1	Typical Nodes	7-8
7-2	Typical Links.....	7-10
8-1	Examples of Structural Objects.....	8-5
9-1	Typical Protocol Entities.....	9-8
11-1	Examples of Typical Service Entities	11-6

1 INTRODUCTION

1.1 SCOPE

This document describes a Reference Architecture for Space Data Systems (RASDS) that is intended to support modeling of space data systems and operations. The RASDS provides a standardized framework and methodology for modeling space system architectures and related high-level designs, which individual working groups may use within CCSDS, or in International Standards Organization (ISO) TC20/SC13 or ISO TC20/SC14, or for projects within the space agencies or other organizations adopting this Recommended Practice.

The extended approach in this revision has been renamed RASDSv2; it is specifically adapted for the space domain and is aligned with best current practices in the fields of system and software architecture and modeling. While this architecture modeling methodology is intended for use within CCSDS and ISO space systems, it is also suitable for use by mission and project design teams, to describe system architectures and designs within the space domain. It uses a document-based representation and does not propose any specific formal modeling method or tool, but with creation of suitable profiles it can be used with more formalized representations such as Unified Modeling Language (UML) or System Modeling Language (SysML). Examples of this are provided in an Annex.

1.2 PURPOSE

Within CCSDS and ISO TC20 the RASDSv2 will be used for the following purposes:

- a) to establish an overall CCSDS and ISO TC20 recommended methodology for developing and modeling domain-specific architectures;
- b) to define a common language, taxonomy, and set of representations so that challenges, requirements, and solutions in the area of space systems can be readily communicated;
- c) to provide a kit of architect's tools that domain experts may use to describe different specific complex space system architectures;
- d) to facilitate development of CCSDS and ISO TC20 Recommended Standards in a consistent way so that any standard can be used with other appropriate standards in a space system;
- e) to provide a framework and guidelines for presenting the architectural aspects of Recommended Standards developed by CCSDS and ISO TC20 in a systematic way so that their functionality, applicability, interrelationships, and interoperability may be clearly understood.

1.3 APPLICABILITY

The methodology described in this Recommended Practice may be used by agencies, projects, or individual working groups to create models of space system architectures in any relevant CCSDS, ISO TC20, or project documents.

It is important to keep in mind that not all viewpoints are needed for every task. Only those viewpoints that are required for a given purpose need to be used. In many instances, only the Functional and Connectivity Viewpoints may be needed (see 2.4 on selecting specific viewpoints for a given task). The methodology provided in this document may be used in describing the architectures of individual system elements, entire mission space systems, or systems of systems.

New views that align with this framework may also be created as needed, and alternative representations may be adopted where they improve alignment with current practices in specialized subdomains.

The representations that are adopted in this document use simple drawing tools, but the viewpoints, views, terminology, and associated methodology can and have been directly adopted for use with the UML and SysML representations that are provided by typical Model Based Systems Engineering (MBSE) tools (see annex C for examples).

As a Recommended Practice, this document provides a tool for CCSDS and ISO TC20. Its use is encouraged in all CCSDS and ISO TC20 space systems documents where system or reference architecture descriptions are provided, and where the analytical and descriptive methodologies provided in this document will be useful.

1.4 RATIONALE

Several different standard methods that are currently available for the description of software-intensive systems architectures were analyzed.¹ These all share the concepts of developing a consistent set of terminology and modeling elements, and also a relevant set of viewpoints, views, and specifications, with which to describe systems and their architectures.

All these typical methods assume that the elements of these systems are fixed in place or move over or near the surface of the Earth and that they are typically in continuous, instantaneous, communication over what are nominally error-free communications channels

¹ These include the ISO Reference Model of Open Distributed Systems (RM-ODP, reference [1]), the IEEE Recommended Practice for Architectural Descriptions of Software-Intensive Systems (IEEE 1471-2000, reference [2]), Software, Systems, and Enterprise - Architecture Description (IEEE/ISO/IEC 42010-2022, reference [14]), the Standard for Application and Management of the System Engineering Process (IEEE 1220-2005, reference [3]), OMG Unified Modeling Language (UML, references [6], [7], and [8]), Systems Modeling Language (SysML, references [9] and [10]), DoD Architecture Framework (DoDAF, reference [11]), the Open Group Architecture Framework (TOGAF, reference [12]), the Unified Architecture Framework (UAF) (reference [32]), the foundational ISO Basic Reference Model (ISO-BRM, reference [13]), and others.

that suffer only occasional disruptions. The physical environment in which many systems operate is often given only broad consideration.

Systems operating in space tend to violate many of these basic assumptions for terrestrial systems and have particular concerns associated with orbital/planetary distances and the effects of the space plasma environment. Space elements may only occasionally be in contact with one another, typically require use of very expensive and over-subscribed ground and space communications assets, are strongly affected by the physical environment in which they have to operate, and usually cannot easily be repaired or replaced. These environmental issues affect what must be done to provide reliable communications between elements, how control interactions must be designed, and how these systems must be developed, launched, tested, and operated.

RASDSv2 provides a methodology and an abstract model for the description of the functionality, physical deployments, communication, structures, and operations of space systems that accounts for the realities of operating in the space environment. This is a domain-specific architectural approach adapted to the requirements of space systems.

Several separate viewpoints may be required in any given architecture description, depending upon the specific needs of a project. The viewpoints that the original CCSDS RASDS document defined were adapted for the architectural description of systems in space: Enterprise, Functional, Connectivity, Communications, and Information. This RASDSv2 version adds three other viewpoints: Services, Operational, and Physical (a superset of Connectivity), to support the needs of CCSDS and ISO TC20/SC14. Other viewpoints may be identified as needed, and this core set may be extended.

This is intended to be a pragmatic and useful document, including a set of straightforward drawing conventions that have been chosen to ensure that the diagrams for any view can be unambiguously interpreted. The objects that appear in any viewpoint have concrete and clearly documented representation, syntax, and semantic elements. This document defines and adopts a consistent methodology that can form the basis for a more formal MBSE representation of space systems such as might be developed in a UML (reference [6]) or SysML (reference [9]) tool. Such formal representations using SysML have been developed for various projects. More discussion of this topic is provided in annex C.

RASDSv2 is consistent with the current version of ISO 42010-2022 (reference [14]), *Software, Systems, and Enterprise—Architecture Description*, and earlier architecture references [1] and [2]. While these ‘meta-architecture’ documents describe ‘core terms, definitions and relationships for the Architecture Description’, define the meaning of terms like viewpoint and view, and identify relevant software architecture frameworks, they do not provide specific instances of such viewpoints.

The RASDSv2 methodology leverages this ISO 42010 meta-metamodel and provides concrete descriptions and specific viewpoints suitable for a wide variety of system architecture descriptions in the space system domain. For those who are interested, the derivation of this meta-model is documented in annex B.

1.5 DOCUMENT STRUCTURE

Section 2 provides an overview of RASDSv2 in a narrative style and briefly introduces the key architectural concepts and the set of viewpoints that are used.

Section 3 introduces and describes basic concepts and defines specific terms, elements, viewpoints, objects, and representation styles that are used throughout this document.

In sections 4 through 12 of RASDSv2 documents in detail each of the viewpoints introduced in section 2, formalizes each object and its interfaces, concepts, representation, special terms, and gives practical examples of how to use it.

Section 13 presents an example of how to extend RASDSv2, using the basic concepts, to describe new views on a system.

Annexes A–E provide additional information on use of RASDSv2, its relationship to other methods and models, including MBSE, UML and SysML, and a summary of the terminology and acronyms used in the document.

1.6 REFERENCES

The following publications contain provisions which, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were valid. All publications are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the publications indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

- [1] *Information Technology—Open Distributed Processing—Reference Model: Architecture—Part 3*. 2nd ed. International Standard, ISO/IEC 10746-3:2009. Geneva: ISO, 2009.
- [2] *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE Std 1471-2000. New York: IEEE, 2000.
- [3] *IEEE Standard for Application and Management of the Systems Engineering Process*. IEEE Std 1220-2005. New York: IEEE, 2005.
- [4] *Security Architecture for Space Data Systems*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 351.0-M-1. Washington, D.C.: CCSDS, November 2012.
- [5] *Reference Architecture for Space Information Management*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 312.0-G-1. Washington, D.C.: CCSDS, March 2013.
- [6] *OMG® Unified Modeling Language (OMG® UML)*. Version 2.5.1. formal/2017-12-05. Needham, Massachusetts: Object Management Group, December 2017.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

- [7] *OMG Unified Modeling Language™ (OMG UML), Infrastructure*. Version 2.4.1. formal/2011-08-05. Needham, Massachusetts: Object Management Group, August 2011.
- [8] *OMG Meta Object Facility (MOF) Core Specification*. Version 2.5.1. formal/2019-10-01. Needham, Massachusetts: Object Management Group, October 2019.
- [9] *OMG Systems Modeling Language (OMG SysML™)*. Version 1.6. formal/19-11-01. Needham, Massachusetts: Object Management Group, November 2019.
- [10] *SysML Extension for Physical Interaction and Signal Flow Simulation*. Version 1.0. formal/18-05-03. Needham, Massachusetts: Object Management Group, June 2018.
- [11] “The DoDAF Architecture Framework Version 2.02.” 2.02, August 2010. Chief Information Officer, U.S. Department of Defense.
<https://dodcio.defense.gov/Library/DoD-Architecture-Framework/>. (4/15/2024)
- [12] *The Open Group Architecture Framework (TOGAF)*. Version 10.0 Enterprise Edition. San Francisco: The Open Group, April 2022.
- [13] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. 2nd ed. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.
- [14] *Software, Systems, and Enterprise—Architecture Description*. 2nd ed. ISO/IEC/IEEE 42010:2022. Geneva: ISO, 2022.
- [15] *Overview of Space Communications Protocols*. Issue 4. Report Concerning Space Data System Standards (Green Book), CCSDS 130.0-G-4. Washington, D.C.: CCSDS, April 2023.
- [16] *The Application of Security to CCSDS Protocols*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 350.0-G-3. Washington, D.C.: CCSDS, March 2019.
- [17] *Security Threats against Space Missions*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 350.1-G-3. Washington, D.C.: CCSDS, February 2022.
- [18] *Space Communications Cross Support—Architecture Requirements Document*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 901.1-M-1. Washington, D.C.: CCSDS, May 2015.
- [19] Peter Shames and Joseph Skipper. “Toward a Framework for Modeling Space Systems Architectures.” In *Proceedings of SpaceOps 2006 (19 June 2006–23 June 2006, Rome, Italy)*. Reston, Virginia: AIAA, 2006.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

- [20] Mark W. Maier. “Architecting Principles for Systems-of-Systems.” *Systems Engineering* 1, no. 4 (February 1999): 267-284.
- [21] *CCSDS Bundle Protocol Specification*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 734.2-B-1. Washington, D.C.: CCSDS, September 2015.
- [22] *Operations Concept for a Solar System Internetwork (SSI)*. IOAG.T.RC.001.V1. Washington, DC: IOAG, 15 October 2010.
- [23] *Space Packet Protocol*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 133.0-B-2. Washington, D.C.: CCSDS, June 2020.
- [24] *Space Link Extension—Return Channel Frames Service Specification*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 911.2-B-4. Washington, D.C.: CCSDS, July 2023.
- [25] *Cross Support Transfer Services—Monitored Data Service*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 922.1-B-2. Washington, D.C.: CCSDS, September 2022.
- [26] *Service Oriented Architecture Modeling Language (SoaML) Specification*. Version 1.0.1. formal/2012-05-10. Needham, Massachusetts: Object Management Group, May 2012.
- [27] *Application and Support Layer Architecture*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 371.0-G-1. Washington, D.C.: CCSDS, November 2020.
- [28] Peter M. Shames, Marc A. Sarrel, and Sanford Friedenthal. “A Representative Application of a Layered Interface Modeling Pattern.” In *Proceedings of the 26th Annual INCOSE International Symposium (IS 2016) (July 18–21, 2016, Edinburgh, Scotland)*. San Diego, California: INCOSE, 2016.
- [29] Shames, Peter, et al. “NASA Integrated Network Monitor and Control Software Architecture.” In *Proceedings of SpaceOps 2012 (June 11–15, 2012, Stockholm, Sweden)*. Reston, Virginia: AIAA, 2012.
- [30] Charles Kohlhase, ed. *The Voyager Neptune Travel Guide*. JPL Publication 89-24. Pasadena, California: JPL, 1989.
- [31] “ISO/IEC/IEEE 42010: Conceptual Model.” ISO/IEC/IEEE 42010. <http://www.iso-architecture.org/42010/cm/>. (4/16/2024)
- [32] *Unified Architecture Framework (UAF) Domain Metamodel*. Version 1.2. formal/22-07-03. Needham, Massachusetts: Object Management Group, July 2022.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

- [33] Yaniv Mordecai. “Model-Based Protocol Specification.” *Systems Engineering* 22, no. 2 (March 2019): 188–210.
- [34] *NASA Procedural Requirements*. NPR 7123.1D. Washington, DC: NASA, 2023.
- [35] *Information Technology—Database Languages SQL*. 6th ed. ISO/IEC 9075:2023. Geneva: ISO, 2023.
- [36] Steve Harris and Andy Seaborne, eds. “SPARQL 1.1 Query Language.” Version 1.1, 21 March 2013. W3C. <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>. (4/16/2024)
- [37] *Space Assigned Numbers Authority (SANA)—Role, Responsibilities, Policies, and Procedures*. Issue 3. CCSDS Record (Yellow Book), CCSDS 313.0-Y-3. Washington, D.C.: CCSDS, October 2020.
- [38] *CCSDS SANA Registry Management Policy*. Issue 2. CCSDS Record (Yellow Book), CCSDS 313.1-Y-2. Washington, D.C.: CCSDS, October 2020.
- [39] *Spacecraft Onboard Interface Services—Specification for Dictionary of Terms for Electronic Data Sheets*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 876.1-M-1. Washington, D.C.: CCSDS, March 2024.
- [40] “OWL 2 Web Ontology Language Document Overview.” W3C Recommendation. 2nd ed., 11 December 2012. <http://www.w3.org/TR/xinlude/>.
- [41] “Protégé.” Stanford University. <https://protege.stanford.edu/>. (4/16/2024)
- [42] *Business Process Model and Notation (BPMN)*. Version 2.0.2. formal/2013-12-09. Needham, Massachusetts: Object Management Group, December 2013.
- [43] *Security Architecture for Open Systems Interconnection for CCITT Applications*. Recommendation X.800. Geneva: ITU, 1991.
- [44] *Baseline Capabilities for Enhanced Global Identity Management and Interoperability*. Recommendation ITU-T X.1250. Geneva: ITU, September 2009.
- [45] *Spacecraft Onboard Interface Services—XML Specification for Electronic Data Sheets*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 876.0-B-1. Washington, D.C.: CCSDS, April 2019.
- [46] *Systems and Software Engineering—System Life Cycle Processes*. 2nd ed. ISO/IEC/IEEE 15288:2023. Geneva: ISO, 2023.
- [47] J. Postel. *Transmission Control Protocol*. STD 7. Reston, Virginia: ISOC, September 1981.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

- [48] *OMG Systems Modeling Language™ (SysML®)—Part 1: Language Specification*. Version 2.0 Beta 2. Needham, Massachusetts: Object Management Group, February 2024.
- [49] Jean-Luc Voirin. *Arcadia Language Reference: Meta Model*. La Défense, France: Thales, 2023. <https://mbse-capella.org/resources/arcadia-reference/Arcadia%20Language%20-%20MetaModel.pdf>
- [50] “Equivalences and Differences between SysML and Arcadia/Capella.” Capella | Open Source MBSE Tool. https://mbse-capella.org/arcadia_capella_sysml_tool.html.

2 OVERVIEW

2.1 GENERAL

This overview section introduces the basic RASDSv2 modeling terminology and concepts in a narrative form. Many of these terms will be familiar, but some have special meanings in RASDSv2. Annex E provides a comprehensive glossary that, can be consulted as needed for specific definitions of terms.

2.2 RELATING ARCHITECTURE AND SYSTEMS ENGINEERING

It is generally understood that all systems (like all buildings) have an architecture, even if that may not always be well documented. It also appears to be well understood that the more complex the system the more it is useful, if not essential, to document and analyze the architecture before starting detailed system design. In fact, it is frequently the case that some sort of architecture views will be constructed during early design phases, even if there is not a formal system architecture created, reviewed, and approved as part of the process.

Interestingly, the classical ‘Systems Engineering Vee’ is often shown without any reference to systems architecture. (See figure 2-1, which is borrowed from the NASA Systems Engineering [SE] Handbook, reference [34]). This ‘Vee’, and variations on it, ‘Dual V’, ‘W’, appear in many related processes, including Agile development, and but that only some of these explicitly include systems architecture as a formal phase in the process. The ISO Systems Engineering Lifecycle Processes, ISO/IEC 15288:2002 (reference [46]), provides a simplified Systems Architecture Design Process in subclause 5.5.4.

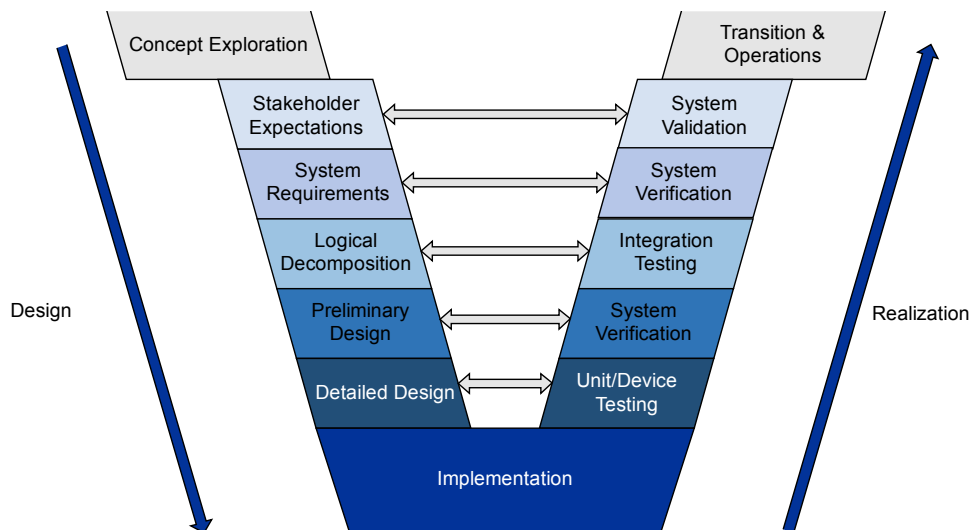


Figure 2-1: Systems Engineering ‘Vee’²

² From the NASA SE Process NPR 7123.1D.

In the NASA Systems Engineering Vee, systems architecture is referenced at essentially the phase called Logical Decomposition and Preliminary Design. The Logical Decomposition step is defined as:

The decomposition of the defined technical requirements by functions, time, and behaviors to determine the appropriate set of logical and data architecture models and related derived technical requirements. Models may include functional flow block diagrams, timelines, data control flow, states and modes, behavior diagrams, operator tasks, system data, metadata, data standards, taxonomy, and functional failure modes.

So ‘functions and behaviors’, and ‘logical and data architecture models’, are an intended outcome of the logical decomposition process, but there is no stated process, features, nor conformance criteria for such a set of products. By contrast, The Open Group Architecture Framework (TOGAF) (reference [12]), shows several stages of architecture development (vision, business, information systems, and technology architectures, embedded in an enterprise metamodel) all feeding into a central requirements management process and then the design and development processes.

In the abstract, the relationship between systems architecting and systems engineering might be represented as shown in figure 2-2.

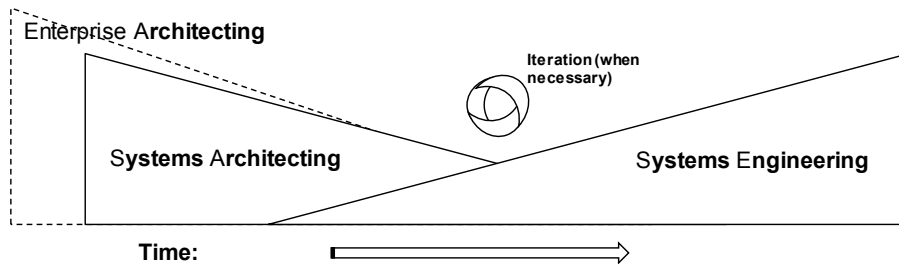


Figure 2-2: Systems Architecture vs Engineering

This discussion of the differences between architecting and engineering aside, what this document presents is an entirely practical methodology, one that is grounded in formalisms, but is suitable for direct application by working systems architects on their current projects.

2.3 OVERVIEW OF RASDSv2 VIEWPOINTS

2.3.1 GENERAL

Each viewpoint is a particular perspective on the specification of a complete system, established to model those aspects of a system relevant to the identified area of concern during the design of the system. The viewpoints are intentionally independent to simplify reasoning about the complete specification. Mutual consistency among the viewpoint specifications is ensured by the architecture descriptions defined by RASDSv2 and the use of an integrated object model that describes their elements and relationships. (See 3.2.5 for formal definitions of all of these architecture terms.)

Figure 2-3 provides an overview of all the defined viewpoints, the concerns that they each address and the kinds of objects that they are used to represent.

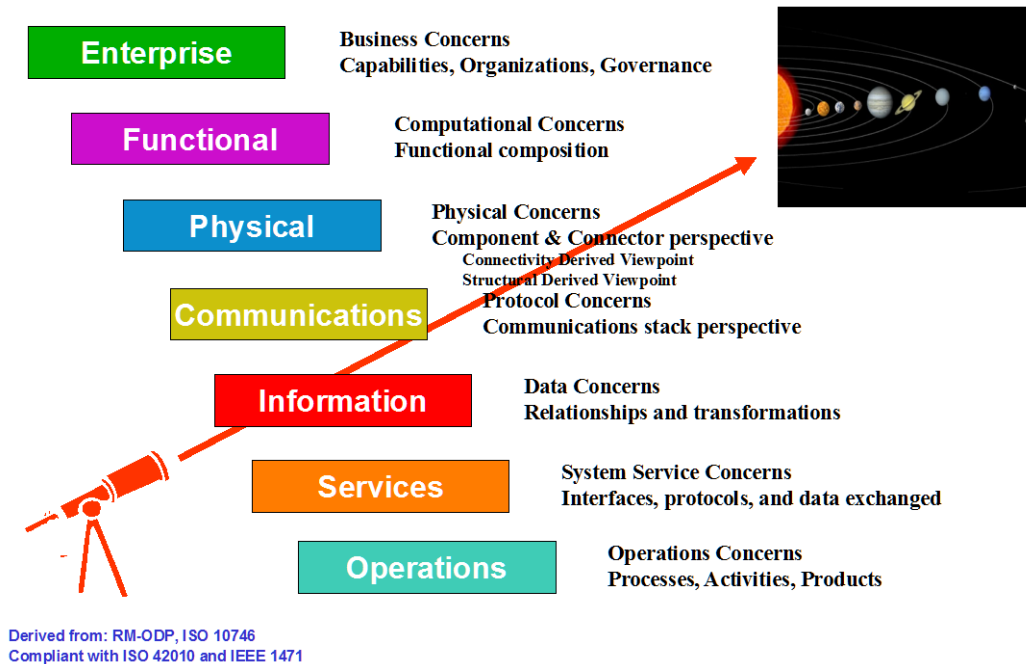


Figure 2-3: RASDSv2 Viewpoints, Concerns, and Objects

This figure introduces a specific set of color codes that will be referenced in the definitional object and ontology diagrams that are included in each viewpoint. Elsewhere in RASDSv2 there is no special meaning assigned to colors, but the introductory diagrams in each viewpoint of this document use these colors to distinguish elements that have their ‘home’ in a given viewpoint and those that are referenced, by correspondence from other viewpoints. These concepts are introduced formally in the RASDSv2 conceptual ontology, figure 2-4, and in related foundational ontology diagrams that occur in each viewpoint section.

2.3.2 RASDSv2 VIEWPOINT FEATURES

The RASDSv2 framework provides seven specific and complementary viewpoints on the system and its environment, along with two derived viewpoints:

- The **Enterprise Viewpoint** focuses on the purpose, scope, and policies of a space system. It can be used to describe the organizational entities and relationships; their roles, requirements, goals, objectives, scenarios, constraints; and how to meet them.
- The **Functional Viewpoint** describes the functional decomposition of a space system into abstract objects that interact at interfaces. It describes the functionality provided by the space system, the behavior of the functional elements, and their functional decomposition.

- The **Physical Viewpoint** describes the engineered decomposition of a space system into physical components and their connections, and the external environment within which it operates. It describes the physical deployment and environmental aspects of space system elements, and the physical forces and behavior (motion, radiation, gravity) acting on the components. The connections may be manifestly material (electrical, weldments, bolts, mating surfaces, joints), or they may be more energetic (radiation, thermal, magnetic, gravitational). This viewpoint ‘owns’ the physical elements, but different aspects may be analyzed in derived viewpoints, such as Connectivity and Structural.
 - The **Connectivity Viewpoint** is derived from the Physical Viewpoint. It specifically addresses the engineered decomposition of the space system into components (often referred to as nodes) that communicate across connectors (links). The Connectivity Viewpoint describes the communications aspects of the physical deployment of the space system. The links may be manifestly material (network or data cables), or they may be more energetic Radio Frequency [RF] and optical signals). The Connectivity Viewpoint may also be used to address the allocation of implemented functions (as engineered software or hardware objects) to these nodes.
 - The **Structural Viewpoint** is also derived from the Physical Viewpoint. It specifically addresses the engineered decomposition of the space system into components that are physically connected one to another. The Structural Viewpoint describes the physical deployment and connection aspects of the space system, its physical decomposition, and its interactions with the rest of the physical environment within which it operates. These structural connections are manifestly material: bolts, weldments, joints (fixed, flexural, or rotational), mating surfaces, etc.
 - Other derived Physical Viewpoints: There may be other physical, or energetic, aspects of space system design that can be handled directly in the Physical Viewpoint or in other viewpoints derived from it. Derived viewpoints may be required for different energetic exchanges, such as thermal, gravitational, or electrostatic.
- The **Communications Viewpoint** focuses on the mechanisms and functions required to engineer, document, and implement the protocols, protocol stacks, and communications standards for a space system. This includes implementation choices and specifications, protocol stack choices, and allocation of this communications functionality to engineered components of the system. This viewpoint is treated separately in RASDSv2 because it is essential for describing how end-to-end communications are designed and handled in space systems.
- The **Information Viewpoint** focuses on the kinds of information handled by the system, the structure and semantics of the information, and the interpretation of that information. It describes the information managed by the space system along with the structure, content, semantics, type, relationships, and constraints on the data used within the system.

- The **Service Viewpoint** is an aggregated view that focusses on the kinds of services provided by the system, the functions exposed at the system interfaces, the operations provided by the services, the kinds of information exchanged across these interfaces, and details of the interface bindings. It describes the exposed behavior at the space system interface. The details of the Information Objects exchanged at the interface are defined by correspondence in a related Information View and the details of the protocols are defined by correspondence in a related Communications View.
- The **Operations Viewpoint** focuses on the kinds of operations supported by the system, the processes, procedures, activities, and the kinds of operational behaviors carried out by the system, whether carried out by people or systems elements. The details of any Information Objects exchanged at operational interfaces are defined by correspondence in a related Information View. The details of the systems elements are described in a corresponding Functional or Connectivity View.

2.3.3 CONCEPTUAL MODEL AND CORRESPONDENCES BETWEEN VIEWPOINTS

Each of the RASDSv2 viewpoint specifications is intended to be orthogonal to the others, and a description of a given system from any one viewpoint should be self-consistent. Additionally, many of the objects defined in their ‘home’ RASDSv2 viewpoint will also have identifiable relationships or correspondences with objects defined in other viewpoints. This is an essential element of the methodology. In figure 2-4 the relationships among the core set of objects defined in RASDSv2 are shown in a conceptual model, using a simple form of graphical ontology.

In figure 2-4 only the top-level (or core) objects and attributes in RASDSv2 are shown, those most central to understanding how the core objects from all the different viewpoints are related. Each of these core objects represents a class of objects, as will be described in the following sections. These classes of objects may have subclasses, and all have an associated set of attributes. The full sets of attributes for each of these objects and their classes are shown in the more detailed ontology diagrams that appear in each of the later sections of this document. Finally, figure 2-4 captures only the static relationships among objects; it does not capture any of the dynamic behavior of objects, in either a functional or physical sense.

The element named ‘Metamodel’ in the upper right-hand corner of figure 2-4 represents the RASDSv2 viewpoint specifications, views, and associated user concerns. Each viewpoint specification can be thought of as a perspective on a system that defines the objects and rules for constructing views, and each permits only a subset of objects and representations relevant for a given concern to be analyzed. Each of these top-level objects is defined in a ‘home’ viewpoint in RASDSv2, but many of them will have representations or correspondences in other viewpoints.

Thus the ‘home’ viewpoint for Information Objects is the Information Viewpoint, but representations of Information Objects will often appear in the Enterprise, Functional, and Connectivity Viewpoints. Similarly, abstract Functional Objects defined in the Functional

Viewpoint have correspondences with implemented engineering objects (hardware or software) in the Connectivity Viewpoint. Communications Objects, in the form of protocol stacks, are often shown in correspondence with Connectivity Viewpoint objects (nodes) and the links that connect them.

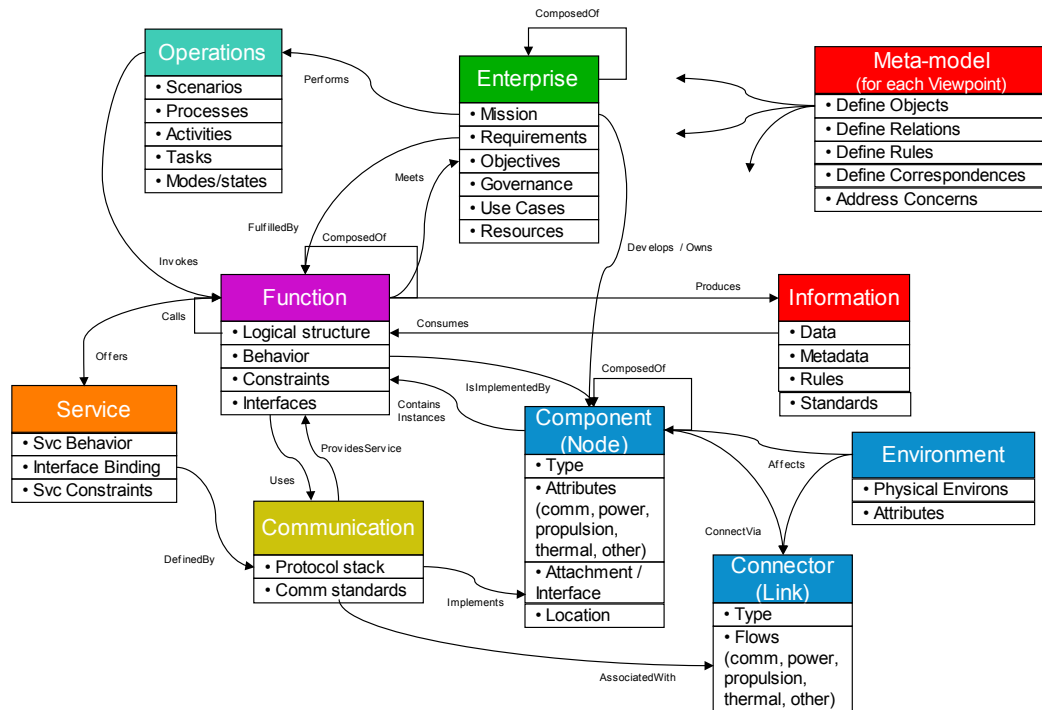


Figure 2-4: RASDSv2 Ontology: Concept Model, Objects, and Relationships

Any set of RASDSv2-compliant system specifications that use different viewpoint specifications should not contain contradictory statements; that is, the views should be mutually consistent. Thus, a complete specification of a system can include descriptions of the correspondences relating elements in one view to elements in another view, and this helps to ensure that the model is consistent. The minimum requirement for consistency in a set of RASDSv2 specifications for a system is that they adhere to the rules for the selected viewpoints and embody within the set of specifications the relevant correspondences defined in this reference architecture.

In many of the examples used in this document, color is used to distinguish different classes of objects. In general, RASDSv2 does not assign any particular meaning to the use of color, and the user is free to employ color and other ‘decorations’ wherever it assists in clarifying the representation of the architecture or bringing consistency to a set of diagrams.

There is, however, one specific use of color, and that is in the viewpoint definitions that are introduced starting in figure 2-3 and elaborated upon in the object ontology in figure 2-4. These colors are treated as a ‘color key’ in all the foundational object definitions and individual object ontologies that appear in sections 4 through 12. This is the only defined use of a color key in this document. In the rest of the document, colors are used as ‘decoration’ without any special defined meaning.

2.4 SELECTING RASDSv2 VIEWPOINTS FOR A GIVEN USE

To describe the architecture of a particular space system, RASDSv2 defines seven primary viewpoint specifications, and two derived ones, as just described. The user must decide which of these viewpoints will be of value to describe a particular space system, and if that system can be characterized with fewer than all seven viewpoints. Often only two or three viewpoints are needed to define simple system architectures. More complex and thorough architecture descriptions may require use of the full set. Depending on the complexities of the architecture, multiple perspectives of the same viewpoints may also be required, potentially with different levels of decomposition or different kinds of views.

Different users may find that one specific viewpoint provides them with a most familiar or useful ‘point of entry’. For instance, a data system application designer might favor the Functional Viewpoint, where a structural engineer might favor the Physical Viewpoint, and the end-to-end information system engineer might favor the Communications Viewpoint as a starting point. What RASDSv2 provides is a methodology that helps relate all these different viewpoints to provide an integrated whole.

Sometimes a viewpoint specification will contain references to objects that are representations of related objects in another viewpoint. An example of this is the Physical Viewpoint that is newly introduced in RASDSv2. In RASDSv2 the Physical Viewpoint ‘owns’ all the physical, three-dimensional, objects in the system, the environment within which (and with which) they interact, and all their physical and environmental attributes. However, different stakeholders will typically have different concerns and focus on different physical aspects: the structural engineer is concerned with the masses, centers of gravity, strength, and structural/flexural properties of connected components; but the communications engineer is concerned with the flows of data, whether using cables (which also have mass) or RF or optical transmissions, and the thermal engineer is concerned with the flow of heat.

The basic set of Physical Objects is the same, but the aspects that are considered and their attributes differ. In RASDSv2 the Physical Viewpoint has been identified as the core viewpoint for Physical Objects, with the expectation that users will adopt one or more derived viewpoints: structural, communications, thermal, power, as needed for their specific purposes. The core Physical Object attributes are intended to be referenced, via correspondence, in any of the derived viewpoints that are required for a given system model.

If it is impossible to capture all the important aspects of the system with the objects and viewpoints described in these viewpoint specifications, the user may define a new viewpoint specification by leveraging the basic concepts and approaches described in section 3. An example of how this might work in the space operations domain is provided in section 13, in which RASDSv2 has been used as an architecture framework and then expanded to encompass considerations of project lifecycle and physical scope.

2.5 SECURITY VIEWS IN RASDSv2

The CCSDS publication guidelines mandate that security topics be addressed in CCSDS Blue and Magenta Books, and these are to include the following information: security background/introduction; statements of security concerns with respect to the CCSDS document; data privacy and integrity; authentication of communicating entities; control of access to resources; availability of resources; auditing of resource usage; potential threats and attack scenarios, consequences of not applying security to the technology (e.g., loss of life, loss of mission). All of these are aspects that must be assessed when considering security assurance. Security assurance is the degree of confidence one has that the security measures, both technical and operational, work as intended to protect the system and the information it processes.

There is not a single ‘security viewpoint’, in RASDSv2 largely because security is such a cross-cutting topic, and it potentially touches on most of the defined viewpoints. Accordingly, this document provides guidance to users on how to describe security approaches within the context of system architecture, and each viewpoint provides the means to address the security topics that are relevant in that viewpoint. Security topics pertinent for each viewpoint are explicitly identified and briefly addressed in each section. Detailed explanations of security topics and approaches are treated separately in the CCSDS Security Architecture (reference [4]) and in related security documents (references [16] and [17]).

3 RASDSv2 MODELLING CONCEPTS

3.1 GENERAL

This section defines in a more formal way the concepts and terms for systems architecture modeling that are used throughout this document. The RASDSv2 is, at its core, a model-based systems engineering approach to specifying system architectures. Modeling using objects as part of developing a systems architecture provides a formalization that uses well-established abstraction and encapsulation design practices that are familiar from structured programming and that are adopted in most MBSE methods.

These abstractions specifically allow the descriptions of system functionality to be separated from the details of system implementation.

Encapsulation allows the hiding of the mechanisms of service provision from the service user, the hiding of design heterogeneity, the localization of interaction points, and the implementation of security.

The object modeling concepts cover:

- basic model features, providing rigorous definitions for the core set of concepts (object, interface, action, and interaction) that form the basis for RASDSv2 system descriptions and are applicable in all viewpoints;
- specification concepts that address notions such as object type and class, which are necessary for reasoning about objects and the relationships among objects, providing general tools for design, and establishing clear viewpoint specification languages;
- structuring concepts that build on the basic model features and the specification concepts to provide useful viewpoints from which to describe space system architectures, address recurrent structures that appear in distributed systems, and cover such concerns as role, behavior, capability, and communication.

3.2 DEFINITIONS

3.2.1 OVERVIEW

The following concepts and terms are used commonly in the viewpoints presented in sections 4 through 11. In each of the sections that present viewpoint specifications, any definitions that are essential for that viewpoint may be repeated, and definitions that are used only in that viewpoint will be provided.

3.2.2 BASIC ELEMENTS³

An **entity** is any concrete or abstract thing of interest. For example, an entity may be a physical instrument, a computer, a piece of software, or a set of functions performed by a system.

NOTE – In general, the word ‘entity’ can be used to refer to anything, but in the context of modeling it is reserved to refer to things in the universe of discourse being modeled.

An **element** is a constituent part of something; any thing that is one of the individual parts of which a composite entity is made up; an identifiable component, process, or entity of a system.

Abstraction is a mechanism and practice to reduce and factor out details so that one can focus on a few related concepts at a time. It is the process of extracting the underlying essence of a concept, removing any dependence on real-world objects with which it might originally have been connected, and generalizing it so that it has wider applications.

An **object** is an abstract model of an entity in the real world. It contains information, has behavior, and may offer services. A system is composed of interacting objects. An object is characterized by that which makes it distinct from other objects.

3.2.3 PROPERTIES OF ELEMENTS

A **type** specifies the set of values allowed and the primitive operations that an object can provide. Types are grouped into classes, which share the same primitive operations.

An **attribute** is a characteristic of an object, that is, a construct that system designers use to add additional information to system elements (e.g., objects, modules, types) to define their functionality.

An **action** is something that happens within an object, either with or without participation of another object. An **interaction** is an action performed by an object with participation of another object or with its environment.

A **behavior** is a set of actions performed by an object for some purpose.

³ The definitions in this section, and others, have been derived from ISO 42010 (reference [14]), RM-ODP (reference [1]), TOGAF Enterprise Architecture (reference [12]), and other well recognized sources, but integrated into a self-consistent whole. In common with MBSE & RM-ODP, all the viewpoints in RASDSv2 are defined in terms of objects, but this terminology is not strictly the same as ‘object oriented’ design terminology. In this key instance, and in many others, the reader is cautioned to attend to the specific definitions provided for terms used in RASDSv2.

An **activity** is a specification of behavior described as a sequence of actions.

A **constraint** is a limitation or implied requirement that limits the design solution or implementation, is not changeable by the enterprise, and is generally nonallocable.

An **interface** is a set of interactions provided by an element for participation with another object for some purpose, along with constraints on how they can occur.

NOTE – An interface represents a set of mechanical, electrical, signal, or other properties that describe some aspect of an element's connection to, or interaction with, another element.

Configuration describes a collection of objects able to interact at interfaces. A configuration determines the set of objects involved in each interaction along with constraints on their interactions.

A **service** is a provision of an interface of an element to support actions of another element.

A **service interface** is a mechanism to enable access to a set of one or more functions of an element, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.

Relationship describes the way that two or more entities can be associated with one another.

Structure refers to the relationships among a set of elements that contribute to the properties of the whole and enable them to interact.

A **role** describes the way in which an entity participates in a relationship, an object's set of behaviors and actions associated with the relationship of that object with other objects.

Syntax is the grammar defining the valid set of symbols and well-formed linguistic constructs of a language.

Semantics are the rules by which syntactic expressions and model elements are assigned meaning.

Interoperability⁴ refers to the technical capability of two or more systems or components to exchange information and to use the information that has been exchanged.

⁴ Multiple degrees of interoperability are possible, ranging from basic Physical Layer (e.g., frequency, modulation, and coding) compatibility up to Network and full Application Layer/service information exchange.

3.2.4 LOGICAL AND PHYSICAL ELEMENTS

An **objective** is something to be done or achieved. Objectives tend to be precise, tangible, and concrete.

A **goal** is an aim or purpose toward which effort may be directed. Goals tend to be broad or abstract and to state general intentions.

A **function** is the set of actions or activities performed by some element to achieve a goal. The transformation of inputs to outputs may include the creation, modification, monitoring, or destruction of elements.

A **Logical Object** is an abstract entity that may be considered separately from any particular implementation or deployment. It has no physical manifestation except as part of a model, but it may have associated behaviors and interfaces. Enterprise, Functional, and Information Objects are all logical objects.

A **logical link** is the locus of relations among Logical Objects. It may be considered separately from any particular implementation or deployment and has no physical manifestation except as part of a model.

NOTE – A logical object interacts with other objects over a logical link. A Physical Object (component) interacts with other objects using some physical link (connector).

An **aggregation** is several things grouped together or considered as a whole: aggregation is also the act of gathering things together.

Composition is a form of aggregation. Composition may be recursive.

A **composite object** is an object composed of two or more objects via aggregation. The behaviors of the composite object are determined by those of the objects that it aggregates.

A **location** is a point or extent in space.

The **environment** is a complex of external factors that acts on a system and determines its course and form of existence. The environment of some system or object consists of the substances, circumstances, objects, influences, or conditions by which it is surrounded or in which it occurs.

NOTE – An environment may be thought of as a superset, of which the given system is a subset. An environment may have one or more parameters, physical or otherwise.

A **resource** is anything available to a system that can support the achievement of objectives; any physical or virtual element that may be of limited availability within a system, such as hardware, software, programs, information, data, and other devices that are in use within or connected to a given system.

3.2.5 SYSTEMS ARCHITECTURE TERMS (FROM ISO 42010)

Architecture is the concepts and rules that define the structure, semantic behavior, and relationships among the parts of a system in its environment. It includes the elements (models of entities) that compose the system, the relationships among the elements, the constraints that affect those relationships, a focus on the parts of the system, and a focus on the system as a whole.

Architecting is the process of conceiving, defining, documenting, maintaining, improving, and certifying an architecture throughout the lifecycle of a system. It is both a science and an art.

A **model** is an abstract formal specification of the structure and/or function of a system.

A **system** is a set of interacting or interrelated elements (people, hardware and software, facilities, equipment, material, and processes [automated as well as manual procedures]) that form a unified whole and whose behavior is intended to satisfy customer and/or operational needs.

NOTE – Every system has an architecture and includes a set of entities, even if the architecture is not clearly and accurately described.

An **architecture description** is a work product used to express an architecture. It may contain stakeholders, concerns, viewpoint specifications, viewpoints, views, correspondences, representations, and other elements.

A **stakeholder** is an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.

Concerns are those interests that pertain to the system's development, its operation, or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability.

Perspective in systems architecture is the choice of a context or a reference (or the result of this choice) from which to describe, categorize, explain, or codify system design, typically for comparing with another.

NOTE – To choose a perspective is to choose a value system related to a set of stakeholder concerns. An enterprise perspective relates to an organizational value system. A functional perspective relates to a capability value system.

A **viewpoint** is set of conventions, achieved using a selected set of architectural concepts and structuring rules, for the creation, interpretation, and use of an architecture view to frame one or more particular concerns within a space system.

A **viewpoint specification** defines a pattern or template from which to construct individual views, and it establishes the rules, techniques, and methods employed in constructing a view.

A **view** is a representation of a system from the perspective of a set of concerns. Views are themselves modular and well formed, and each view is intended to correspond to exactly one viewpoint. A view may include representations or correspondences to elements defined in other viewpoints.

Correspondence is an identified relationship from an element in one viewpoint to a related element in another viewpoint. It may be used to express a wide range of relationships, such as equivalence, transformation, composition, refinement, consistency, or constraint.

Aspects capture a set of characteristics or features of the entity of interest in its environment to address concerns within an architecture description.

A **representation** is some way of organizing, manipulating, presenting, and storing information; a visual or tangible rendering of something.

A **requirement** is a formal statement of

- a) an attribute to be possessed by the element or a function to be performed by the element;
- b) the necessary performance for the attribute or function;
- c) the measuring process to be used in verifying that the necessary performance has been met.

A **specification** is a set of requirements or other descriptive information for a system or classifier.

A **policy** is the set of guidelines and constraints on the behaviors and states exhibited by the objects in the system.

A **standard** is a formal specification that defines and governs functions and protocols at interfaces of a data system. It can describe capabilities in detail and establishes the requirements to be met by interfacing subsystems to achieve compatibility.

3.3 GRAPHICAL REPRESENTATIONS

The icons shown in figure 3-1 are used throughout this document. Some minor variants of these icons are introduced as needed in the body of the text.

The intent behind these representation choices is to provide a distinct set of icons associated with the different kinds of objects used in each viewpoint. This is to avoid the frequently seen phenomenon of everything (system, team, software element) being represented by the same rectangular box, and every interaction being represented by the same solid line. These same kinds of distinctions of object and connection types can be transferred into an MBSE tool as well, with no loss of meaning, by defining specific viewpoints and views and creating a profile with a suitable set of object stereotypes associated with each of these different kinds of objects.

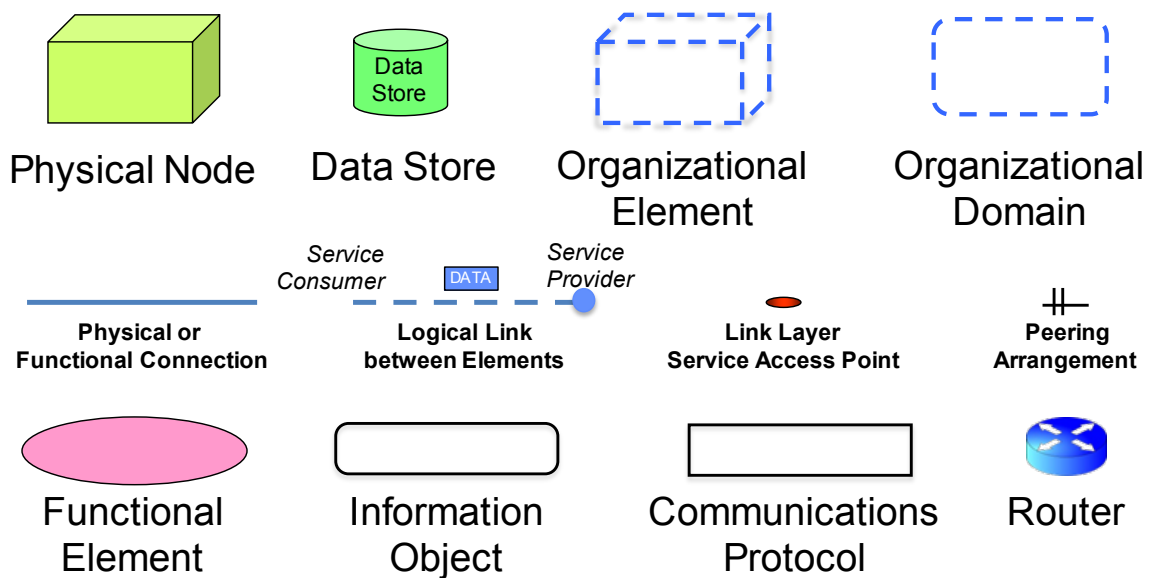


Figure 3-1: Icons Used in This Document

This CCSDS reference architecture recommends this set of icons be adopted and that they be used consistently within any given architecture description project. Other representations can be adopted, but the recommendation is that this be done with care and consistency. It is also acceptable to incorporate additional graphical elements or symbols, to enhance expressiveness, as in figure 4-4 or figure 8-5.

3.4 CHARACTERISTICS OF OBJECTS

As introduced in section 2, each of the viewpoints of RASDSv2 are described by defining the key objects and their interactions. The set of typical interfaces and attributes of objects are shown in figure 3-2. A specific set of representations for the objects and interfaces relevant to each viewpoint are provided in each viewpoint section this document.

Any given object may expose one or more service interfaces and provide one or more core functions. Through its external interfaces, it may call upon other objects to provide services to it. The management interfaces may be explicit (for instance, a service management call to a Protocol Entity) or they may be implicit and be represented by internal tables or configuration items. The only objects used in RASDSv2 that do not exhibit all these interfaces are Information Objects.

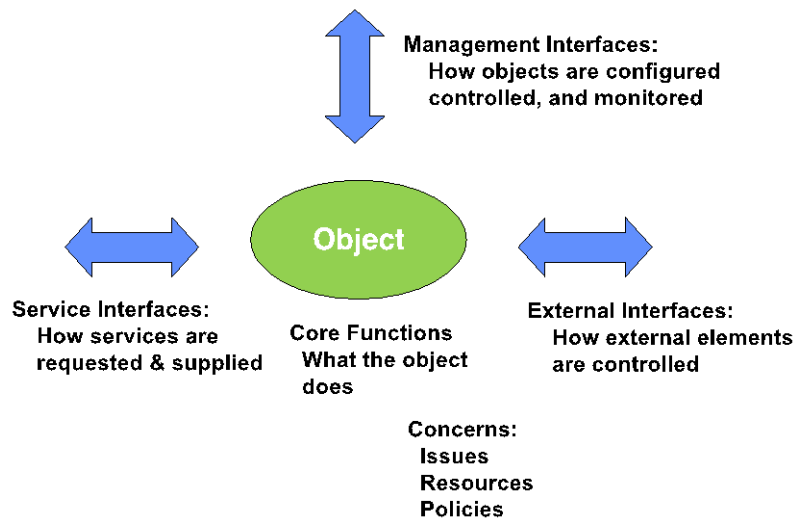


Figure 3-2: Unified Representation of Objects

The types of arrows used to indicate different kinds of interfaces in this, and other similar object overview diagrams, has no special significance in RASDSv2, but double headed arrows are typically used to represent input/output interfaces and single headed arrows are used to indicate directionality of the interfaces. Other kinds of representations (see figure 3-3) may use a specific set of arrow types to convey specific meanings.

Each viewpoint provides an ontology diagram describing the primary and secondary objects defined in each viewpoint and the relationships among them. These diagrams are themselves a form of Information View, and they use the kinds of arrows shown in figure 3-3 to represent these relationships. (See the Information Viewpoint, section 10, for more discussion on various forms of information modeling.)

Read as sentence with subject = start of arrow, verb = label, and object = end of arrow

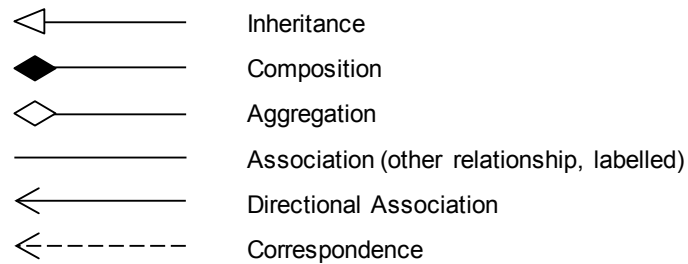


Figure 3-3: Relationship Types among Objects (derived from UML)⁵

Correspondence is an identified relationship from an element in one viewpoint to a related element in another viewpoint. It may be used to express a wide range of relationships, such as equivalence, transformation, composition, refinement, consistency, or constraint.

Figure 3-4 provides an example of the kind of ontology diagram that is provided for each viewpoint. It displays composition, directional associations, and correspondence (dashed line) relationships.

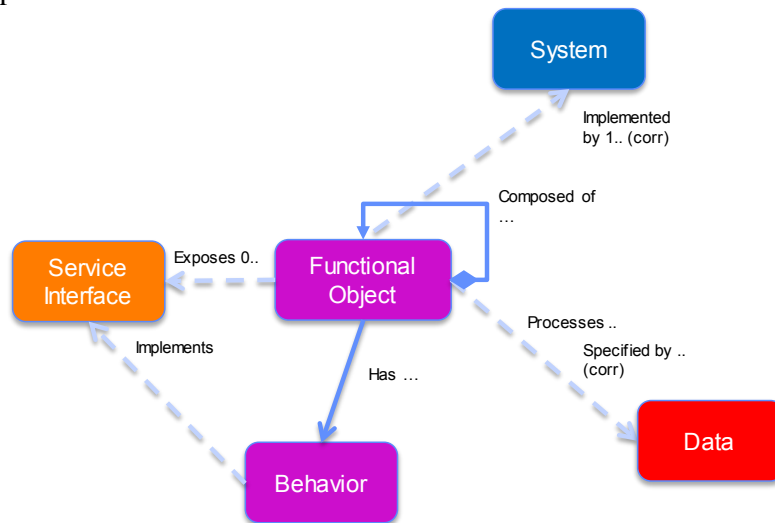


Figure 3-4: Example of a RASDSv2 Viewpoint Ontology (Functional)

All these primary object definition and ontology diagrams use the color keys for viewpoint objects that are defined in figure 2-4.

⁵ Relationships are inherited from UML, with Correspondence, derived from ISO 42010, added as a named extension to Association.

4 ENTERPRISE VIEWPOINT

4.1 OVERVIEW

The Enterprise Viewpoint⁶ addresses the complex organizational relationships and roles involving various resources (spacecraft, instruments, ground systems) and personnel (scientists, staff, and contractors) that may be distributed among multiple organizations (space agencies, science institutes, companies, etc.). The Enterprise Viewpoint also addresses other organizational aspects, such as policies, requirements, governance, assets, roles, and capabilities.

4.2 CONCERNS

Concerns addressed by the Enterprise Viewpoint are:

- the organization responsible for the system;
- the capabilities provided by the system;
- the purpose, scope, and policies for the system;
- the objectives, concepts of operations, and scenarios for the system;
- the requirements and constraints on the system;
- roles played by the system elements.

4.3 CONCEPTS FOR ENTERPRISE VIEWPOINT

The **Enterprise Viewpoint** of a space system focuses on the community, purpose, scope, roles, and policies for that system. This viewpoint includes organizations as well as other Enterprise Objects that have assigned roles, responsibilities, and interactions.

In the Enterprise Viewpoint, a space system is depicted as a set of Enterprise Objects and their relationships, interactions, and the roles that they perform. Enterprise Objects that have significant resources may appear in an Enterprise View as **Facilities**.

An **Enterprise Object** represents an entity that is governed by a single authority that has its own objectives and policies for operating the object.

⁶ The Enterprise Viewpoint is based on the enterprise viewpoint of RM-ODP and Enterprise Architecture methods such as TOGAF. Some modifications have been made to better describe the concerns of space systems and commonly adopted enterprise object names, such as Space Enterprise, mission, program, and project are adopted.

An Enterprise Object may be a component of another larger Enterprise Object, which may in turn be a component of a third, even larger, Enterprise Object. Enterprise Objects may participate wholly or in part in other Enterprise Objects.

A **Facility** is a physical infrastructure element that supports the use of services and other resources.

A **Resource** is anything available to a system that can support the achievement of objectives; such as hardware, software, programs, information, data, and other devices that are in use within or connected to a given system. In this context a **Resource** is an Enterprise Object that has some role, offers services, and performs some action within a system. A resource may serve more than one activity.⁷

An organizational Enterprise Object may **own** a facility or resource Enterprise Object.

Ownership means having administrative and fiscal responsibility for the owned element and the right to exclusively control and use that which is owned for one's own purposes. It is the state or fact of having exclusive possession or control of some object, facility, intellectual property, or some other kind of property.

Not every organizational object owns facilities or resources. Some resources are owned by one organization and used by others. The term **cross support** is used to describe an agreement between two or more organizations to exploit the technical capability of interoperability for mutual advantage, such as one organization offering support services to another in order to enhance or enable some aspect of a space mission.

4.4 CHARACTERISTICS OF ENTERPRISE OBJECTS

4.4.1 GENERAL

The characteristics of Enterprise Objects are shown in figure 4-1. Enterprise Objects are characterized by their roles, objectives and resources, and their interactions involve Requirements, Agreements, contracts, and constraints such as policies and rules. They exchange information such as requirements, memoranda of understanding, service/support agreements, interface control documents, and so on.

Interfaces among Enterprise Objects are often created because of shared science or exploration goals and may involve cross-support agreements, interoperability requirements, and agreements on data sharing and access. Various standards for information exchange or documenting procedures are often employed as the means for enabling these interfaces to work.⁸

⁷ System management, lifecycle views on systems, scenario specifications, and other aspects that are relevant to the Enterprise Viewpoint may also be addressed in this viewpoint.

⁸ See Information and Operations Viewpoints (sections 10 and 12, respectively).

The Enterprise Viewpoint may also be used to represent Scenarios and Operations Concepts. This is the primary system viewpoint where personnel, operations issues, policies, and other organizational concerns are expressed. Roles of Enterprise Objects may include terms like owner, operator, science user, service provider, contractor, developer, tester, manager, and data acquisition, data relay, orbiter, lander, or other descriptive names.

In collaborative (or commercial) joint enterprises attention will need to be paid to policies and agreements, and formal governance arrangements may need to be established and sustained.

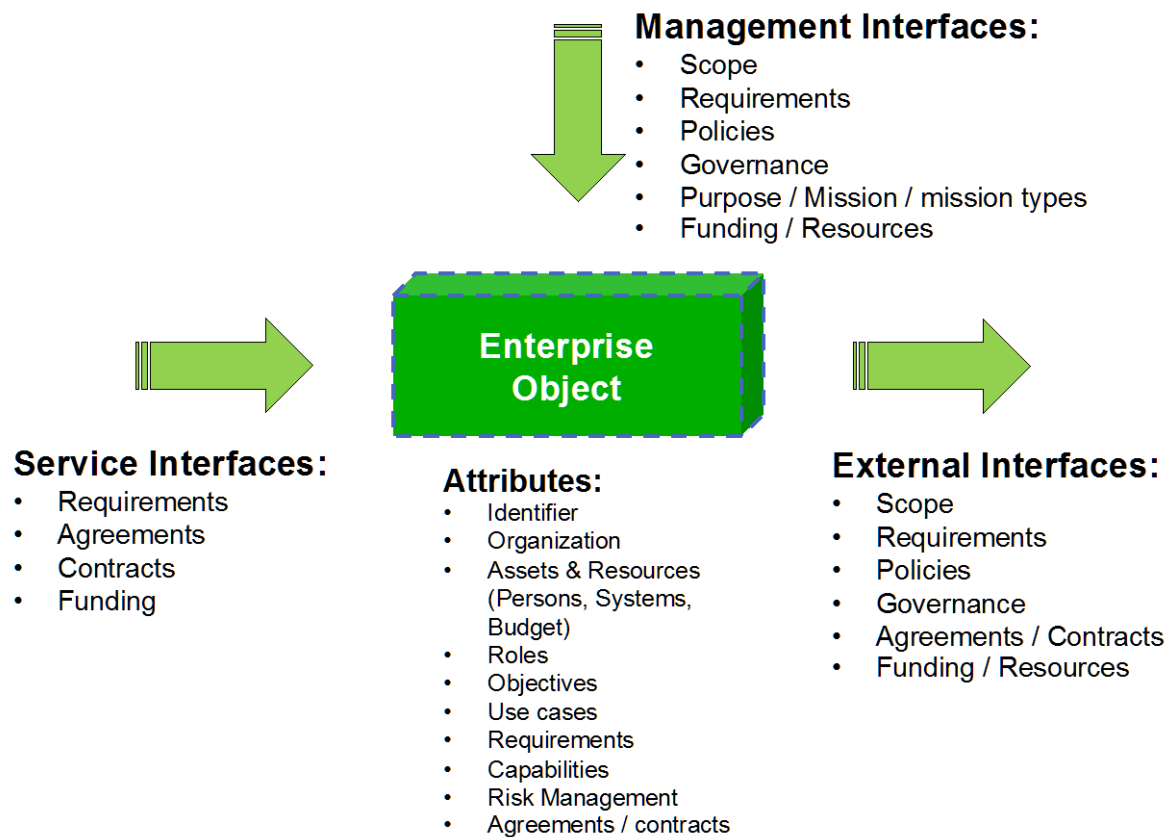


Figure 4-1: Overview of Enterprise Objects

4.4.2 KEY OBJECTS AND RELATIONSHIPS

4.4.2.1 General

The following elements may appear in the Enterprise Viewpoint:

- Enterprise Objects:
 - types: Organizations, both formal and informal (missions, projects, communities, with their roles and responsibilities), and Assets (resources, people, and facilities or other elements having enterprise operational or service roles);

- attributes: name, role, objectives, point of contact, location, members, assets, resources, provided services, types, interfaces and data, interaction modes, requirements, constraints;
- Domains (boundaries of responsibility or ownership);
- relationships (ownership, membership, participation, roles, contractual);
- information (defined instances of documents, agreements, contracts, policies, requirements, objectives, goals, scenarios, membership lists, interface specifications, where formal specifications of the data to be exchanged are found in the Information Viewpoint).

4.4.2.2 Conceptual Object Model

Figure 4-2 provides a conceptual model of the key Enterprise objects and some of their relationships to the most closely related objects from other viewpoints, such as systems and resources from the Connectivity Viewpoint and Operations Viewpoint objects.

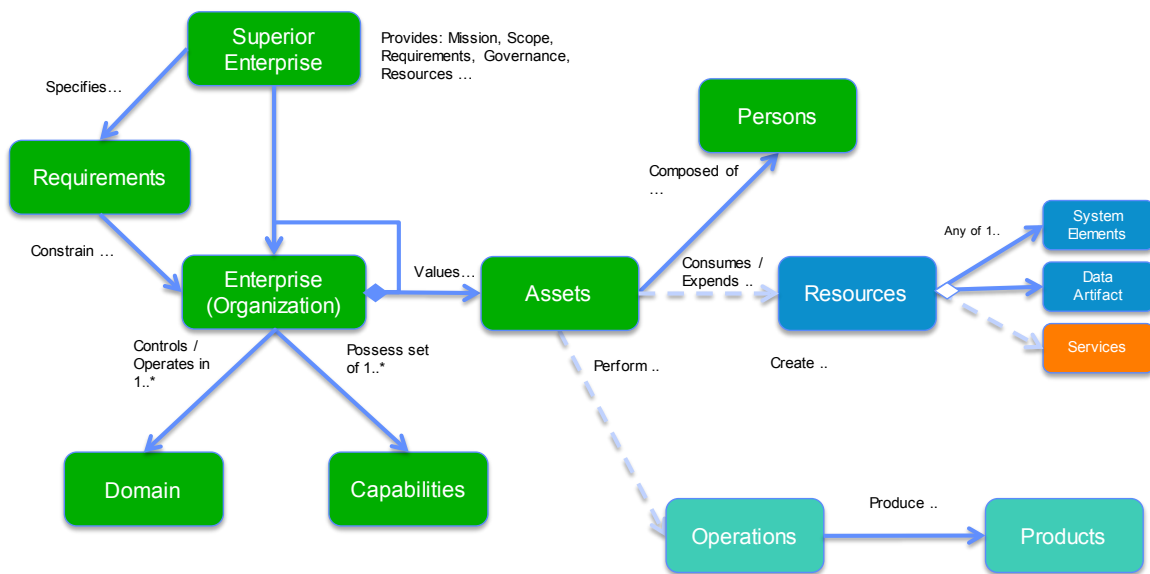
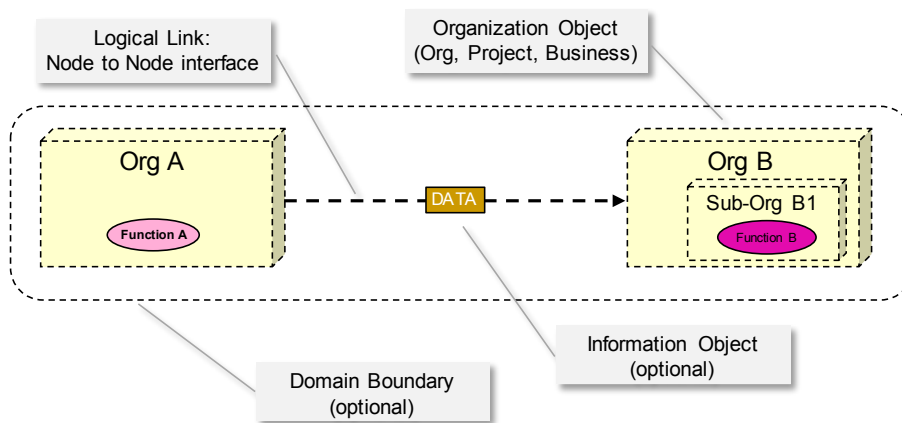


Figure 4-2: Ontology of Enterprise Objects

NOTE – Consistent with figure 2-4, objects shown in green are defined within the Enterprise Viewpoint. Other objects, shown in different colors, are referenced by correspondence from other viewpoints.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS



Specific and Generic Object Types and Containment:






-  Denotes a specific Organization (org, project, business), may have embedded components
-  Denotes a function an organization carries out (optional, see Functional Viewpoint)
-  Denotes a Logical Link (information flow of some sort) between two Organizations
-  Denotes an information object (optional, allows data to be characterized, see Information Viewpoint)
-  Denotes an organization / domain boundary (ownership / responsibility)

Figure 4-3: Representation of Enterprise Objects

4.4.2.3 Object Representation

Figure 4-3 shows the recommended, document-style, representation of Enterprise Objects using the adopted drawing method. As with all the viewpoint representations, the intent is to adopt a drawing style that is both expressive and sufficiently unique to the viewpoint.

4.5 TERMS FOR THE ENTERPRISE VIEWPOINT

4.5.1 ATTRIBUTES FOR ENTERPRISE VIEWPOINT

A **Requirement** specifies a condition or capability that must be met or possessed by a system or system element to satisfy an agreement, standard, specification, or other formally imposed document.

A **Use Case** is a list of actions or event steps typically defining the interactions between an actor and a system to achieve a goal. A Use Case may describe a situation where a system may be used or a potential scenario in which a system receives an external request and responds to it.

A **Capability** is the ability to achieve a desired outcome under specified conditions using a combination of activities and resources to satisfy a stakeholder need.

A (Capability) **Requirement** is a type of requirement describing the capability that the organization or system must provide.

Risk Management is the program and supporting processes to manage information security risk to organizational operations (including mission, functions, image, reputation), organizational assets, individuals, other organizations, and the nation, and includes: (i) establishing the context for risk-related activities, (ii) assessing risk, (iii) responding to risk once determined, and (iv) monitoring risk over time.

A **contract** is an agreement that specifies certain legally enforceable rights and obligations pertaining to two or more parties. A contract typically involves the transfer of goods, services, money, or a promise to transfer any of those at a future date.

4.5.2 UNIQUE TERMS FOR ENTERPRISE VIEWPOINT

An **Organization** is a formal group of people with one or more shared goals.

A **Space Enterprise** (e.g., NASA) is a top-level, autonomous entity that is dedicated to the exploration and/or exploitation of space. It has its own objectives, resources, and policies, and it is not a component of any other Space Enterprise.

Under a federated approach (see below) two or more space enterprises (e.g., NASA and ESA) may support common Enterprise objectives.

A **Community** (e.g., Earth Science) can exist within one Space Enterprise or across multiple Space Enterprises. It is distinguished by being bound by common objectives and relationships and offers a set of resources that can be shared within the Community and with other Communities.

A **Domain** (e.g., NASA Code Y) is a type of Community that is under single organizational, administrative, or technical control. A domain may have resources, policies, access control, and possibly constraints on quality of service.

A Domain may be subdivided into Subdomains. Multiple independent Domains may be organized into a Federation.

A **Federation** (e.g., UN Committee on Earth Observation Satellites [CEOS] or CCSDS) is a Community consisting of multiple Domains that come together to share resources while each domain retains its authority over its own resources. Federations are governed by negotiated agreements.

A Federation may include only some members of a Domain or Subdomain (e.g., a particular Earth Observing project). Members of a Federation agree on rules for sharing resources and for joining and/or leaving the federation.

A **scenario** is a specific sequence of activities that describes system behaviors. A scenario may be used to describe a set of interactions of system elements. Scenarios may be used to derive **use cases**.

An **operations concept** is a verbal or graphic statement, in broad outline, of assumptions or intent regarding the operation of the system. The concept of operations frequently is embodied in observing plans and operations plans. The concept is designed to give an overall picture of the operation of the system.

An **Asset** is a person, data, or other resource that is valued by an organization.

A **Role** is a set of assignments or job functions which may be assigned to a Person or other system entity.

4.6 EXAMPLE ENTERPRISE OBJECT TYPES

The following are examples of typical types of Enterprise Objects, each representing a class of organization. These are listed in table 4-1.

These example objects, in one form or another, with whatever names are used for them, are involved in many space systems and some of their containment relationships. How any specific Space Enterprise is decomposed into component Enterprise Objects, and how they are named, depends on the organization.

Table 4-1: Example Enterprise Objects

Enterprise Objects	Description
Mission	An Enterprise Object that is responsible for designing, building, and/or operating one or more spacecraft.
Project	An Enterprise Object that is responsible for designing, building, and/or operating one or more space system components.
Program	An Enterprise Object that is responsible for one or more Missions or Projects.
Standards Organization	An Enterprise Object that defines relevant information system, communication protocol, data exchange, or other standards or specifications.

4.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH ENTERPRISE VIEWPOINT

Systems in a RASDSv2 model will typically be represented as a set of elements, for example, people, engineered objects (hardware and software), facilities, equipment, material, and

processes (automated as well as manual procedures) that are related and whose behavior satisfies customer/operational needs. As such, a System is an abstract object that may be described in RASDSv2 by a set of Enterprise, Functional, or Connectivity Views. Only the organizations, facilities, and other resources that are part of a system are directly addressed in an Enterprise View.

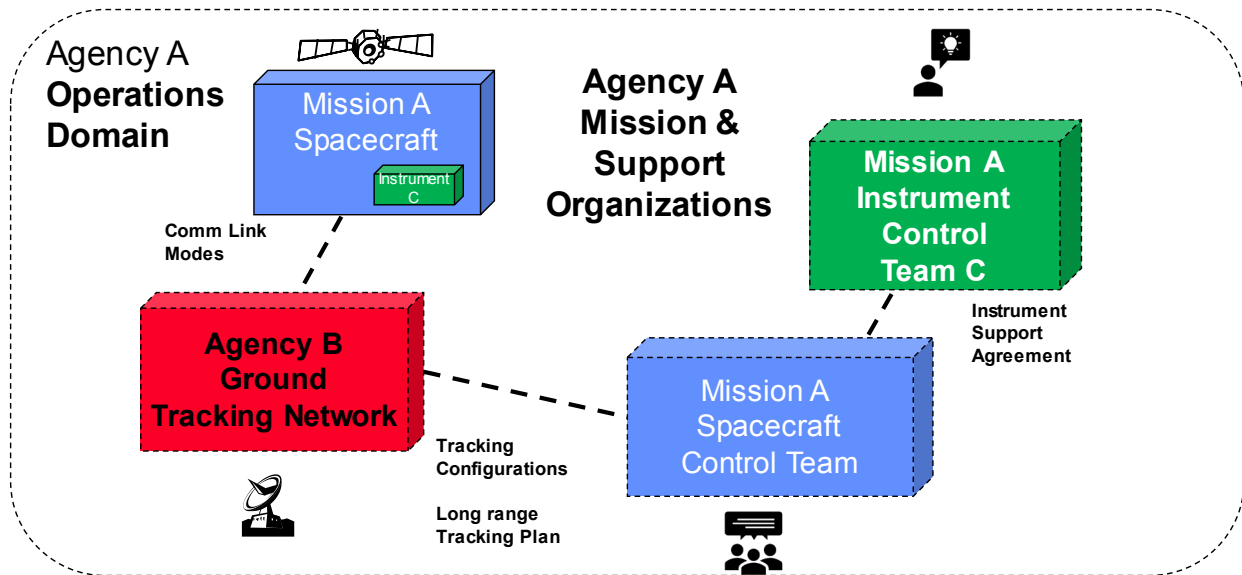


Figure 4-4: Simple Example of a Single Mission Enterprise View

A simple example of an Enterprise View for a single mission is shown in figure 4-4, in which three Enterprise Objects (Mission A Spacecraft and Spacecraft Control Team, and a related Instrument Control Team) are shown as dotted boxes in relationship to an Agency B Ground Tracking Network (GTN). For this purpose the GTN is being treated as inside the Agency A domain, when in reality it may belong to an entirely different Agency. This diagram shows a very simplified view of a mission from the Enterprise Viewpoint, but more complicated relationships may also be modeled as needed.

A simplified abstract view of the Enterprise Objects involved in the operation of the Mars Exploration Program Federation are shown in figure 4-5 together with the interfaces between them. The Mars Relay Orbiter (MRO) and Mars Science Lander (MSL) are missions of NASA, and Mars EXpress (MEX) and ExoMars are missions of ESA. A separate S/C contractor is involved in operating the MRO project and the Deep Space Network (DSN), which is an entirely separate organization from any of the missions, providing deep space communications services. Various formal agreements are in place to define the required bounds of cross-support.

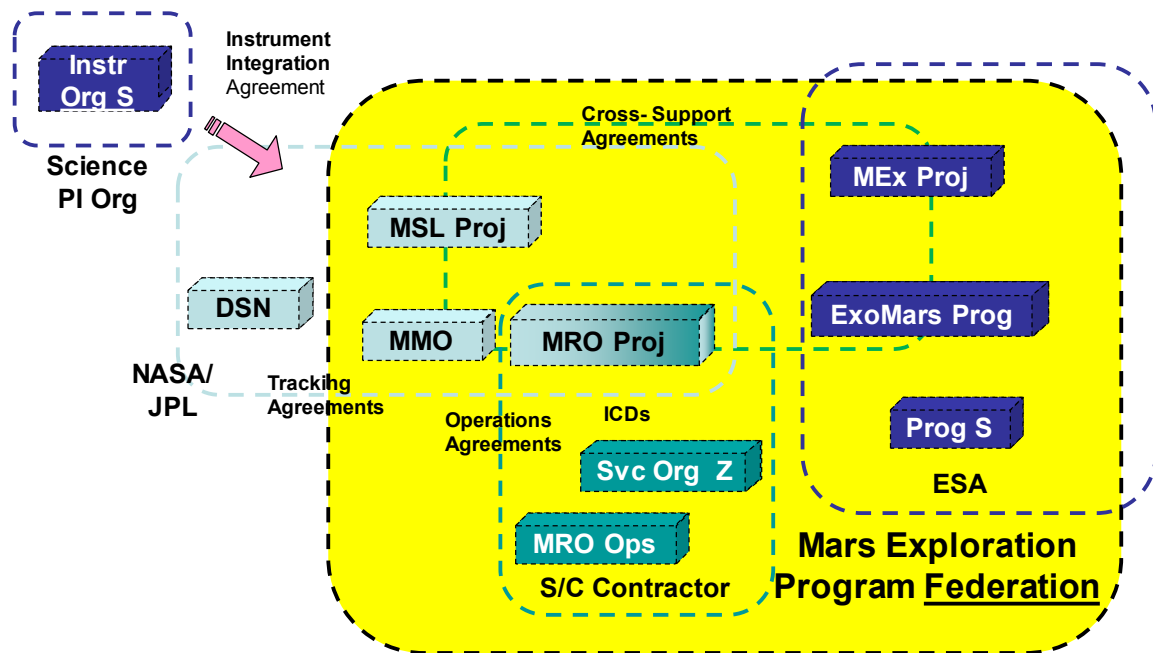


Figure 4-5: Example of an Enterprise View (Mars Exploration Federation)

Organizational elements are shown as dashed 3-D boxes and the boundaries of the domains are shown as dashed rounded boxes.⁹

Enterprise Objects¹⁰ involved in the operations of Mission Z are shown in figure 4-6 together with some of the interfaces between them. Mission Z is a joint mission between Agencies ABC and QRS, and therefore some Enterprise Objects belong to Agency ABC and some to Agency QRS. Agency WXY provides the Launch Vehicle and support infrastructure.

⁹ The Enterprise Viewpoint does not explicitly define use of an overview diagram such as the DoDAF OV-1 or AV-1. However, it is often useful to provide such an overview of an Enterprise to provide easily accessible context for understanding the major objects and roles in the system.

¹⁰ Any graphical icons (spacecraft, antenna, etc.) used in this and other diagrams have no special significance in RASDSv2. Such icons may be employed where they provide additional information as to what is being represented.

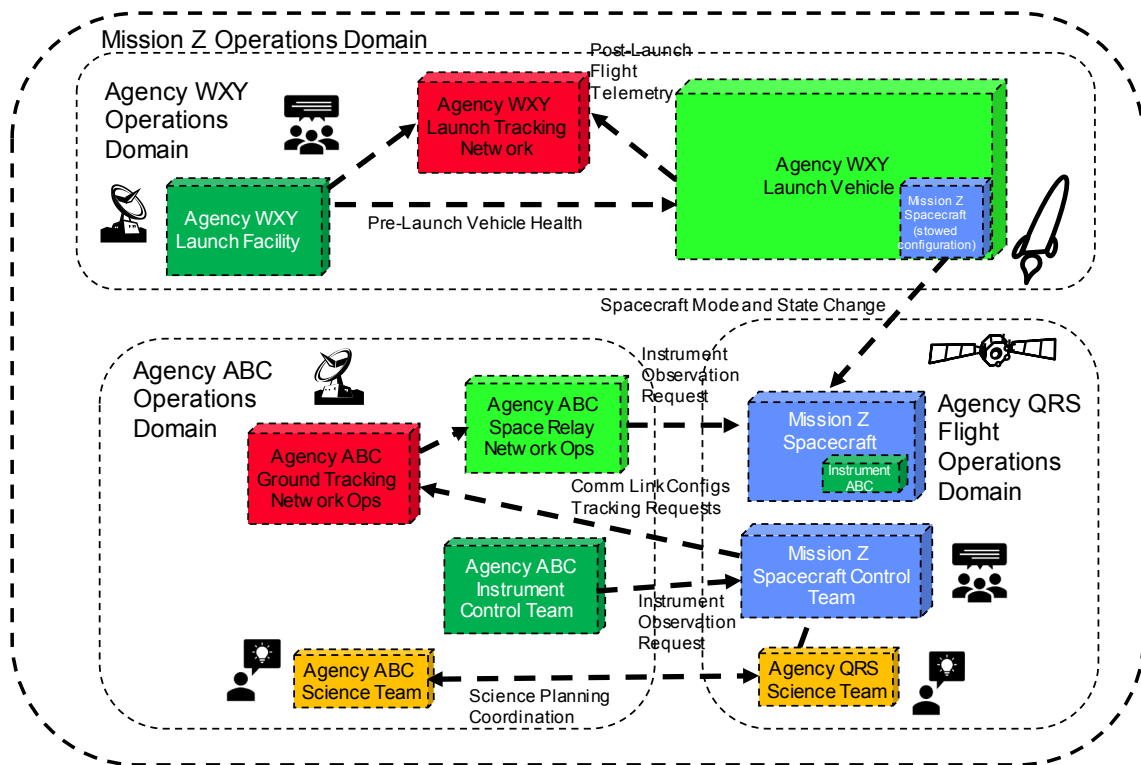


Figure 4-6: Example of Multi-Agency Enterprise Launch View (Mission Z)

Two primary kinds of elements are shown in this Enterprise Viewpoint: organizations and their resources (teams or facilities). Many of these multi-agency collaborating missions are based upon quid-pro-quo arrangements, involving some sort of agreement or contract, and they require a relationship of trust and interdependence between organizations.

4.8 ENTERPRISE ARCHITECTURE AND SYSTEMS ARCHITECTURE

Enterprise Architecture is a somewhat separate discipline from Systems Architecture and there are various methods and tools available to support development of Enterprise Architectures. One of the most prevalent methods at this point is TOGAF (reference [12]), and the Unified Architecture Framework (UAF) (reference [32]) is also now in use. Particularly for planning and managing the growth of large enterprises, these methods and tools can provide substantial benefits. Figure 4-7, redrawn from TOGAF in the RASDSv2 Information Model style, represents the kinds of topics that are addressed by the TOGAF methodology.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

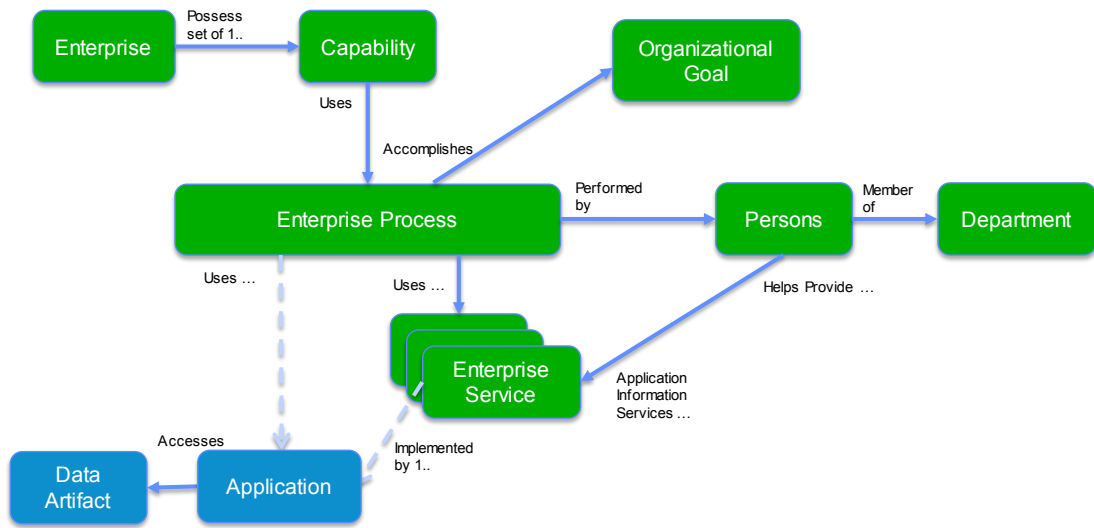


Figure 4-7: Enterprise Architecture Ontology (Adapted from TOGAF)

An Enterprise Architecture may describe capabilities and processes, and identify services, without specifying the technical architecture of the systems and the applications that provide those services. RASDSv2 provides the means to define the technical architecture as well, as shown in figure 4-8.

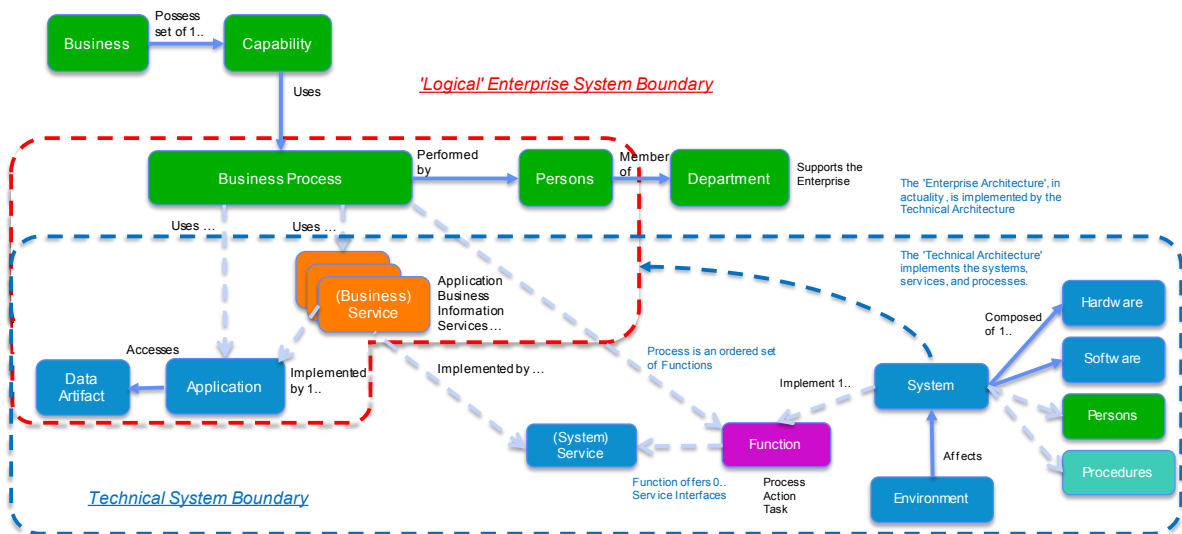


Figure 4-8: Enterprise and Technical Architecture Ontology Relationships

There are defined relationships between the Enterprise architecture and the Technical architecture, as described in the other viewpoints.

4.9 SECURITY TOPICS IN THE ENTERPRISE VIEWPOINT

Every viewpoint has its own set of specific security topics. In the Enterprise Viewpoint the security topics that may be addressed include governance, organizational roles, policies, rules, identities, trust relationships, domain boundaries (e.g., organizational or operational vs. science) and cross-support security agreements. The intent of much of this is to provide security assurance, a measure of the degree of confidence one has that the security controls operate correctly and protect the system as intended. The implementation mechanisms to enable assessment of assurance and to enforce these rules and agreements are detailed in other views. Security responsibilities, analysis of threats, counter-measures, and issues are addressed in detail in a separate Security Architecture document (reference [4]) and in other related security documents within CCSDS, US National Institute of Standards and Technology (NIST), and ISO.

Examples of Specific Enterprise Security Topics:

- security assurance;
- risk management:
 - policies,
 - plan (authorization and accreditation);
- governance:
 - Authorization To Operate (ATO);
- requirements;
- rules;
- interface agreements;
- budgets;
- personnel;
- roles;
- resources;
- assets.

5 FUNCTIONAL VIEWPOINT

5.1 OVERVIEW

The Functional Viewpoint¹¹ separates the analysis of abstract functional elements' logical interactions and abstracted data exchanges from the engineering concerns of how functions are implemented, where they are allocated, how they transfer information, which protocols are used, and what language is used to implement them. Keeping the analysis of the functional behavior required of a system separate from the details of how (and where) to implement it provides a degree of freedom to separate functional design from the technical details that must be explored in doing implementation design trades.

5.2 CONCERNS

The concerns addressed by the Functional Viewpoint are:

- the functional decomposition of the system into objects that interact at interfaces;
- the identification of the Data Objects that are exchanged;
- the abstract behavior of the system, its interactions, and constraints.

5.3 CONCEPTS FOR FUNCTIONAL VIEWPOINT

The **Functional Viewpoint** of a space system focuses on the behavior, structure and interaction of the functions performed by that system. This viewpoint addresses Functional Objects, their behavior, the logical connections between them, the information they exchange, and their logical interfaces and interactions.

The behavior of a function is the set of actions performed by this element to achieve an objective. A **Functional Object** performs actions to achieve an objective of a space system or to support actions of another Functional Object, and this may involve data transformation, generation, or processing in performing those actions.

Functional Views define Functional Objects to control and manage system behavior, such as planning, scheduling, monitoring, and other active control elements that are part of describing the functional behavior of the system. They also describe processing functions and the logical flows of information among these objects.

To describe the full behavior of a complex system, separate depictions of data flows, control flows, and management flows may be shown for a given set of Functional Objects. These flows may use the same or different interfaces on the same Functional Object. Several

¹¹ The Functional Viewpoint corresponds to the computational viewpoint of RM-ODP. The computational viewpoint of RM-ODP describes the structure of application processes in a distributed processing system.

separate views of the same Functional Objects, all of which obey the same rules, may be required to show all of the different aspects of the objects and interactions that compose the Functional Views of a system.

The Information Objects that appear in the Functional Viewpoint are references to the Information Objects that are fully described in the Information Viewpoint. The details of how these Information Objects are defined, described, and controlled are covered in the Information Viewpoint (section 10).

A Functional view shows behavior and other attributes and the logical flow of information among objects. In the engineering of any given system, implemented instances of these Functional Objects may be allocated to one or more nodes as represented in the Connectivity or Structural Viewpoints. The physical means for providing communications connections among implemented functions are treated as protocol stacks in the Connectivity Viewpoint (section 6), as are the physical attributes of the connections and their behavior. The physical means for providing connection or articulation functions will be treated in a Structural Viewpoint. These allocation processes are part of usual Systems Engineering practices, as described in ISO 15288 and elsewhere.

5.4 CHARACTERISTICS OF FUNCTIONAL OBJECTS

5.4.1 GENERAL

The overview of Functional Objects is shown in figure 5-1.

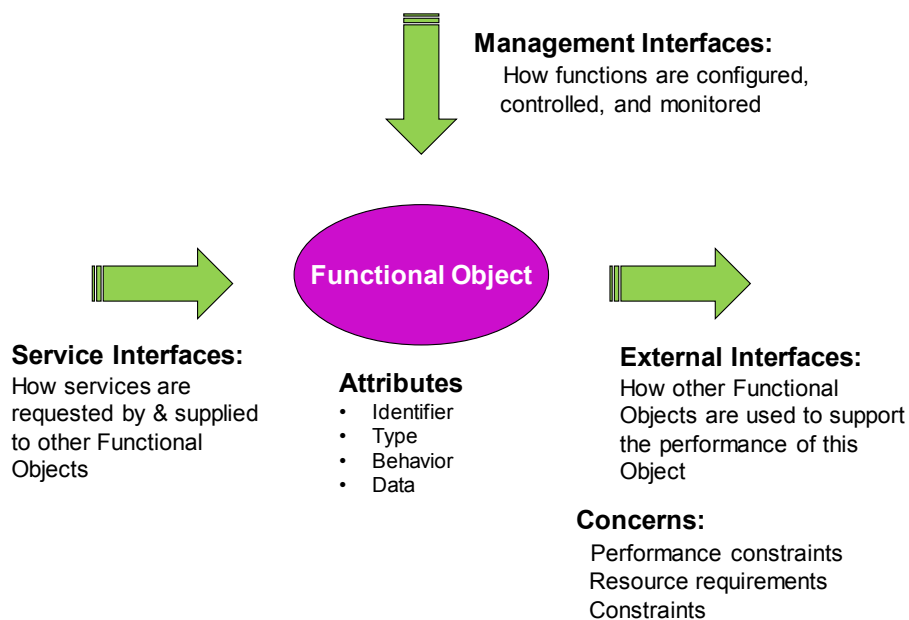


Figure 5-1: Overview of Functional Object

Each Functional Object has three categories of interfaces: service interfaces, external interfaces, and management interfaces. Every Functional Object has one or more interfaces through which the actions of the object are invoked. These interfaces may be shown explicitly or just implied as the locus of the connection between one Functional Object and another.

5.4.2 KEY OBJECTS AND RELATIONSHIPS

5.4.2.1 General

The following elements may appear in the Functional Viewpoint:

- Functional Objects (abstract set of functions, their behaviors, interfaces, and configurations);
 - types: data source, data sink, data transformation, control, planning, monitoring, analysis;
 - attributes: role, name, type, behavior, interface signature, data types handled, interaction modes, constraints, allocated requirements;
- logical links (connections between Functional Objects, connected to associated logical behavior and properties);
- relationships (configuration, precedence, control and data flows, management flows, allocations);
- information (representations of data that are exchanged among Functional Objects, where formal specifications for exchange are found in the Information Viewpoint).

5.4.2.2 Conceptual Object Model

Figure 5-2 shows the ontology of Functional Objects in relationship with other RASDSv2 objects.

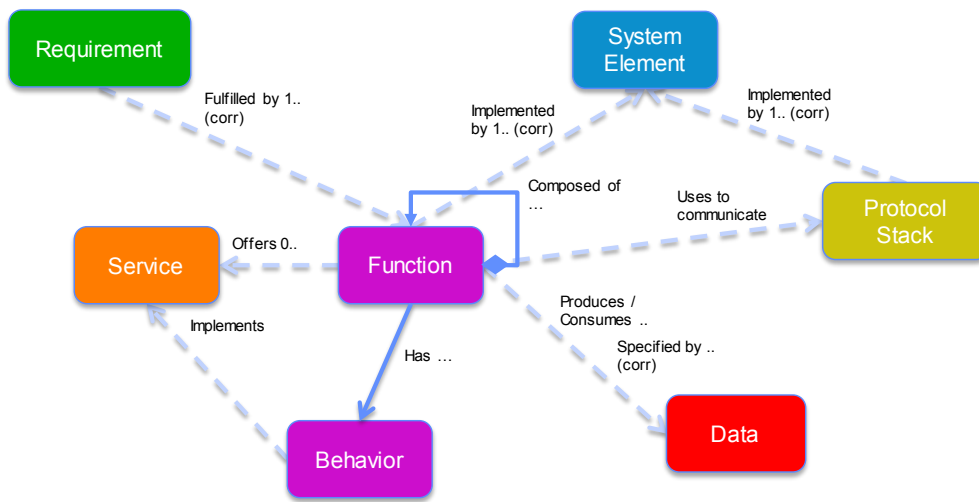


Figure 5-2: Ontology of Functional Objects¹²

Functions have behaviors that fulfill enterprise requirements. These functions are implemented by systems elements that may use protocol stacks to communicate. Some of these functions may offer formal services that may be fully documented in a separate Services view.

Figure 5-3 shows the standard representation defined for Functional Objects.

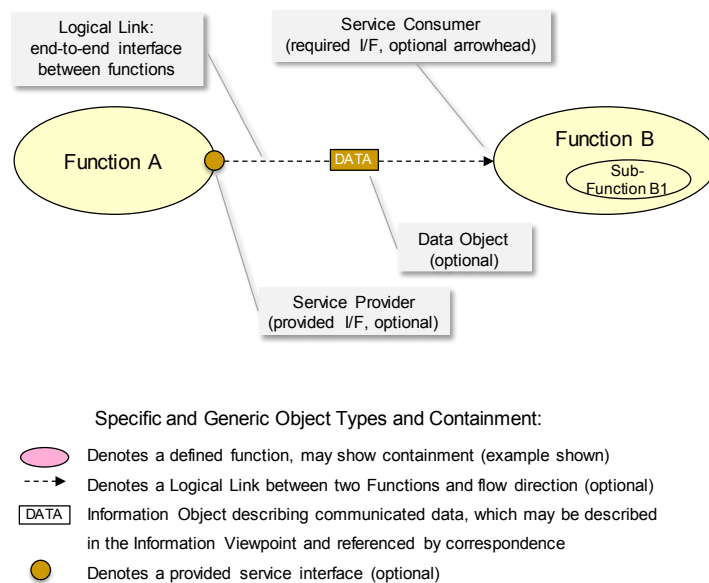


Figure 5-3: Representation of Functional Objects

¹² Objects shown in magenta are defined within the Functional Viewpoint. Other items, shown in different colors, are referenced by correspondence from other viewpoints.

5.5 TERMS FOR FUNCTIONAL VIEWPOINT

5.5.1 ATTRIBUTES FOR FUNCTIONAL VIEWPOINT

The **Identifier** for a Functional Object is a unique name assigned within some context that allows the object to be uniquely identified.

The **type** assigned to a function is the subset of all possible types that have meaning in a given context. The user is free to create classes of types that are meaningful for user needs.

Behavior is the extrinsic aspects of what a function does, such as perform some algorithm or transform data in some way.

The **Data** attribute of a function is the means to define what classes of data it will process.

5.5.2 UNIQUE TERMS FOR FUNCTIONAL VIEWPOINT

Abstraction is a mechanism and practice to reduce and factor out details so that one can focus on a few concepts at a time. In this context it is the separation of the description of system functionality from the details of system implementation.

Information Management Functional Objects are active functional elements that support the location, access, delivery, and management of Information Objects.

5.6 EXAMPLE FUNCTIONAL OBJECT TYPES

Table 5-1 shows a set of typical high-level Functional Objects used in space systems; they may occur on orbit, on the surface of a planet, or both. These are provided only as examples, and any given system may add other functions, decompose them differently, or use other names for the same functions. For instance, Orbit Determination and Trajectory Design could be named separately in one system or aggregated and called Flight Dynamics in another. These examples intentionally show only very high-level functions, which will typically be further decomposed during the design process.

Depending on the system, Functional Objects may be decomposed into subfunctions, each of which is performed by a component Functional Object of the parent Functional Object. How Functional Objects are decomposed into component Functional Objects depends heavily on the system design and local practice, and it is beyond the scope of this reference architecture to define specific decompositions of these Functional Objects.

Table 5-1: Example Functional Objects

Functional Objects	Description
Experiment Control	A function to control an experiment or observation (data acquisition, sample acquisition, etc.).
Data Transport	A function to manage and control the execution of data transport functions supplied by Communications Objects.
Directive Execution	A function to execute a set of directives (goals or a time-ordered set of directions).
Directive Management	A function to manage remotely a set of directives (goals or a time-ordered set of directions).
Directive Generation	A function to generate a set of directives (goals or a time-ordered set of directions) based on a mission plan.
Monitor and Control	A function to monitor the status of other Functional Objects and to request execution of necessary actions when a predefined anomaly or deviation occurs.
Mission Planning	A function to generate a mission plan (time-ordered set of goals or sequence of activities).
Spacecraft Analysis	A function to analyze the status of a spacecraft using data from a data store.
Mission Analysis	A function to analyze the status of instruments and to assess the level of achievement of mission goals, using data from a data store.
Tracking	A function to steer an antenna to maintain communications links with a spacecraft or a ground station.
Radiometric Data Collection	A function to collect radiometric data (e.g., range and Doppler).
Orbit Determination	A function to estimate the state vector of a spacecraft using radiometric data and possibly image or other position-sensitive data taken by the spacecraft.
Trajectory Design	A function to design the trajectory of a spacecraft including plans for orbit change maneuvers.
Space Transport	A function to change a spacecraft orbit or location.

Table 5-2 shows several typical infrastructure Functional Objects. These are also Functional Objects, but they are distinguished because they typically provide supporting services for the more application-oriented Functional Objects shown in table 5-1. Sometimes these infrastructure objects may be shown supported by a separate application stack layer, as described in the CCSDS Application and Support Layer Architecture (ASL) document (reference [27]).

Table 5-2: Typical Infrastructure Objects

Functional Objects	Description
Information Management	A set of functions to store, locate, access, and deliver data (see the Information Viewpoint for more details on these elements).
System Management	A set of functions to monitor, manage, configure, and control other functions in a system, usually via their management interfaces.
Messaging Middleware	A set of functions to provide services for naming, locating, accessing, and interfacing with elements of a distributed system. May also be a Communications Viewpoint set of objects.

5.7 EXAMPLES OF A SPACE SYSTEM DESCRIBED FROM THE FUNCTIONAL VIEWPOINT

A simple example of a Functional View is shown in figure 5-4, where a set of related Functional Objects are represented as ovals, and the functional interactions are represented by dashed lines. These interactions take place at the interfaces of these objects. In the top example only the flows are shown. In the bottom example simple representations of Information Objects, described more fully in section 10, are shown as being exchanged across these logical links, and the Provided Interfaces of these Functional Objects are also identified. Either representation may be used.

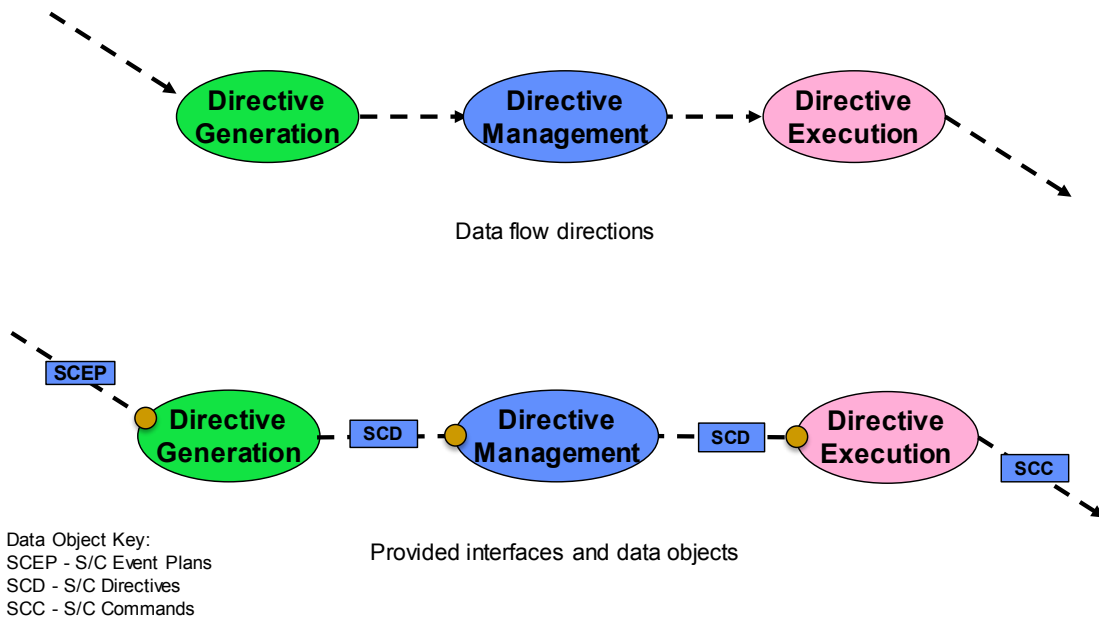


Figure 5-4: Simple Example of a Functional View

A Functional Object may be composed of other Functional Objects. A formal or informal group of Functional Objects that provide some service in a space system, such as a related set of navigation or data processing services, may be modeled as a higher-level element in a Functional View. The decomposition of these elements may also be shown in related views. The Information Objects that are exchanged across the interfaces between Functional Objects are abstractions, the full specification for these Information Objects will be found, by correspondence, in a related Information View.

Figure 5-5 shows a representative set of Functional Objects used at an Application Layer in typical space systems together with the logical interactions that occur among them (shown with dotted lines). The points of connection are at the interfaces, shown as bubbles on the edges of the functions in this view, denoting the Provided Interfaces. On these logical links flow various forms of information, which are shown in this particular view as labelled rectangles. These should be referenced to Information Objects defined in an Information View. (See section 10 for more discussion of the Information Viewpoint where the means to describe these Data Objects are defined.)

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

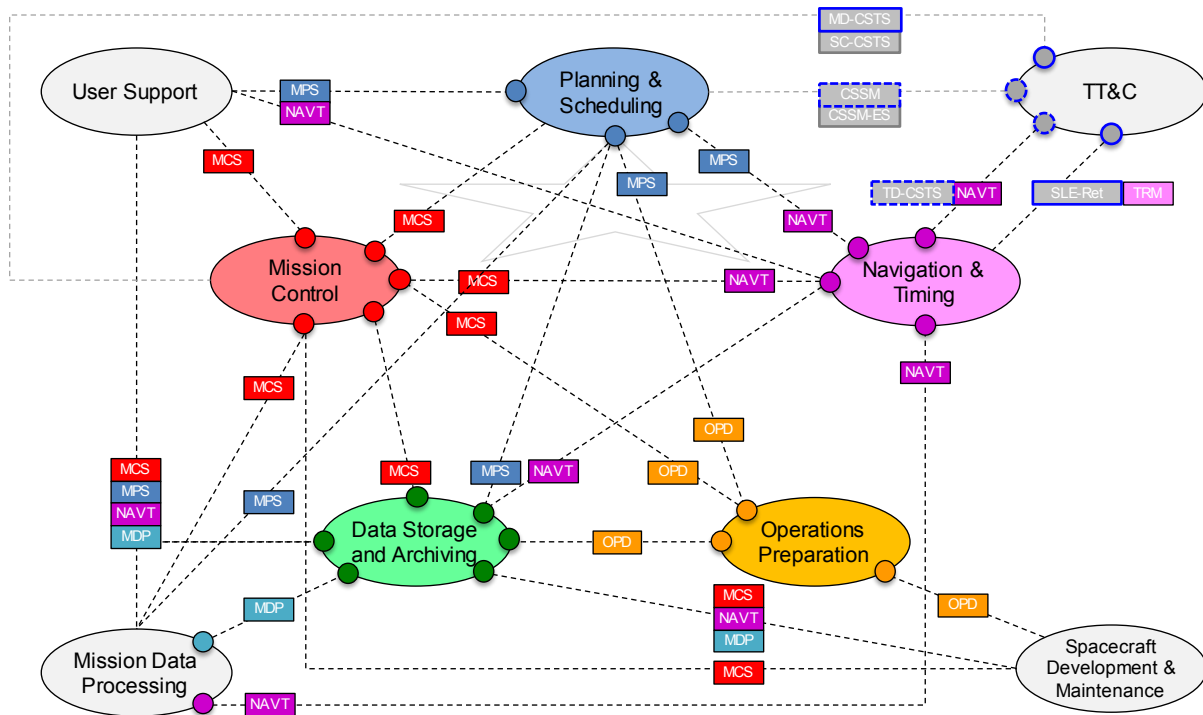


Figure 5-5: Example of Functional View (Functional Objects, Provided Interfaces, and Data)¹³

Other views of this same set of Functional Objects might be shown, including views showing the decomposition of each of these high-level functions into lower-level ones, or a view showing the control flows between directive generation, execution, and data acquisition. All these views may be developed using the same Functional Viewpoint specification.

RASDSv2 does not provide any specific recommendations for the representation of the internal behavior of Functional Objects. Decomposition to lower levels of detail is one way of showing this. Other formal means, such as use of state charts, activity diagrams, or algorithmic specifications may also be employed as needed in a Functional View. For describing interfaces, any suitable annotation may be adopted to represent and label Information Objects or to label interfaces, or the features of the Service Viewpoint may be employed if more formal service interfaces must be defined.

¹³ Figure 5-5 is borrowed from CCSDS 371.0-G-1, figure 4-2.

5.8 EXAMPLE OF SPACE SYSTEM WITH INFORMATION MANAGEMENT INFRASTRUCTURE

A specific set of Information Management Functional Objects may be included in the Functional Viewpoint. These are the elements of an information infrastructure that support the location, access, delivery, and management of Information Objects. These Information Management Functional Objects are Functional Objects, but they are often considered together with Information Objects because of their close relationship to them.

The upper part of figure 5-6 shows a representative set of Functional Objects that might be used to carry out some information management activity. The lower part of figure 5-6 shows the set of Information Management Functional Objects (query service, registry service, repository service, product service) that provide an infrastructure for managing, accessing, locating, processing, and distributing the information exchanged by other Functional Objects.

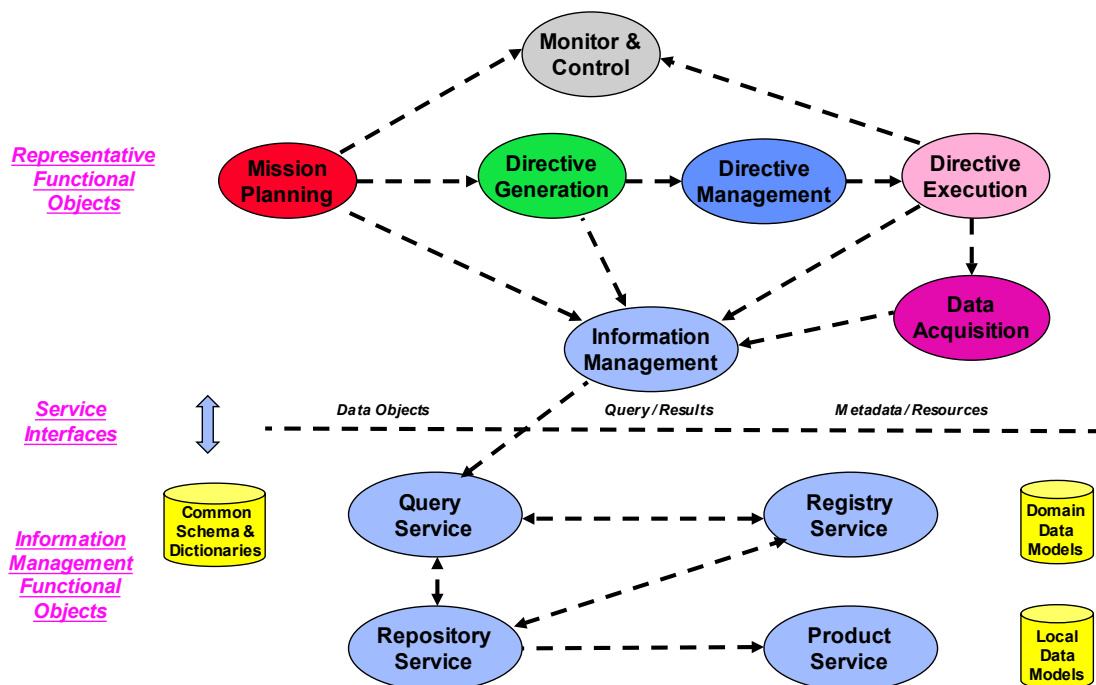


Figure 5-6: Representative Functional Objects and Information Management Infrastructure Elements

In some systems, these infrastructure elements may be instantiated by simple files, tables, or even data stored in memory. In other systems, these will be system-level functional elements, implemented as subsystems and using various commercial elements like data base management systems or distributed system frameworks.

These basic Information Management Functional Objects may be composed into a broad set of information management services to support mission operations functions as well as on-board data management. They may also be combined with other functions that do transaction management or data ingest to produce federated data systems and back-end archival systems.

A more complete description of these Information Management Functional Objects, their functions, and interfaces is separately addressed in the Reference Architecture for Space Information Management (reference [5]).

These information management services may be remote from a system and customized for specific purposes or classes of data. They may manage different kinds of data structures, such tables, lists, or knowledge graphs, and they may use standardized query languages like SQL (reference [35]) or specialized languages like SPARQL (reference [36]). CCSDS itself manages a set of registries in the Space Assigned Numbers Authority (SANA) (reference [37]) that are accessible on the Web and contain specialized data relating to its standards and the uses that are made of them. These are documented in the Registry Management Policy (RMP) (reference [38]) and companion documents. Many CCSDS standards refer to these or create other new registries as needed.

5.9 SECURITY TOPICS IN THE FUNCTIONAL VIEWPOINT

In the Functional Viewpoint, the Functional Objects that are used to implement security policies and approaches are defined. These may include: access control interfaces on functions and specific functional elements such as authentication, source level encryption, and key management subsystems. Some of these may be shown as Functional Objects in their own right (e.g., Public Key Infrastructure [PKI] management function), or just as attributes of other Functional Objects (e.g., access control on a management or control function).

Examples of specific functional security elements:

- threat assessment of all functions;
- verification (authentication);
- confidentiality (encryption);
- risk assessment;
- threat assessment;
- access monitoring/intrusion detection;
- event logging;
- identity management/Identity, Credential, and Access Management (ICAM);
- key management;
- application management;
- network management;
- perimeter management;
- vulnerability scanning and management;
- validation and normalization of input parameters.

6 PHYSICAL VIEWPOINT

6.1 OVERVIEW

The Physical Viewpoint provides the core description of the physical elements that operate in space where connections between elements, physics of the elements, and interactions with forces in the external environment are considered. More specific viewpoints, using more specific terms in their domain of discourse, can be derived from this Physical Viewpoint.

The ‘derivation’ of other viewpoints enables the derived viewpoints implicitly to carry the following aspects:

- a graphical model of composition in which the connectors and physical interfaces that connect Physical Objects appears in the physical viewpoint and in derived viewpoints:
 - the graphical model of composition makes it possible to explain the deployment of separate instances of the same class of objects throughout space,
 - the authors of derived viewpoints can use the connections to explain relationships and dynamics among the components;
- some concerns of security that involve Physical Objects may be inherited by derived viewpoints;
- for architectures that do not need to describe different physical aspects, the core physical viewpoint can provide a place for physical specifications without using new viewpoints;
- for architectures that describe many physical aspects, the derived viewpoints provide means to organize those specifications.

Two examples of derived viewpoints, the Connectivity Viewpoint and the Structural Viewpoint, appear in sections 7 and 8.

For analysis of space systems in general, all the physical aspects of the system, including the propulsion, power, thermal, structural, and so on, aspects associated with them, must be considered and represented in this Physical Viewpoint or in derived viewpoints.

The Physical Viewpoint and derived viewpoints include all aspects of space system design dealing with the composition of physical elements, their physical connections, and the allocation of functionality to these elements. The physical elements include processors, instruments, storage devices, radios, bus structures, and other components as well as hardwired links, buses, and RF and optical links. The Physical Viewpoint and derived viewpoints are where these engineering issues are handled, along with the issues associated with choosing the best strategy for how to implement the selected logical functionality in hardware and software components.

The derivation of physical viewpoints provides a means to balance the distribution of architectural information across an architectural description. An architecture that leaves

most physical description to be resolved by engineers who implement the architecture can package most of its physical description in a Physical Viewpoint. Sometimes an architectural description must specify details in one or more viewpoints derived from the Physical Viewpoint. The Connectivity Viewpoint is an example of this need in architectural descriptions of the communications aspects of space systems.

6.2 CONCERNS

The concerns addressed in the Physical Viewpoint are:

- the physical decomposition of the system into objects that interact at interfaces;
- the concrete behavior of the system, its interactions, and constraints;
- The interactions of the system with the physical world and energetic forces.

6.3 CONCEPTS FOR THE PHYSICAL VIEWPOINT

The Physical Viewpoint of a space system focuses on the Physical Objects that compose the system. This Viewpoint addresses Physical Objects, their behavior, the physical connections between them, the physical quantities they exchange, their physical interfaces, and interactions within the system and with the outside environment and forces.

The behavior of a physical element is the set of roles or actions performed by the element to achieve an objective. A Physical Object performs actions to achieve an objective of a space system or it plays a role to connect other Physical Objects, and this may involve trajectories, locations, orientations, temperatures, or provision or consumption of resources.

Physical Views define Physical Objects to realize system behavior, such as traveling, stabilizing, articulating, conducting energy, and other roles and actions that are part of describing the physical behavior of the system. They also allocate instances of functions among these objects.

For describing the full behavior of a complex system, separate depictions of data flows, energy flows, and articulations may be shown for a given set of Physical Objects. These different kinds of flows may use the same or different interfaces on the same Physical Object. Several separate views of the same Physical Objects, all of which obey the same rules, may be required to show data connectivity, structure and articulations, propulsion, attitude control, navigation, thermal control, radiation, and other different aspects of the objects and interactions that compose the Physical Views of a system.

A Functional view shows behavior and other attributes and the logical flow of information among objects. In the engineering of any given system, implemented instances of these Functional Objects may be allocated to one or more Physical Objects as represented in the Connectivity Viewpoint. The physical means for providing communications connections

among functions distributed in space are treated in the Connectivity Viewpoint (section 7), as are the physical attributes of the connections and their behavior.

6.4 CHARACTERISTICS OF PHYSICAL OBJECTS

6.4.1 GENERAL

The Physical Objects may have any of all the interfaces shown in figure 6-1.

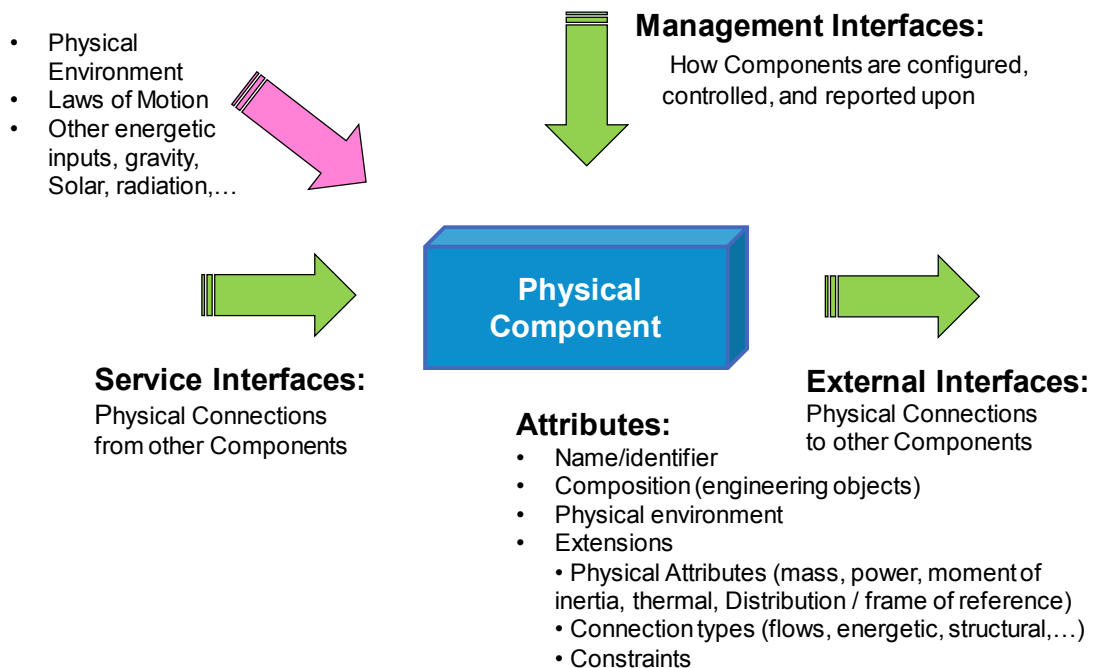


Figure 6-1: Physical Object Overview

The Physical Objects may offer four categories of interfaces: service interfaces, external interfaces, environmental interfaces, and management interfaces. Every Physical Object has one or more interfaces through which it interacts with the physical environment, according to laws of motion and energetic exchanges. External interfaces of Physical Objects provide the means to invoke function instances of other Physical Objects. Service interfaces provide the means to invoke and manage functions implemented by the object.

Good practice identifies generalized sets of Physical Objects as an aid to re-use. Specialized sets are developed only as needed. Current definitions of design patterns are examples of a similar approach now used in software development.

6.4.2 KEY OBJECTS AND RELATIONSHIPS

6.4.2.1 General

The following elements may appear in the Physical Viewpoint:

- Physical Objects that implement functions (abstract set of functions, their behaviors, interfaces, and configurations);
 - types: data source, data sink, data transformation, control, planning, monitoring, analysis;
 - attributes: role, name, type, behavior, interface signature, data types handled, interaction modes, constraints, allocated requirements;
- Physical Objects that provide energetic exchanges:
 - nature of energetic exchange, such as heat, electrical power, electromagnetic radiation, structural continuity, momentum, angular momentum;
- logical links (connections between Physical Objects, connected to associated logical behavior and properties);
- relationships (configuration, precedence, control and data flows, management flows, allocations);
- Information (representations of data that are exchanged among Connectivity Objects, where formal specifications for exchange are found in the Information Viewpoint).

6.4.2.2 Ontology of Physical Objects

Figure 6-2 shows the ontology of Physical Objects in relationship with other RASDSv2 objects.

It also explicitly covers the relationship between components and the environment.

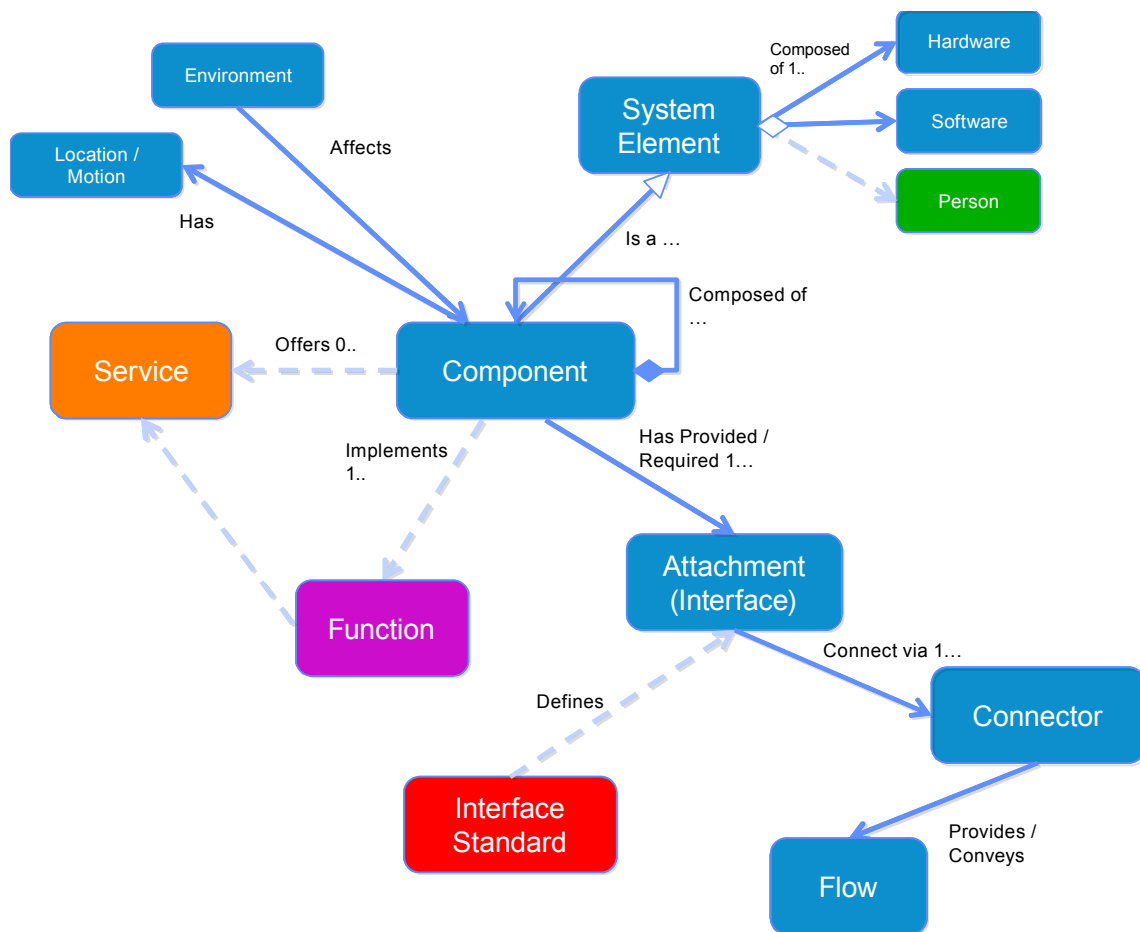
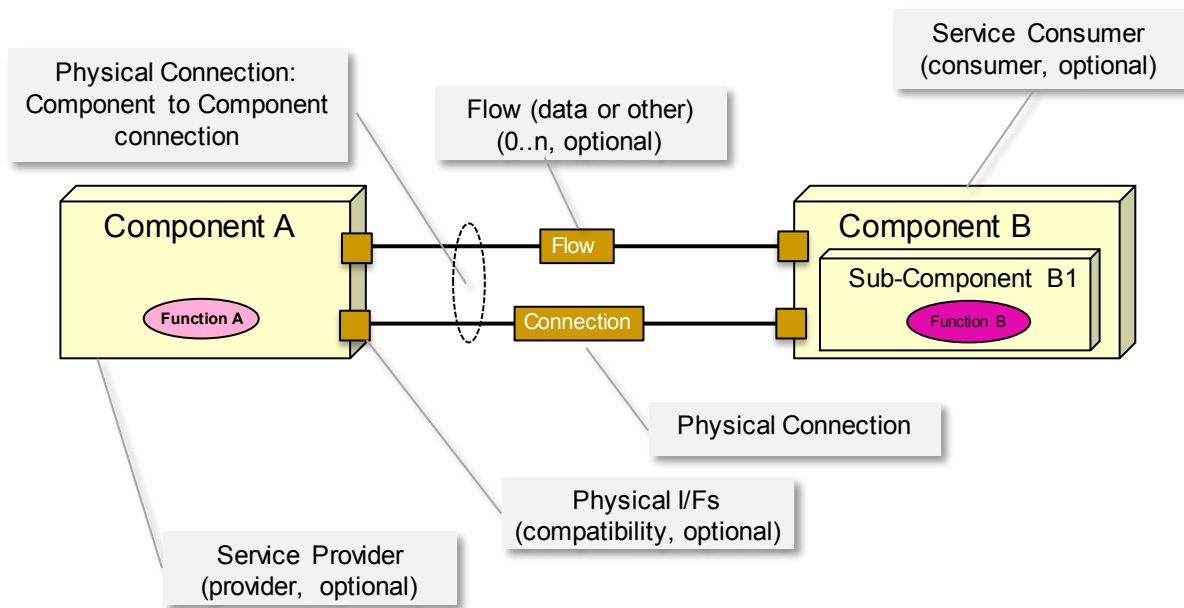


Figure 6-2: Ontology for Physical Viewpoint

Physical components and connectors interface at an attachment point. The characteristics of the attachment may be defined by an interface standard.

6.4.2.3 Representation of Physical Objects

In RASDSv2 diagrams, Physical Objects are represented using the drawing style shown in figure 6-3. The focus here is on physical characterization, connections, and flows, which may involve energy, fluids, or data.



Specific and Generic Object Types and Containment:






-  Denotes a specific Node (physical component), may have embedded components
-  Denotes an implementation of a defined function, may be software or hardware (optional and referenced by correspondence, see Functional Viewpoint)
-  Denotes a Physical Link (Connection of some sort) between two Nodes
-  Denotes a Physical Interface (optional, allows interface type to be characterized)
-  Information Object describing flow or connection, which may be fully defined in the Information Viewpoint and referenced by correspondence

Figure 6-3: Physical Object Representation

NOTE – It is recognized and acknowledged that these simple drawings of physical components and connectors do not capture all the details and nuance that typical mechanical or electrical engineering CAD/CAM drawing tools are capable of, and it is not expected that they would obviate such engineering tools. What these views do permit, however, is to allow the functional, logical, physical, structural, and communications aspects of a system to be related in a coherent way.

6.5 TERMS FOR PHYSICAL VIEWPOINT

6.5.1 ATTRIBUTES FOR PHYSICAL VIEWPOINT

mass properties: Center of mass, inertia matrix.

connections: The types of connections and the sub-components that they connect. Thermal conductivity, wiring harness, etc.

physical environment: Limits on environmental properties, such as pressure, ambient temperature, etc.

configuration: DIP switch settings, jumpers, etc.

constraints: Operational limits, such as mechanical stress, flexibility, angular momentum storage, electrical power storage, etc.

hardware: The mechanical, magnetic, electrical, and electronic devices or components of a system used for producing, collecting, processing, storing, or transporting data.

software or computer programs: The components of information systems that provide operating instructions for specific task-based applications that run on computing hardware.

firmware: Software that is contained in a read-only memory (ROM) device. It is typically treated as software unless there is a reason for showing the hardware component itself.

engineering object: An implementation or realization of some abstract function. It may be implemented as hardware (node) or as software (application or software component).

6.5.2 OTHER TERMS FOR PHYSICAL VIEWPOINT

A **component** is a physical entity operating in a physical environment. A component is a configuration of engineering objects forming a single unit for the purpose of location in space and embodying a set of functions. A component has some well-understood, possibly rapidly moving, location, and it may be composed of two or more (sub)components.

Each component has one or more ports where connections to other components are made. Any given port on a component may expose one or more provided or required service interfaces.

In some contexts, especially the derived Connectivity Viewpoint, a component may be called a node.

A **connector** is a thing which links two or more things together. A connector may be rigid, flexible, hinged, rotational, articulated, or simply energetic. Connectors connect components at a port.

In some contexts, especially the derived Connectivity viewpoint, a connector may be called a communications link.

Structural Objects are concrete elements that support the location and orientation of instruments in a spacecraft in a Structural Viewpoint (see section 8). Structural Objects usually have no data interfaces, but often have thermal interfaces that appear in a Thermal Viewpoint.

Spacecraft includes spacecraft that travel in space, rovers, habitats, and other elements in space or on a remote planetary surface.

6.6 TYPICAL OBJECTS IN THE PHYSICAL VIEWPOINT

Table 6-1 shows typical Physical Objects used in space systems. These are provided only as examples, and any given system may decompose these differently or use other names for the same objects. These examples intentionally show only very high-level objects, which will typically be further decomposed during the design process.

Depending on the system, Physical Objects may be decomposed into component objects. How Physical Objects are decomposed into component objects depends heavily on the system design and local practice, and it is beyond the scope of this reference architecture to define specific decompositions of these Physical Objects.

Table 6-1: Example Physical Objects

Physical Objects	Description
Spacecraft	A moveable device that may communicate with other spacecraft or with ground stations during a mission. Spacecraft can appear in the Connectivity Viewpoint as nodes. The Structural Viewpoint may describe the internal structure of a spacecraft.
Thruster	An actuator that provides a reaction force to a spacecraft in flight by ejecting mass. If the force passes through the center of mass of the spacecraft, then the thruster moves the spacecraft along a trajectory; otherwise, the thruster changes the attitude of the spacecraft.
Star Tracker	A sensor that recognizes arrangements of stars and reports its orientation in celestial coordinates.
Inertial Measurement Unit	A device that measures and reports changes in orientation and location during flight.
Torque Bar	An actuator that interacts with ambient magnetic fields to apply torque to the attitude of a spacecraft in flight.
Computer	A device that provides software implementations (often called instances) of Functional Objects.
Bulkhead	A part of the structure of a spacecraft that acts as a wall.
Antenna	A device that sends and receives electromagnetic signals. In the Connectivity View, antennas form the interfaces for communication links between spacecraft and between spacecraft and ground stations
Subnetwork	A device for electrical communication between components onboard a spacecraft in the Connectivity View.

6.7 EXAMPLES OF A SPACE SYSTEM DESCRIBED WITH PHYSICAL VIEWS

Figure 6-4 shows a simple set of Physical Objects used in a space system together with the interactions that occur among them.

Figure 6-4 represents a connection for on-orbit servicing using a structural conduit and showing fluid flow. Other derived views of this same set of Physical Objects might be shown, such as a Connectivity View showing pressure telemetry sent from the fluid pump to signal when to stop pumping.

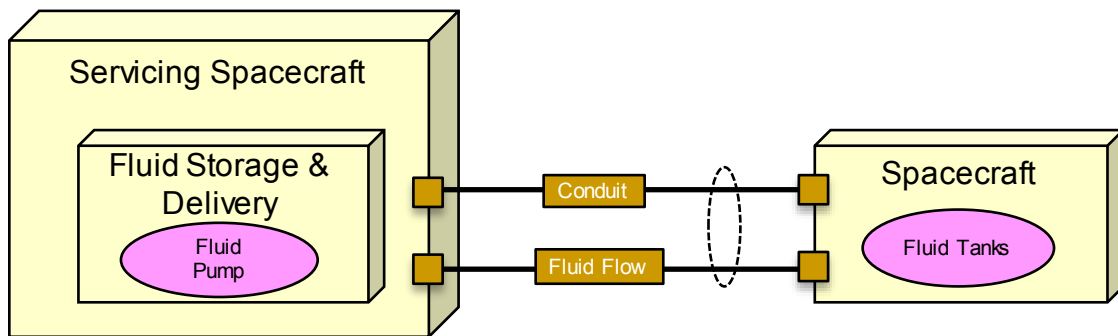


Figure 6-4: Simple Example of Physical View Connector

A simple example of a Physical View is shown in figure 6-5, in which a few 3-D boxes represent components attached to a flat panel (which is also a component). Figure 6-5 omits details that will be resolved by engineers who implement the architecture, such as coordinate systems for the science instrument and for the spacecraft. The connections between the boxes and panels might be welded or bolted mating surfaces, but these details are not shown in this view.

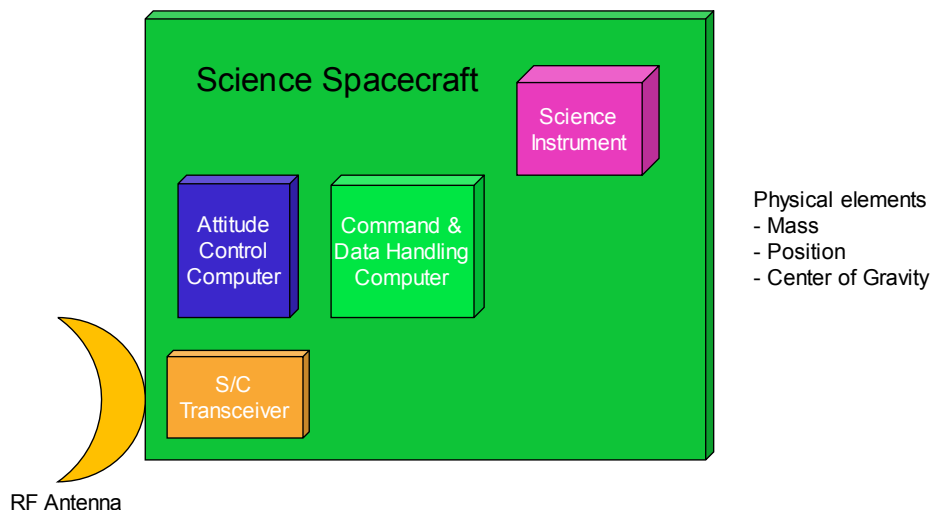


Figure 6-5: Physical View Example Composition

This 3-D representation of components was chosen because this viewpoint deals with Physical Objects. Components are physical engineering objects; the other primary elements that may appear in the Physical Viewpoint are explicit connectors. If more architectural details are wanted, it may be necessary explicitly to include both a structural viewpoint and communication viewpoint in the architecture description.

6.8 SECURITY TOPICS IN THE PHYSICAL VIEWPOINT

In the Physical Viewpoint, the Physical Objects that are used to implement security functions, policies, and approaches are defined. Threats relating specifically to elements of the physical viewpoint should be considered (reference [17]). These may include: access control interfaces on data systems, such as authentication, source-level encryption, and key management subsystems. Some of these may be shown as Physical Objects in their own right (e.g., PKI management system), or just as attributes of other Physical Objects (e.g., access control on a facility). Passive security features may include warnings to integrators of devices that require special security, for example, using red and black wires for secure subnetworks and common subnetworks, respectively.

Examples of specific physical security elements:

- firewalls;
- routers;
- security modules;
- walls, doors, locks;
- badge readers;
- Virtual Private Network (VPN) servers;
- ElectroMagnetic Interference (EMI) shielded spaces;
- identity devices;
- network/security devices (Intrusion Detection System [IDS], key management);
- spoofing signals, etc.

7 CONNECTIVITY VIEWPOINT—DERIVED

7.1 OVERVIEW

The Connectivity Viewpoint¹⁴ is derived from the Physical Viewpoint, and it is used to represent the communication aspects of physical elements that operate either in space, where connections between elements, the physics of motion, and interactions with forces in the external environment are considered, or on the surface of the Earth or some other physical body. The Connectivity Viewpoint deals with the composition of these physical elements and specifically their communication connections and interactions to provide for end-to-end communications paths and network connections.

For the description of space data systems, the Connectivity Viewpoint is where consideration is given to nodes (components), links (connectors), computational and data transport functions, external forces that affect communications (motion, interference), and other considerations related to the engineering of data system communications functionality and performance.

Frequently there are elements of space data systems that are in motion through space, and consequently connectivity issues associated with pointing, scheduling, long Round-Trip Light Times (RTLTs), intermittent visibility, and low signal-to-noise ratios all must be considered. All these challenges must be dealt with using special protocols, functionality, and controls. The Connectivity Viewpoint is used to address all these aspects of space data systems.

7.2 CONCERNS

Concerns for the Connectivity Viewpoint are:

- the mechanisms and functions required to support distributed communications between objects in the system;
- the selected allocation of functions to the nodes of the system, including their implementation choices and constraints on implementation, connections, configuration, and operations imposed by the communication links and the environment;
- the behavior and performance of elements in the system, including their capabilities, physical motion, and their interactions with the physical environment.

¹⁴ The Connectivity Viewpoint is one aspect of the Engineering Viewpoint of RM-ODP. It is called out separately in RASDSv2 because it exposes broad communications and related physical issues and constraints in the design of space data systems, which are distinct from those encountered in typical terrestrial distributed systems.

7.3 CONCEPTS FOR THE CONNECTIVITY VIEWPOINT

The **Connectivity Viewpoint** is an engineering view on a space data system that shows engineering objects, which may be hardware or software. The components and connections of the Physical Viewpoint correspond directly to the terms ‘nodes’ and ‘links’ in the Connectivity Viewpoint.

The Connectivity Viewpoint is focused on a **node** and **link** view of a system, the composition of the nodes, the physical connections among nodes used for communications, their physical and environmental constraints, and their physical dynamics as it affects communications.

The Connectivity Viewpoint also describes how the abstract functional design described in the Functional Viewpoint may be implemented as software engineering objects, that is, applications or software components, or hardware engineering objects, and how these are allocated to the major hardware engineering objects (nodes) of the system. These engineering objects are the ‘to be implemented’ versions of the Functional Objects which were described in detail in section 5.

In the Connectivity View, a space data system is depicted with nodes and the physical communications connections among them (links). This view is also used to describe how these nodes move through space and the effects that the environment has upon their behaviors.

This view also includes description of certain aspects of physical behavior of the system, such as spacecraft trajectory, communication view periods, orbits, or the motion of the physical body on which the element is located. The physical behavior is important for understanding the communications challenges from the physical environment in which the systems operate, particularly the motion, discontinuous or disrupted connectivity, and extremely distant and broad distribution of physical devices. Specialized protocols and systems design are needed to provide reliable and secure communications to deal with many aspects of the space physical environment.

7.4 CHARACTERISTICS OF CONNECTIVITY OBJECTS

7.4.1 GENERAL

The primary objects shown in the Connectivity Viewpoint are physical nodes and the links that connect them. The abstract node and its interfaces are shown in figure 7-1.

Each of these interfaces is associated with one or more links attached to the node. Physical links attach to physical ports on nodes, and these links provide the means for nodes to communicate. Each node typically implements one or more functions (defined in a functional view), either as software or hardware engineering objects. The allocation of the set of Functional Objects to nodes and determination of their implementation choices is a primary activity associated with the development of the Connectivity Views for a system.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

The services associated with a node are determined by the functional elements that the node implements. So the functional behavior of a node is determined by the functional elements implemented in the node and the mechanisms that enable interactions with functional elements in other nodes. The functional elements allocated to a node have associated logical interfaces, and these become associated with the physical ports on the node. One physical port may support more than one logical interface, just as an Ethernet port may support multiple services like file transfer, web browsing, or database services.

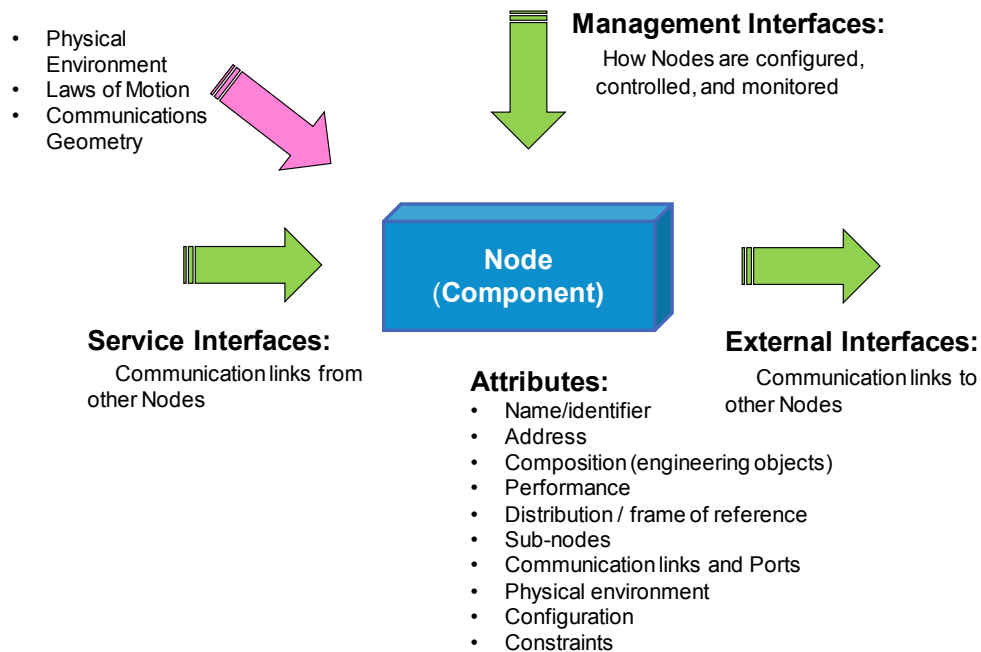


Figure 7-1: Overview of Connectivity Object (Node)

From a computational point of view the physical behavior of a node is determined by its performance characteristics, its processing speed, internal bandwidth, data paths, memory sizes, or other performance-related attributes. The performance of the engineered system, either in a local or an end-to-end sense, may be specified once the performance capabilities of the nodes and links have been specified, the performance requirements of the allocated functions have been determined, and the effects of the environment have been characterized.

When viewed at the coarsest level of granularity, some nodes of a system, such as a spacecraft, will exhibit physical behavior that is determined by the physical forces acting upon the node. These forces may be propulsive or gravitational, or they may be caused by other elements in the environment that determine the velocity, direction of motion, acceleration, or mobility of the spacecraft. The physical location and behavior of the spacecraft (orbit, trajectory, path), the performance of some of its components (e.g., antenna aperture, transmitter power, receiver sensitivity), and the physical characteristics of the environment, all exert a strong influence on the performance of the communications systems and the behavior of the links.

The protocols that implement the communication stacks are described in the Communications Viewpoint and they are selected to deal with these behavioral and environmental factors.

7.4.2 KEY OBJECTS AND RELATIONSHIPS

The following elements may appear in the Connectivity Viewpoint:

- engineering objects (nodes, links, applications):
 - nodes (hardware engineering objects, processing and other computing and data resources, ports, performance, and other associated physical behavior):
 - types: hardware objects, composite hardware objects, ports;¹⁵
 - attributes: name, type, location (place, trajectory, orbit), laws of motion, available resources, physical interfaces, capabilities (e.g., processor speed, throughput, bandwidth, storage capacity, aperture size, etc.), allocated functions, constraints;
 - links (connections between nodes, associated physical behavior and properties):
 - types: space link (RF or optical), physical link (e.g., point-to-point, bus, network, copper, fiber, etc.);
 - attributes: name, type, end points (port on node), physical interfaces, performance (e.g., throughput, bandwidth, frequency band, RTLTL, pointing and view periods, signal attenuation, constraints, environmental effects, etc.), access and ownership;
 - applications (software engineering objects, behavior, and processing/resource requirements [may be layered]):
 - types: software engineering objects, composite software engineering objects;
 - attributes: name, type, algorithms, implemented functions, allocation/deployment to nodes, required resources (e.g., memory, Central Processing Unit [CPU], storage), implemented interfaces (provided and required), implementation language, Operating System (OS)/framework dependencies, constraints;
- Relationships (composition, interfaces, constraints, configurations, allocation);
- Environment (physical environs, physical forces [gravity and others], physical interactions and effects);

¹⁵ Ports, and their characteristics, may be described formally using an Electronic Data Sheet (EDS) (reference [45]).

- Information (defined representations of data that are exchanged among engineering objects, where formal specifications of data architecture are found in the Information Viewpoint).

NOTE – The Connectivity Viewpoint is focused upon space data systems and communications aspects. All other physical aspects are addressed elsewhere in the Physical Viewpoint or other derived viewpoints in this document. (See *Toward a Framework for Modeling Space Systems Architectures*, reference [19], for more information on one such extended approach.) The Connectivity Viewpoint in this document corresponds to the Navigation, Telecomm, and Data System Views in reference (reference [19]).

7.4.3 ONTOLOGY OF CONNECTIVITY OBJECTS

Figure 7-2 shows the ontology of Connectivity Objects. These are derived from the Physical Viewpoint, but with specialization of the terms to address connectivity concerns.

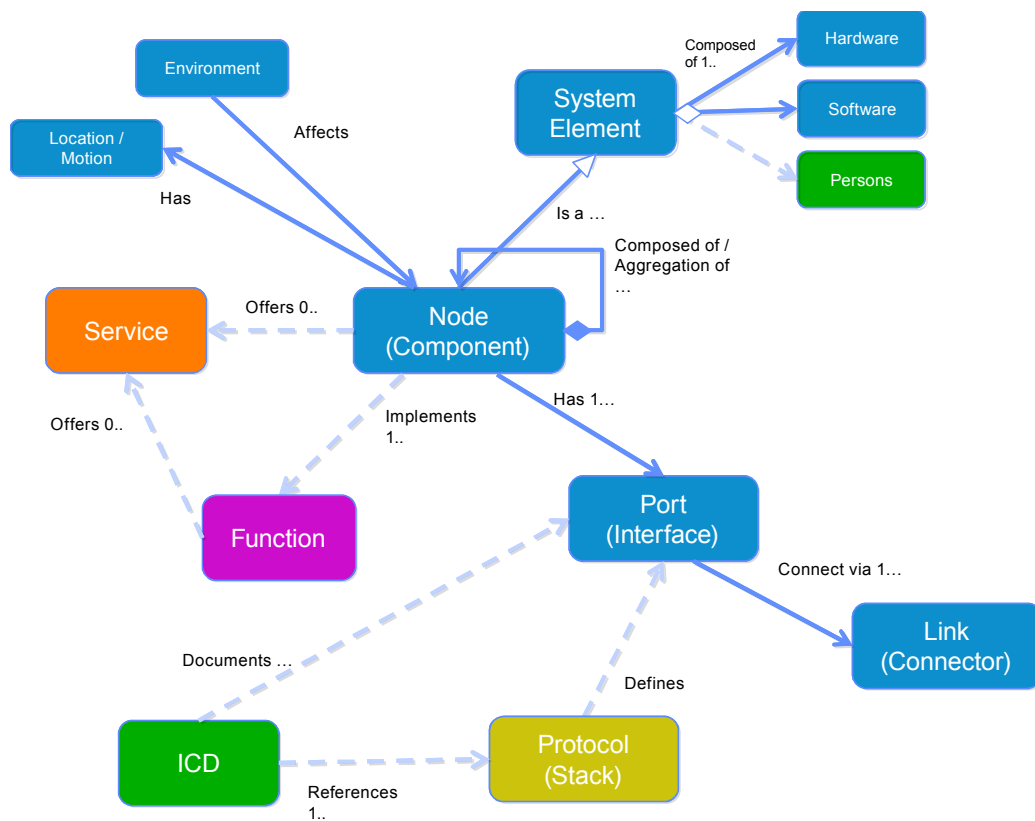


Figure 7-2: Ontology for Connectivity Viewpoint

Connectivity nodes attach to links at a port or component interface. The communications characteristics of the port may be defined by a protocol stack. The port (interface), particularly if it involves two different systems with different ownership, may be documented by an Interface Control Document (ICD).

7.4.4 REPRESENTATION OF CONNECTIVITY OBJECTS

In RASDSv2 diagrams Connectivity Objects are represented using the drawing style shown in figure 7-3. This diagram contains terms ‘node’ and ‘link’, which are used in the Connectivity Viewpoint, and the focus here is on communications connections, protocols, and data flows, not data content.

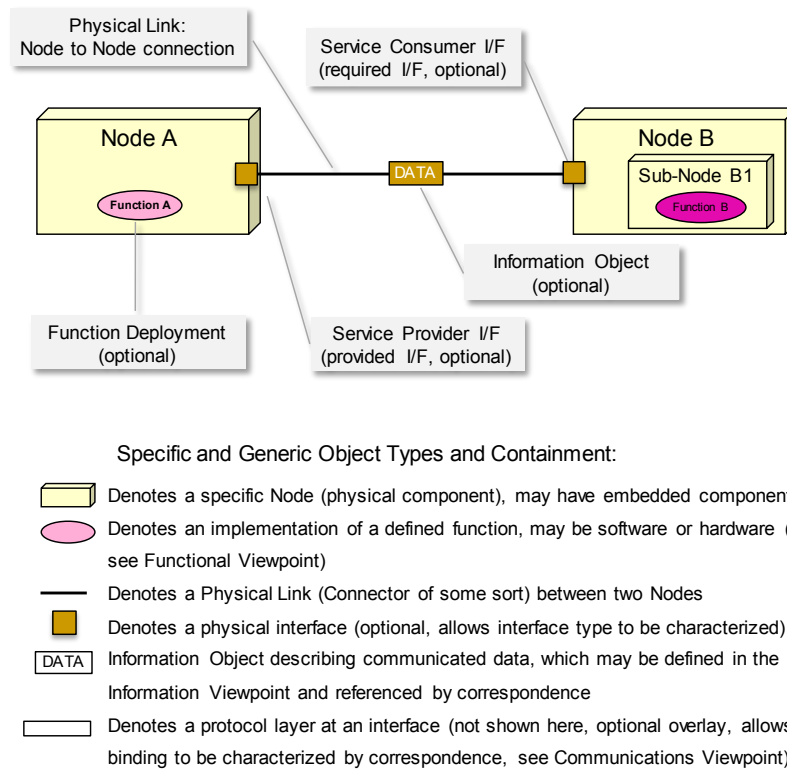


Figure 7-3: Representation of Connectivity Objects

Nodes will explicitly connect via some link. The mass, power, thermal, and other physical properties of nodes are inherited from the Physical Viewpoint.

The link represents some communications constraint. This may be dynamic or static, and it may be concretely physical (cable, copper, fiber) or energetic (RF, optical).

May show allocation of implemented functions in hardware (a sub-component) tied by correspondence to the Functional View where they are defined.

May explicitly identify the Data Objects (defined in Information Views) exchanged by the functions using the link (defined in these views).

May explicitly identify the physical interface or connection type.

May explicitly describe the relationships between nodes and the environment.

7.5 TERMS FOR THE CONNECTIVITY VIEWPOINT

7.5.1 ATTRIBUTES FOR THE CONNECTIVITY VIEWPOINT

A **Unique Identifier** is a value used in specified fields of CCSDS-defined (or other) Data Link Layer data structures. It provides a unique identifier for the node.

A **network address** is an identifier for a node or host on a telecommunications network. Network addresses are designed to be unique identifiers across the network, although some networks allow for local, private addresses, or locally administered addresses that may not be unique. Special network addresses are allocated as broadcast or multicast addresses. These too are not unique.

System performance is the amount of useful work accomplished by a system. Outside of specific contexts, performance is estimated in terms of accuracy, efficiency, and speed of executing computer program instructions or transferring data.

Space environment creates conditions in space that affect the design and operation of spacecraft. Effects on spacecraft can arise from temperature extremes, radiation, space debris and meteoroid impact, upper atmospheric drag, spacecraft electrostatic charging, and gravity.

The **configuration** of a system refers to the arrangement of each of its functional units, according to their nature, number, and chief characteristics. Often, configuration pertains to the choice of hardware, software, firmware, and documentation as well as the selection of control parameters.

A **Constraint** is a limitation or implied requirement that limits the design solution or implementation, is not changeable by the enterprise, and is generally non-allocable.

7.5.2 OTHER TERMS FOR THE CONNECTIVITY VIEWPOINT

A **node** is a physical hardware engineering object that is a run-time computational resource and generally has at least memory and often processing capability. Run-time software engineering objects reside on nodes. A node has some well-understood, possibly rapidly moving, location. A node may be composed of two or more (sub)nodes.

All nodes are hardware engineering objects, but not all hardware engineering objects are nodes. Larger discrete items of hardware in a space system are termed nodes. From the perspective of systems architecture descriptions, below some level of granularity it may not be useful to describe minor hardware elements as nodes, but rather components.

A **Link** is the locus of relations among nodes. It provides interconnections between nodes for communication and coordination. It may be implemented by a wired connection or with some RF or optical communications media. Links implement the primary function of transporting data. Links connect to nodes at a port.

Links are engineering objects, but only some of them are hardware. Some links, such as an Ethernet cable or a CPU backplane are implemented as hardware. Some links, such as an RF or optical link, are a physical effect produced by hardware engineering objects. These links are not physical devices, but are physical manifestations that can be sensed, measured, and analyzed for their information content.

A **port** is the physical element of a node where a link is connected. Nodes may have one or more ports. Each port may connect to one or more physical ports on (sub)nodes that are contained within the node. Where needed a port may be represented as a simple small rectangle at the edge of a node.

A single physical link between two nodes may carry one or more logical connections between applications implemented on those nodes.

An **application** consists of one or more pieces of software designed to perform some specific function; it is a configuration of interacting engineering objects.

The process of **allocation** is mapping between one set of model elements and another. The mapping is often performed as part of the design process to refine the design. Typical examples of allocation include allocation of functions to nodes, logical to physical components, logical to physical links, and software application instances to hardware.

Tracking Station is a node in a Connectivity Viewpoint that occupies a fixed location on a planet (including Earth) or asteroid.

7.6 TYPICAL CONNECTIVITY OBJECTS

Table 7-1 shows typical nodes that are used in space systems. Which nodes are used in any given space system may differ from system to system, and the following list shows only typical nodes used in many space systems.

Table 7-1: Typical Nodes

Nodes	Description
Spacecraft	A spacecraft (or a lander, rover, balloon, etc.) used to achieve mission goals (e.g., observations or experiments).
Relay satellite	A spacecraft (or a lander) that relays data between spacecraft and a tracking network or between different sets of spacecraft. It may not exist as a Physical Object in all space systems.
Instrument	A component of a spacecraft used to achieve mission goals (e.g., observations or experiments). It may not exist as a Physical Object in all space systems.
Computer	component of a spacecraft or ground system used to process data.
Data storage system	Subsystem used to store and manage data. It may not exist as a separate Physical Object in all spacecraft or ground systems.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Nodes	Description
Ground Tracking Network	Typically a multi-mission subsystem that may be composed of one or more nodes with one or more tracking stations, and possibly a network control center. It is used for communicating with spacecraft and performing radiometric measurements against spacecraft.
Tracking Station	A subsystem of a ground tracking network with an aperture that is used to track spacecraft, transmit commands, receive telemetry, and optionally to produce radiometric and tracking data.
Spacecraft Control Center	A center used for controlling one or more spacecraft.
Spacecraft control facility	A facility that is part of a mission operations system that is used to plan, control, and monitor spacecraft operations.
Instrument control facility	A facility that is part of a mission operations system that is used to plan, control, and monitor instrument operations. It may not exist as a separate facility in all enterprises.
Orbit determination facility	A facility that is part of a mission operations system that is used to analyze radiometric tracking data and to determine the orbit and attitude of a spacecraft. It may not exist as a separate facility in all enterprises.
Trajectory design facility	A facility that is part of a mission operations system that is used to design a spacecraft trajectory and plan maneuvers. It may not exist as a separate facility in all enterprises.
Mission planning facility	A facility that is part of a mission operations system that is used to create, control, and monitor mission operational plans. This may include overall observation and mission scenario planning. It may not exist as a separate facility in all enterprises.
Science facility	A facility that requests activities of a spacecraft and analyzes data obtained from that spacecraft. It may not exist as an enterprise object in all space systems.
Data analysis facility	A facility that is part of a mission operations system that is used to process instrument data and to perform a variety of additional data analyses. It may not exist as a separate facility in all enterprises.
Data Archive Center	A facility that archives data obtained from spacecraft and delivers requested data to a science institute. It may not exist as an enterprise object in all space systems.

Node functionality is implemented by creating a set of instantiated functions, in software, or possibly even in hardware or firmware such as an FPGA. Good practice identifies generalized sets of implemented Functional Objects as an aid to re-use. Specialized sets are developed only as needed. There are several different kinds of software interface design patterns that are now used in software development. It may be noted that even the cloud and data center approaches that are currently in vogue are still composed of nodes and links, just with different deployment and decomposition patterns.

Table 7-2 shows typical links that are used in space systems. Which links are used in any specific space system differs from system to system, and the following list only shows typical links and attributes that are considered in many space systems.

Table 7-2: Typical Links

Links	Description	Attributes
Space Link	A link between a node in space (e.g., a spacecraft) and a node on the ground (e.g., a ground station), or a link between two nodes in space (e.g., between two spacecraft).	<ul style="list-style-type: none"> – continuous vs. episodic connectivity; – pointing and view periods; – frequency band (RF) or wavelength (optical); – delay and signal attenuation; – single vs. multiple access; – bit rate, possibly variable.
Ground Link or Network	A link between two nodes or a network among multiple nodes on the ground.	<ul style="list-style-type: none"> – wide area or local area; – dedicated or public; – single vs. multiple access; – bit rate.
Onboard Link or Bus	A link between two nodes or a bus among multiple nodes on the same spacecraft.	<ul style="list-style-type: none"> – single vs. multiple access; – redundancy; – bit rate.

7.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH CONNECTIVITY VIEWS

Figure 7-4 shows the nodes used to support a mission. In this figure just the major nodes are shown and named, along with the high level characteristics of the links used to connect them. The three 3-D boxes represent nodes and the links between them are shown as solid lines. This 3-D representation of nodes is used because this view deals with Physical Objects. The other primary elements that appear in the Connectivity Viewpoint may be software and hardware engineering objects that implement (and correspondence to) to Functional Objects defined in a related Functional View.

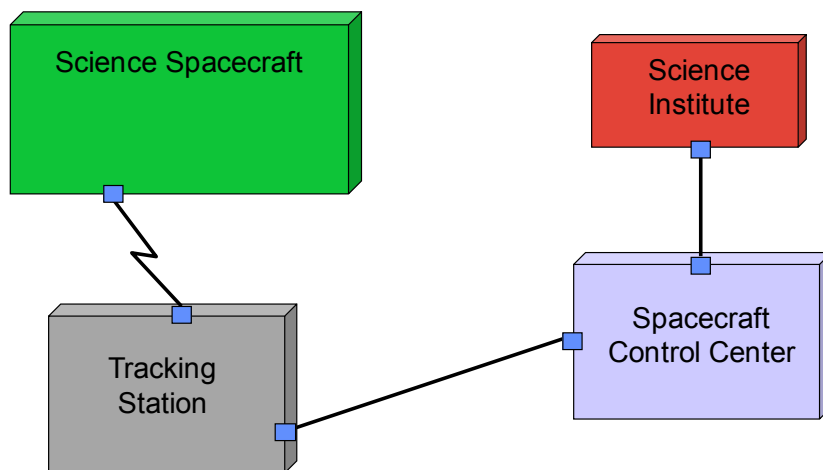


Figure 7-4: Simple Connectivity View (Nodes for Some Mission)

A more complex example of this same Connectivity View is shown in figure 7-5. As part of the engineering process of designing the system, abstract functions are allocated to physical nodes and implementation choices are made about use of software or hardware. In the Connectivity Viewpoint, representations of Functional Objects are shown as engineering objects, either as physical hardware (nodes or hardware engineering objects) or as software (software engineering objects) that are allocated to the nodes of the system.

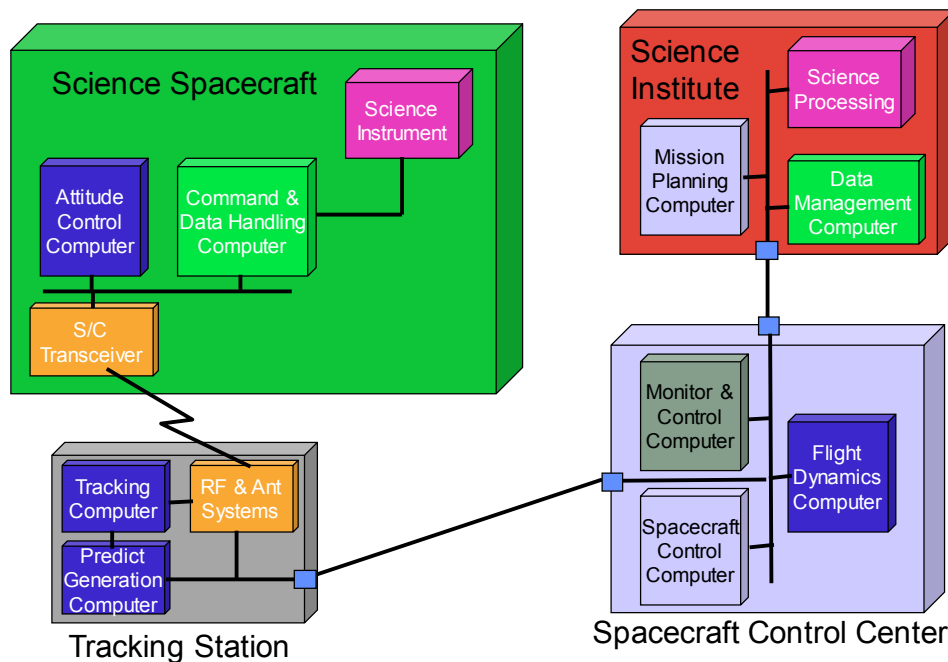


Figure 7-5: Connectivity View with Node Details

When this level of functional allocation must be described, the allocation of Software engineering objects (software or applications) may be shown by overlaying a representation of the implemented functions on the nodes shown in a Connectivity View.

Figure 7-6, below, shows one possible decomposition of the nodes used for Mission A into example component nodes. For clarity the nodes could be colored like the corresponding nodes in figure 7-5 but there is no requirement to do this nor is there anything special about these colors. Any given architecture document may adopt its own color and decomposition hierarchy and naming conventions.

In figure 7-6 the nodes owned by two agencies are shown, along with an overlay of the functions deployed on the different nodes. In reality the ‘as implemented’ functions might be allocated to different hardware nodes than those shown and no one to one mapping can be assumed. Data flows are labelled as to data types.

Of course, the nodes shown in figure 7-6 may be further decomposed into lower-level nodes in separate views, with their own internal links where this level of detail is required. Many systems engineering disciplines provide a hierarchy of names to describe different levels of

decomposition of nodes within a system. The IEEE/ISO/IEC Systems Engineering Planning document (reference [3]) defines the terms system, product, subsystem, assembly, component, and subcomponent, but other terms for system decomposition may be appropriate for different projects, and none is prescribed here.

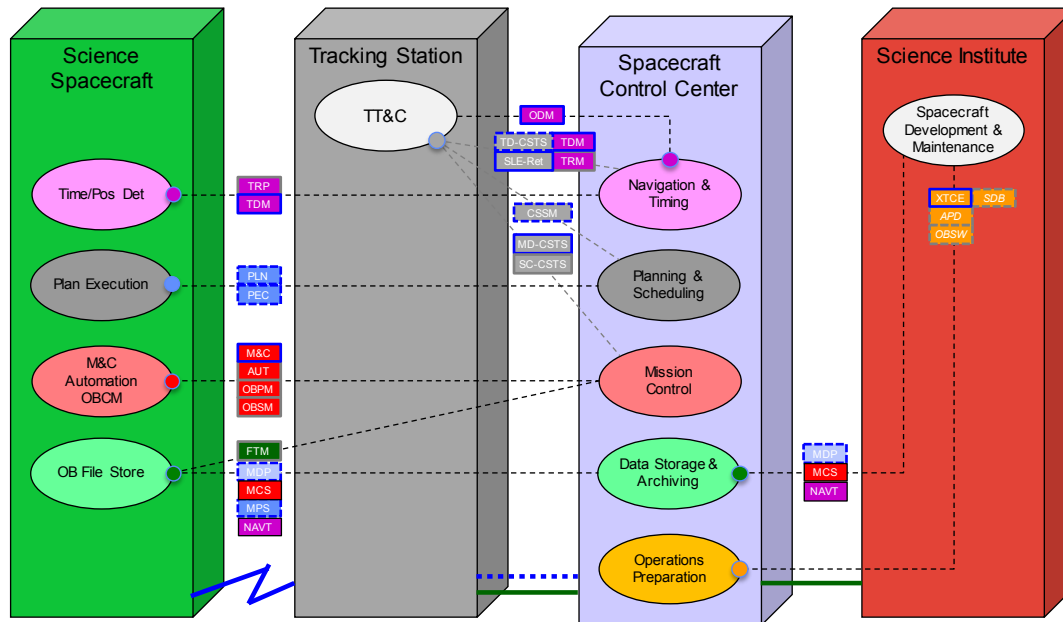


Figure 7-6: Connectivity View with Allocated Engineering Objects

This same methodology supports definition of Systems of Systems views, but no specific recommendation is made other than to suggest starting at the highest level overview of the collection of systems and then using successive decomposition and clear specification of interfaces at each level, in as many views and expanded details as are required. In *Architecting Principles for Systems-of-Systems* (SoS—reference [20]) one of the key points of emphasis is that each system has its own purpose and that the leverage point for architecting the SoS is at the communications interfaces between these entities. This is, of course, also true of all external and internal interfaces of any system elements. A paper that uses RASDSv2 methods and adopts SysML for modeling a system-of-systems architecture is *NASA Integrated Network Monitor and Control Software Architecture* (reference [29]).

As an example of how allocation works, figure 7-7 shows how the functionality defined by the set of example Functional Objects shown in figure 5-5 might be allocated to the high level nodes that were just shown in figure 7-7. This Connectivity View diagram shows how the functional elements, implemented as engineering objects, might be allocated to nodes. On a more detailed view the explicit choice of implementation options for implementing these functions in hardware or software engineering objects might be shown.

Once the combination of the performance support requirements of the implemented engineering objects and the performance capabilities of the nodes and links have been defined analysis of the end-to-end performance of the system may be determined.

Figure 7-6 could be redrawn to represent an autonomous spacecraft, with specialized copies of the Planning, Directive Generation, and Monitor and Control functions moved on-board. Exploring the implications of these allocation options on system functionality, performance, and support requirements is possible once all the elements of the Functional View have been identified and the allocations of these within a Connectivity View have been specified.

Connectivity views have other uses during trade studies to select between hardware and software implementation options. Consider the problem of implementing an image compression function for a high-performance telemetry system. Two possible approaches might be to implement the compression function directly in software, perhaps on the Command and Data Handling (C&DH) processor, or to implement a hardware compressor that might be a board integrated into the flight data recorder.

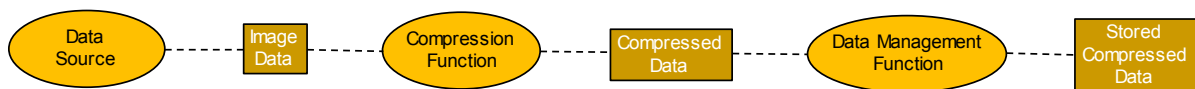


Figure 7-7: Functional View of Image Compression

Both the hardware and software options would implement the identical functional flow of data as shown in figure 7-7, but the connectivity views look significantly different, as shown in figures 7-8(a) and 7-8(b) and the performance characteristics of these two options would also be significantly different.

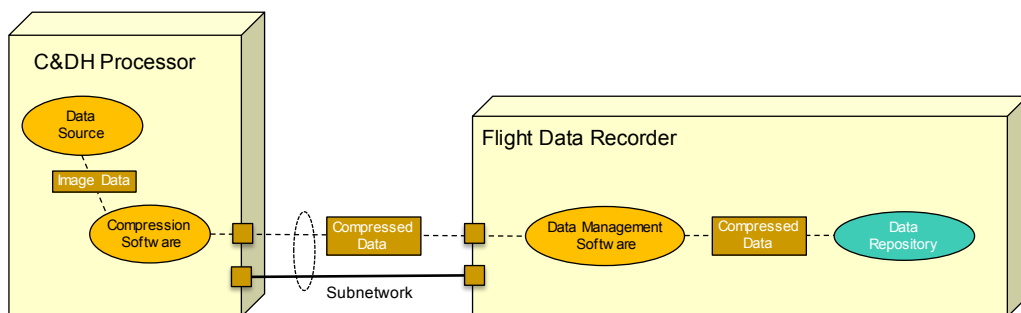


Figure 7-8(a): Connectivity View of Software Compression Approach

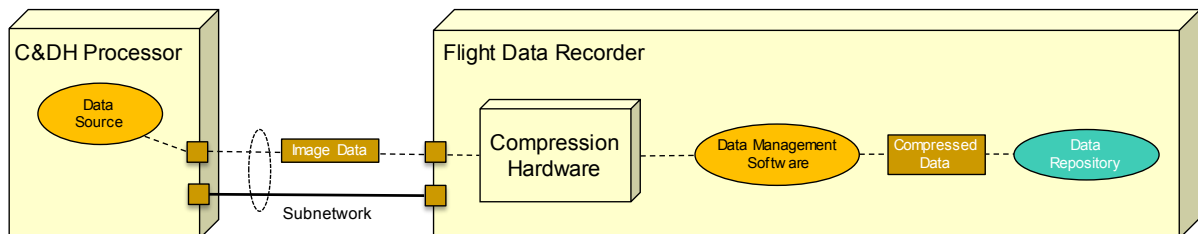


Figure 7-8(b): Connectivity View of Hardware Only Compression Approach

Of course, there will also be mass, power, implementation and evolvability issues associated with these choices that must also be factored into any final design decision.

7.8 SECURITY TOPICS IN THE CONNECTIVITY VIEWPOINT

In the Connectivity Viewpoint security topics are dealt with by the physical elements that are used to implement security policies and barriers. These might include: secure routers and firewalls, hardware security modules, and possibly physical boundaries such as shielded rooms or air gaps. At the time of publication specialized security components and approaches may be deployed, such as two-factor identity tokens, Continuous Diagnostics and Mitigation (CDM) and Security Information and Event Management (SIEM) tools, virtualization platforms, and cloud deployments. The kinds of protocol entities that may implement elements of security functionality, such as security protocols or routing filters, will be addressed in the Communications Viewpoint.

Examples of specific connectivity security elements:

- firewalls;
- routers;
- backbone and leaf networks;
- primary and backup network services;
- Wi-Fi servers access point;
- VPN servers;
- compute servers/data centers;
- user devices;
- identity devices;
- network/security devices (IDS, etc.);
- Radio Frequency (RF)/optical connectivity.

8 STRUCTURAL VIEWPOINT—DERIVED

8.1 OVERVIEW

The Structural Viewpoint is derived from the Physical Viewpoint and is used to describe the structural aspects of physical elements and the connectors that physically tie an assembly together, and that allow for articulation of parts. Because the locations and orientations of instruments are important for interpreting their commands and telemetry, the structural viewpoint may also be used to address the mounting of sensors and actuators within in an assembly.

8.2 CONCERNS

Concerns for the Structural Viewpoint are:

- a physical decomposition of the system into objects that form an assembly and that interact at interfaces;
- ability of structure to survive launch, the space environment, and landing;
- ability of structure to provide required agility and field of view for pointing instruments.

8.3 CONCEPTS FOR THE STRUCTURAL VIEWPOINT

The **Structural Viewpoint** of a space system focuses on the physical aspects of structures and articulation of the parts of an assembly. This Viewpoint addresses Structural Objects, their behavior, the physical connections between them, and their physical interfaces and interactions.

The behavior of a Structural Object is the set of actions performed by this element to achieve an objective. A **Structural Object** performs actions to achieve an objective of a space system, and this may involve rigidity, mechanical movement, thermal conduction, or shielding.

Structural Views describe Structural Objects, how assemblies are held together, how instruments are held in a particular orientation, and how instruments may be moved into adaptive working positions.

- The orientations of instruments relative to the coordinate system of the assembly makes it possible to transform measurements, images, and forces to the coordinate system of the assembly.
- Flexibility of Structural Objects enables estimation of error bounds on coordinate transformations.
- Location, orientation, and articulation of Structural Objects determines the field of view of instruments mounted on the structure.

- Mass properties, including center of mass and inertial matrix, of Structural Objects and of onboard instruments determine the response of an assembly to forces such as propulsion and torques.
- Articulation of Structural Objects can be measured by sensors in any control loops, but must be estimated from commands in open control loops.
- Information Objects that carry measurements from sensors and commands to actuators can appear in a view that shows the aggregation of Structural Objects in an assembly. The Information Objects that appear in the Structural Viewpoint are representations of the Information Objects that are fully described in the Information Viewpoint. The details of how these Information Objects are defined, described, and controlled are covered in the Information Viewpoint (section 10).

A Structural view shows stasis, control of articulation, and other attributes of Structural Objects. In the engineering of any given system, allocation of instances of control functions to articulated Structural Objects may be represented. The physical means for providing communications among implemented functions are treated in the Connectivity Viewpoint (section 7), as are the physical attributes of the connections and their behavior.

8.4 CHARACTERISTICS OF STRUCTURAL OBJECTS

8.4.1 GENERAL

The fundamental features of Structural Objects and their interfaces have already been shown in the Physical Object overview, figure 6-1. No additional features are needed for the Structural Viewpoint, only added specificity and a deeper focus on the mechanical and structural aspects of the system.

The interfaces of Structural Objects are classified into four categories: service interfaces, external interfaces, environmental interfaces, and management interfaces. Every Structural Object has one or more interfaces through which it interacts with the physical environment, according to laws of motion and energetic exchanges, such as propulsion and solar heating.

8.4.2 KEY OBJECTS AND RELATIONSHIPS

The following elements may appear in the Structural Viewpoint:

- Structural Objects that provide energetic exchanges;
 - nature of energetic exchange, such as heat, electrical power, electromagnetic radiation, structural continuity, momentum, angular momentum;
- Structural Objects that provide chassis that tie assemblies together or that provide articulation for instruments;

- instruments that implement functions (abstract set of functions, their behaviors, interfaces, and configurations);
 - types: data source, data sink, data transformation, control, planning, monitoring, analysis;
 - attributes: role, name, type, behavior, interface signature, data types handled, interaction modes, constraints, allocated requirements;
- logical links (connections between Structural Objects, connected to associated logical behavior and properties);
- relationships (configuration, precedence, control and data flows, management flows, allocations);
- information (representations of data that are exchanged among Structural Objects and control/acquisition functions, where formal specifications for exchange are found in the Information Viewpoint).

8.4.3 ONTOLOGY OF STRUCTURAL OBJECTS

8.4.3.1 Overview

The ontology of Structural Objects is shown in figure 8-1. It differs from the Physical Object ontology in the specialization of possible flows and the addition of interface standards and an Interface Control Document (ICD). The terms ‘structural element’ and ‘structural connector’ replace the terms ‘component’ and ‘connector’.

Structural components attach to connectors at an attachment point or interface. The characteristics of the attachment may be defined by an interface standard. The attachment point, particularly if it involves two different systems with different ownership, may also be documented by an ICD.

8.4.3.2 Representation of Structural Objects

Structural Objects use the same representation as Physical Objects (see 6.4.2.3 and figure 6-3). All the features of Physical Object representation that are needed for the Structural Viewpoint are already present in the Physical Viewpoint representation. The distinction is that the Structural Viewpoint is focused on the structural aspects of the physical elements and elements in Structural views may focus more closely on the structural details of physical connections.

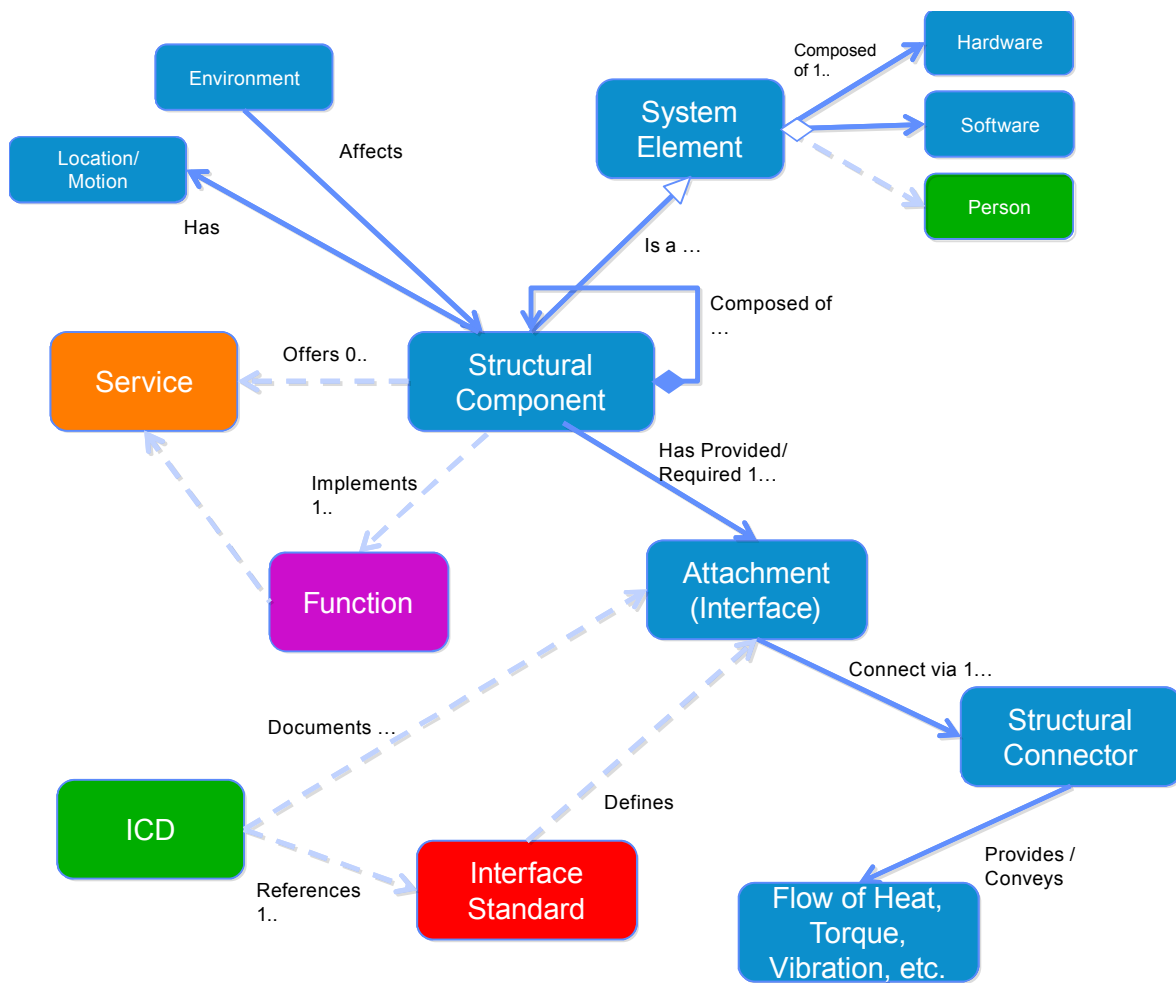


Figure 8-1: Ontology of Structural Objects

8.5 TERMS FOR STRUCTURAL VIEWPOINT

8.5.1 GENERAL

The Structural Viewpoint inherits all the terms from the Physical Viewpoint (see 6.5).

8.5.2 ATTRIBUTES FOR STRUCTURAL VIEWPOINT

Mass: The inertial property of a Structural Object.

Center of mass: The location of the balance point of a Structural Object in the coordinate system of the object.

Inertia matrix: The moments of inertia of a Structural Object arranged in an array.

8.5.3 OTHER TERMS FOR STRUCTURAL VIEWPOINT

Stress: A measure of forces (internal or external) acting over some cross-sectional area of an object.

Location: The coordinates of the origin of the coordinate space associated with a Structural Object in the coordinate system of the assembly that contains the object.

Orientation: The rotation of a Structural Object from alignment of its coordinate system with the coordinate system of the assembly that contains the object.

Flows: Movement of data or of substance or energy. Flows of data are shown using named Data Objects. The formal definitions will be found in Information views. Flows of matter or energy may appear in the Structural Viewpoint.

8.6 TYPICAL STRUCTURAL OBJECT TYPES

Table 8-1 provides examples of some common Structural Objects.

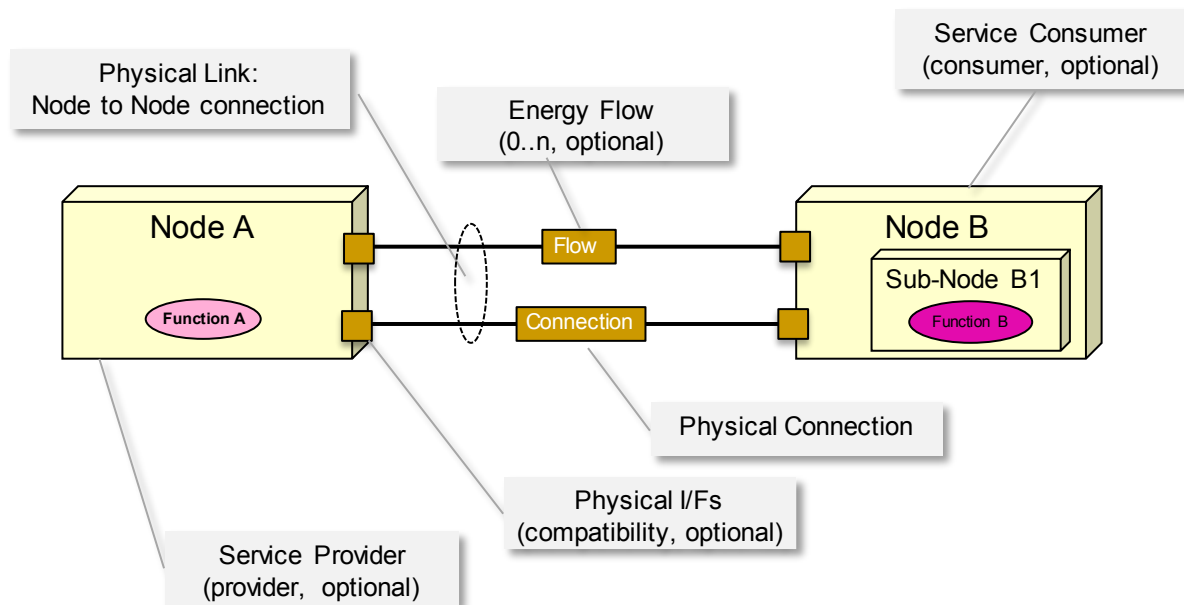
Table 8-1: Examples of Structural Objects

Bolt	A fastener that joins two or more Structural Objects through aligned drilled holes in each object.
Boom	A linear static arm that can be deployed to separate an instrument such as a magnetometer from unwanted influences such as magnetic torque bars in a spacecraft.
Bulkhead	A wall where instruments may be mounted or radiation may be shielded.
Heat Pipe	A structural element with good heat conductivity for transferring heat from a hotter object, such as a computer, to a cooler object, such as a radiator.
Instruments	Concrete elements that support a mission by sensing and/or actuating. In the Structural view, instruments obtain their locations and orientations.
Strap	Flat connector that joins two structural elements forming a corner or abutment.
Weldment	A place where two Structural Objects make contact and have been melted to join rigidly at the point of contact.
Wiring Harness	A collection of cables for distribution of power or data signals between instruments an assembly.

8.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH STRUCTURAL VIEW

As noted earlier, in 6.4.2.3, RASDSv2 only provides a rather ‘cartoon style’ of representation for these structural diagrams. This is not adequate to accurately describe engineering details, but it is useful to quickly explore architecture options and to do so in the context of all the other related views available in an architectural model. The usual mechanical and electrical engineering tools are expected to be used once trade studies of the option space have settled on the best choice.

Figure 8-2 shows a representative set of Structural Objects used in a space system together with the interactions that occur among them. The points of connection are at the mounting interfaces and the hinge contact surfaces. In this view these hinge connections provide articulation points and may also flow heat energy.



Specific and Generic Object Types and Containment:






-  Denotes a specific Node (physical component), may have embedded components
-  Denotes an implementation of a defined function, may be software or hardware (optional and referenced by correspondence, see Functional Viewpoint)
-  Denotes a Physical Link (Connection of some sort) between two Nodes
-  Denotes a Physical Interface (optional, allows interface type to be characterized)
-  Information Object describing flow or connection, which may be fully defined in the Information Viewpoint and referenced by correspondence

Figure 8-2: Representation of Structural Objects

Other views of this same set of Physical Objects might be shown, such as a Connectivity View showing thermistor telemetry, hinge angle commands, and hinge angle telemetry information flowing through a communications subnetwork to/from a thermal control function (see figure 8-3).

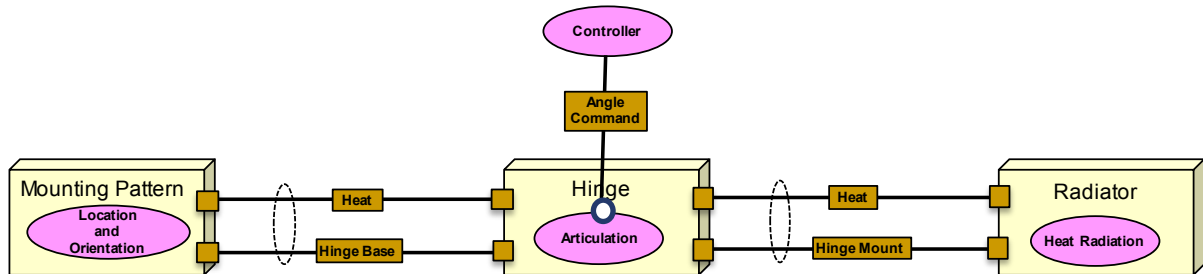


Figure 8-3: Example View of a Hinge Assembly

As a related example, figure 6-4 could be drawn with a cartoon hose, as shown in figure 8-4, or figure 6-5 could be drawn with a cartoon articulation for the antenna as shown in figure 8-5. These architectural explorations can result in a designer’s choice of implementation technology, and this must be explained in accompanying text. For example, in figure 8-5 a mechanical articulation is specified, ruling out other choices, such as a phased array.

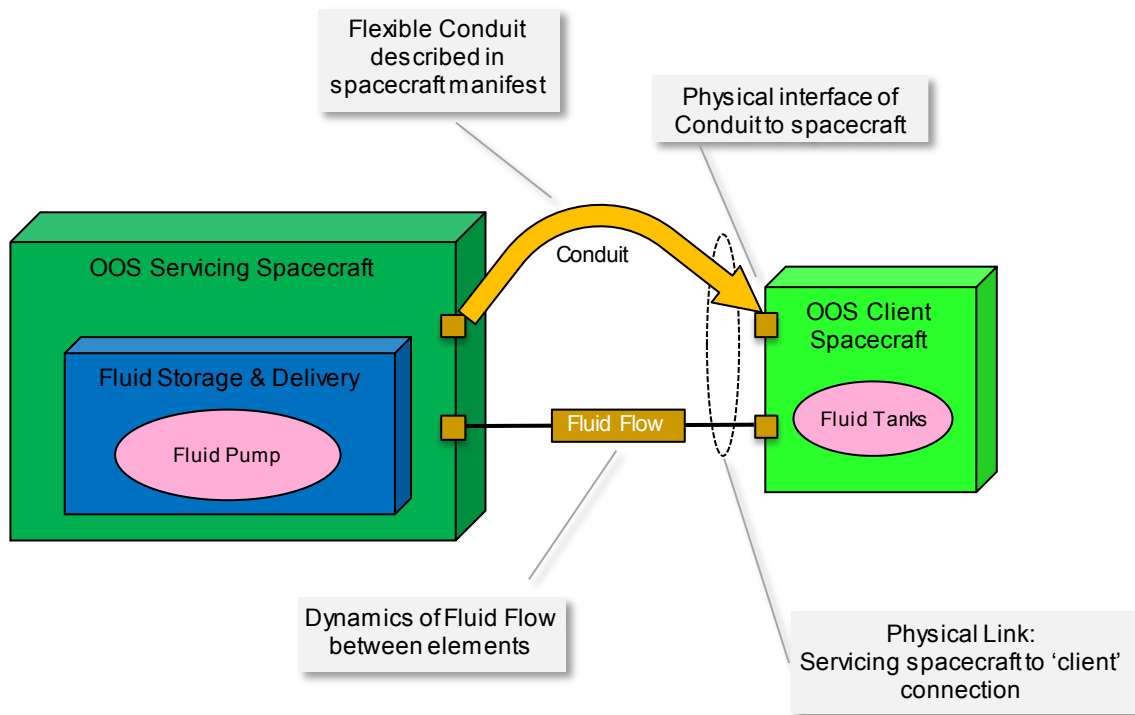


Figure 8-4: A ‘Cartoon’ Conduit Hose Added to Figure 6-4

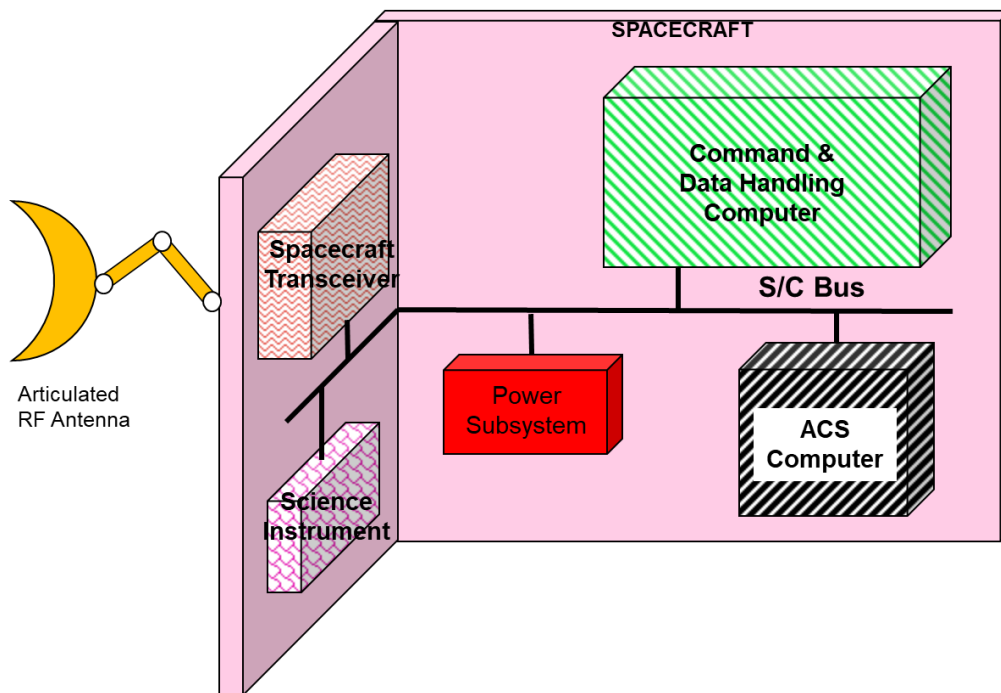


Figure 8-5: A ‘Cartoon’ Physical Bus and Articulated Antenna Added to Figure 6-5

8.8 SECURITY TOPICS IN THE STRUCTURAL VIEWPOINT

In the Structural Viewpoint, the Structural Objects that are used to implement security policies and approaches are defined. Passive security features may include warnings to integrators of devices that require special security, for example, using red and black wires for secure subnetworks and common subnetworks, respectively. RF shields may protect devices that could radiate sensitive information about their operation, or that could be affected by other RF generators, such as an antenna.

Examples of specific structural security elements:

- gates;
- doors;
- air gaps;
- RF shields.

9 COMMUNICATIONS VIEWPOINT

9.1 OVERVIEW

The Communications Viewpoint¹⁶ defines the communications protocols and the layered sets of protocols (stacks) that are required to support communications among the software or hardware engineering objects in a space data system. These protocols need to meet the requirements on performance and the constraints imposed by physical connectivity, environmental, and operational challenges. The Communications Viewpoint is used to describe these layered communications protocols and their deployment and features, and to address these technical aspects of space data systems.

This is the viewpoint where the lower five layers of the ISO seven-layer communications stack are typically addressed. Application Layer protocols, including specialized messaging of web application protocols may also be addressed using this viewpoint specification. This viewpoint is orthogonal to the other, upper-layer/application, viewpoint, where multiple perspectives on the applications in a distributed system may be provided. However, at any point where protocol details and layering at an interface must be described, views from this viewpoint should be employed.

9.2 CONCERNS

Concerns for the Communications Viewpoint are:

- the choice of communications and data transfer standards in the system;
- the end-to-end communications protocol functionality and reliability;
- design of the protocol specifications and the services they provide;
- the relevant interfaces, protocol behaviors, and interactions;
- alignment of Required and Provided Interfaces between two adjacent layers;
- behavior of end-to-end protocol design within environmental constraints;
- support for design, evaluation of suitability, and integration into the rest of the system.

¹⁶ This Viewpoint is related to both the Engineering (implemented functionality) and Technical (standards) Viewpoints of RM-ODP, but it shows the specifics of how protocols are layered to implement an interface, and the details of protocol behavior. It is addressed separately in RASDSv2 because of the need for specialized protocols to deal with the physical challenges affecting the design of systems communicating in space.

9.3 CONCEPTS FOR COMMUNICATIONS VIEWPOINT

The **Communications Viewpoint** is a space data systems engineering and technical view that focuses on the mechanisms and functions required to design and implement protocols and communications standards for a space data system, including implementation choices, and specifications and allocation of communications functionality to engineered components of the system.

This viewpoint is used to provide details in all layers of the ISO seven-layer model. The first three RASDSv2 viewpoints are more directly related to the top, or Application Layer, of the ISO Basic Reference Model (ISO-BRM—reference [13]) and the Information Viewpoint is most closely related to the representational layer of the ISO-BRM model.

In the Communications Viewpoint, the communications aspects of a space data system are depicted with Protocol Objects, and these are called Protocol Entities for alignment with ISO-BRM terminology. To understand their role in an operational context, these are often shown along with representations of the nodes, links, and software engineering objects that are defined separately in the Connectivity Viewpoint. The Communications Viewpoint describes in detail the protocols that are required for the software engineering objects to actually communicate with one another and supports descriptions of the end-to-end information system.

A **Protocol Entity** performs actions to exchange or transfer data in a space system (as distinguished from a Functional Object that generates or processes data). Protocol Entities are used to support interactions between two engineering objects or among groups of engineering objects that are contained in separate nodes. Protocol Entities are often shown as two peer entities communicating with each other over a link between connected nodes or, when using Network Layer protocols, within a network consisting of multiple nodes.

Engineering objects may implement protocols in hardware or software, and the Protocol Entities themselves may be implemented in hardware or software. Some nodes in a space data system may only have communications functions. A Network Layer router (ISO Layer 3) or Data Link Layer bridge (ISO Layer 2) are examples of nodes that typically contain only Protocol Entities (without other functional elements).

While a full ‘typical’ communications stack (application, transport, network, data link, physical) is often used in the terrestrial subdomain of a system, in many space deployments only the lower Data Link and Physical Layers may be specified, with applications providing any upper layer functions that are required. Newer space systems incorporate onboard networking and even networking among and between spacecraft using space qualified networking protocols. Separate Management protocols may be employed for complex, networked systems, and various security protocols may also be employed where deployed system integrity, confidentiality, and reliability are concerns.

9.4 CHARACTERISTICS OF COMMUNICATIONS OBJECTS

The interfaces of Communications Objects (Protocol Entities) are shown in figure 9-1.

The upper (provided) interface of a Protocol Entity to an engineering object or other Protocol Entity is called the Provided Interface, and it is typically described in the protocol specification in terms of requests, indications, responses, and confirmations received from the upper (N+1) layer. The services provided by a Protocol Entity are made available at its service interface, which is called a Service Access Point (SAP).

Protocol entities also receive services from a lower layer (N-1) of the protocol stack. The services required of the lower layer may be described in the protocol specification as the Required Interface. These are not always fully described and may be treated as an implementation detail.

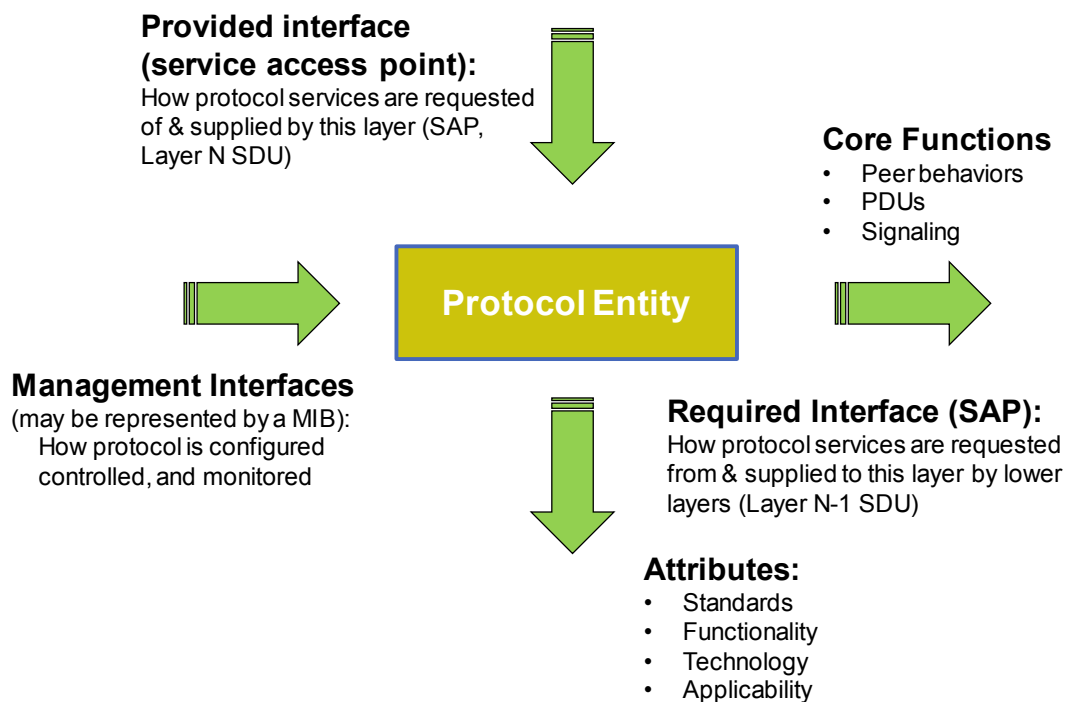


Figure 9-1: Communication Object Overview

Protocol Entities communicate with peer Protocol Entities at the same layer, either directly or indirectly, through the stack of lower-layer protocol entities. The logical interactions between peer Protocol Entities at the same ISO layer are described by exchanges of Protocol Data Units (PDUs), and the behaviors that take place within a Protocol Entity, in response to arriving PDUs, are most often described as a state machine or table of state transitions. This state machine describes the actions that the Protocol Entity is to carry out upon arrival of any of several different PDUs or other events. Activities within a Protocol Entity may also be triggered by events such as timers or by a management request from a peer or separate entity.

The management interface of a Protocol Entity may be defined by a relatively static set of configuration parameters defined in a Management Information Base (MIB), or it may be defined as some separate out-of-band interface or protocol, or by an ‘in-band’ protocol that is addressed to an interior or exterior management entity. Specialized management protocols are most often found in higher level (ISO Layer 3 and up) configurations.

9.5 KEY OBJECTS AND RELATIONSHIPS

9.5.1 GENERAL

The following elements may appear in the Communications Viewpoint:

- protocol entities (elements that implement specific protocols, with a SAP and peer interactions optionally shown, protocol stacks):
 - types: protocol purpose (e.g., coding, modulation, link, network, transport, middleware, application service);
 - attributes: name, type, capabilities (e.g., in order, once only, bandwidth efficient, error correcting, delay tolerant), applicability (e.g., deep space, near Earth, in situ), constraints, services (offered, required), interface signature (requests, indications, responses, confirmations), Application Programming Interface (API) where appropriate, standard reference identifier, standards organization;
- protocol design specification elements (PDU description, behavior as state machine or table description), reliability (acknowledged, unacknowledged, selectively acknowledged), other design views of the communications protocol or protocol stack;
- (optional) nodes and links (representations of physical elements from the Connectivity Viewpoint, for context);
- (optional) Software engineering objects (representations of implemented functions from the Connectivity Viewpoint, for context).

9.5.2 ONTOLOGY OF PROTOCOL ENTITIES

Figure 9-2 shows the ontology of Communications Viewpoint Objects. The normative specifications for protocol entities are usually specified by a standard, which is itself an Information Object. The stack of protocols that are defined at the boundary of a system element may provide a service interface that is called an Interface Binding in the Service Viewpoint. The protocol stack also defines the implementation of a communications port on a systems element.

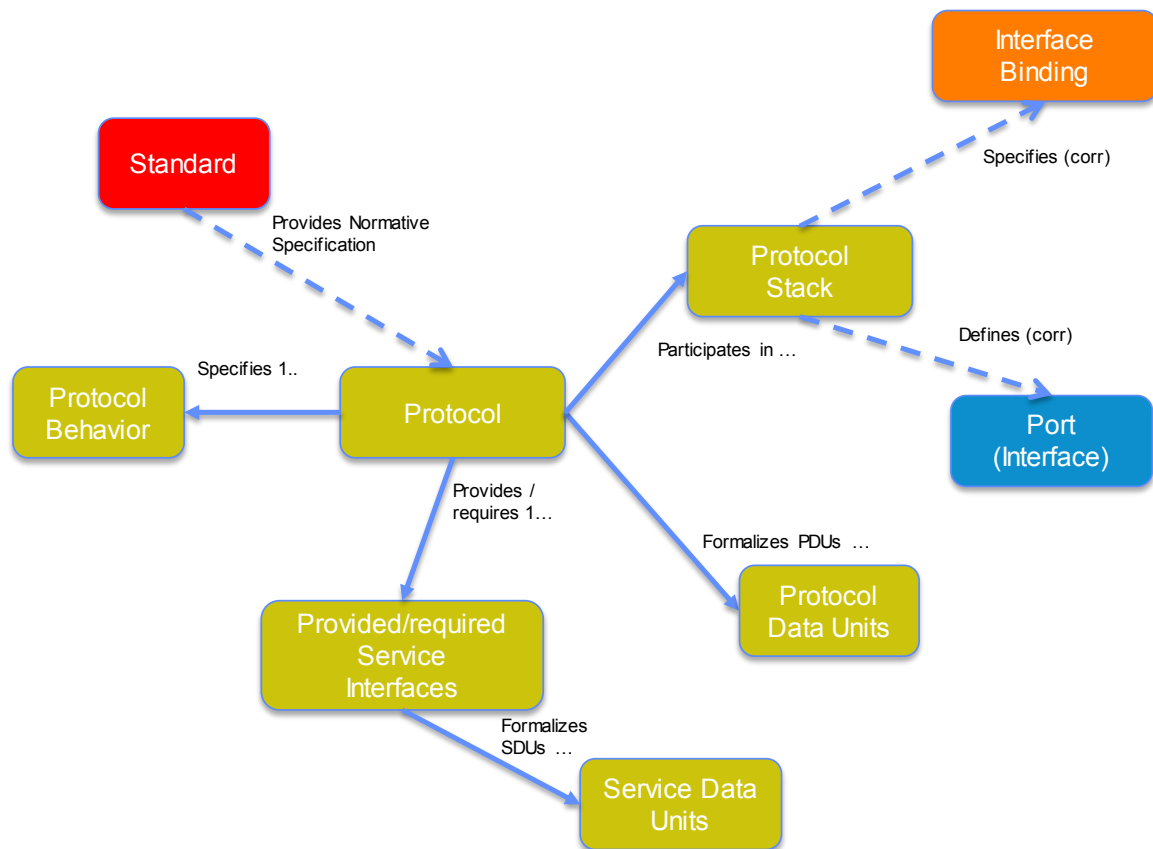


Figure 9-2: Ontology of Communications Viewpoint (Protocol) Objects

9.5.3 REPRESENTATION OF COMMUNICATION OBJECTS

Figure 9-3 shows the defined representation of Communications Objects. Two different kinds of flows are shown, the flow of PDUs between two peer protocol entities at the same layer, and the end-to-end flow down one side of the stack, across the Physical Layer, and up the other. The intended end-to-end flow may only be source to destination (e.g., send a file from the entity on the left to the one on the right) but there may actually be a bidirectional flow of PDUs between the two peer protocol entities that are used to signal, and ensure, reliable transfer. So there may be a logical flow of data, source to destination, that is really different from the actual flow of PDUs.

Two protocol entities are shown, Layer N and Layer $N-1$. Each has a SAP, the interface on top is the Service Provided Interface, offered for use by an upper layer protocol or application. This gives access to the services provided by that protocol layer and, implicitly, by the lower layers. The interface below is the Service Required Interface, and it may (optionally) describe the kinds of services required from a lower layer.

If the Required and Provided Interfaces between an adjacent pair of layers in a protocol stack do not match exactly, then some protocol interface ‘shim’ may be required. Such a shim may be a library that transparently intercepts API calls and changes the arguments passed,

manipulates the data structures, handles the operation itself, or redirects the operation elsewhere. This shim may be ‘thin’, requiring little to handle the impedance mismatch, or it may be ‘thick’ enough to be identified, and documented, as a protocol layer or adapter in its own right. If this shim is formalized in its own standard, it may be represented by a sublayer in a protocol stack diagram, but often it is just mentioned and left as an implementation detail.

While protocol PDUs logically flow ‘horizontally’ between peer level protocol entities, in actuality they flow down (and then up) the stacks on either side. At each layer the PDU for Layer N becomes part of the Service Data Unit (SDU) for Layer $N-1$.

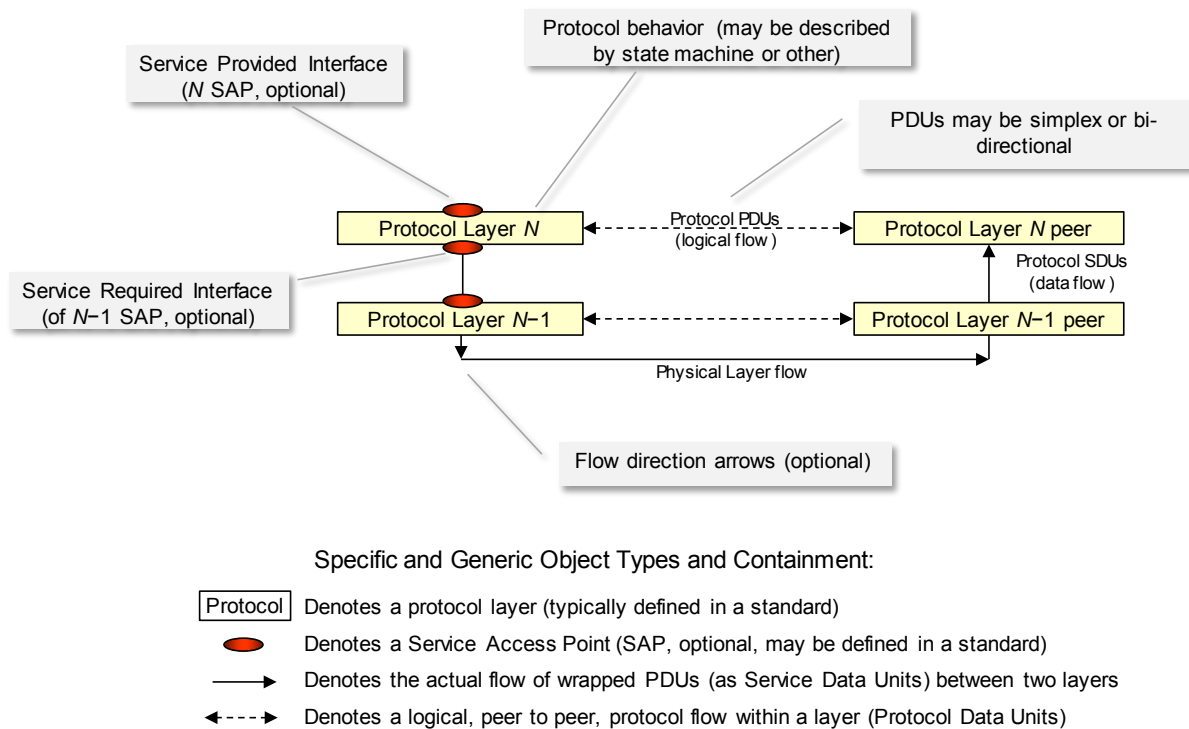


Figure 9-3: Representation of Communication Objects

Often the stack of protocols, with their successive encapsulations of SDUs, where they are shown as data inside lower-level PDUs, may be presented in a diagram that reflects the actual assemblage of bits (or octets) that appear at the lowest layer in the stack, as a Physical Layer ‘on the wire’ view. This might involve a ‘stacked’ set of diagrams as shown in figure 9-4.

9.6 TERMS FOR THE COMMUNICATIONS VIEWPOINT

9.6.1 ATTRIBUTES

Protocol Attributes are the key features of a specific protocol that describe the ISO layer at which it is intended to provide the defined protocol behavior.

A **standard** is a formal specification that defines and governs functions and protocols at interfaces of a data system. It describes in detail the technical capabilities of, and establishes the requirements to be met by, interfacing subsystems to achieve compatibility and interoperability.

Functionality is the ability of a protocol to perform its intended tasks.

Applicability of a protocol is a statement of the fact or quality of applying to a certain situation or range of situations. Many CCSDS (and other) protocols will contain an applicability clause describing where and how it is intended to be used.

The **technology** of a protocol is a description of how it is implemented. The same protocol specification might be implemented in software, firmware, or hardware, depending upon performance or other requirements, such as a requirement to support future updates.

9.6.2 OTHER TERMS FOR THE COMMUNICATIONS VIEWPOINT

Most of the definitions in this section are drawn directly from the ISO-BRM (reference [13]).

An **(*N*)-layer** is any specific layer in a multi-layer protocol stack. The layer above is called the **(*N*+1)-layer**, the layer below is called the **(*N*-1)-layer**. This notation is also used for other concepts in the model which are related to these layers, for example **(*N*)-protocol**, **(*N*+1)-service**.

At a given instant in time during the life of some object, **state** is a condition or situation that determines the set of all sequences of actions in which the object can take part.

A **state machine** is a description of the discrete sequence of states that an object or interaction goes through during its life in response to events, together with its responses and actions. A **state table** is an alternative tabular representation of the same information.

A **Protocol Entity** is an active element within an (*N*)-communications-subsystem embodying a set of capabilities defined for the layer that corresponds to a specific (*N*)-entity-type (without any extra capabilities' being used). Protocol Entities implement protocol state machine behavior.

A **PDU** is a unit of data specified in an (*N*)-protocol, consisting of (*N*)-protocol-control-information and possibly (*N*)-user-data. PDUs are the actual Data Objects that are exchanged between peer protocol entities.

A **protocol** is the set of rules and formats (semantic and syntactic) used to determine the communication behavior of (*N*)-protocol-entities in the provision of (*N*)-services. The behavior of the state machines that operate within a Protocol Entity and the PDUs that are exchanged between these entities specify a protocol.

A **SAP** is the point at which (*N*)-services are provided by an (*N*)-protocol-entity to an (*N*+1)-protocol-entity.

An **API** is a set of definitions of the ways one piece of computer software communicates with another. It is a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software.

A **protocol shim** may be used between an adjacent pair of protocol layers where the required and provided interfaces are not an exact match. It may be a piece of code that transparently intercepts API calls and changes the arguments passed, manipulates the data structures, handles the operation itself, or redirects the operation elsewhere.

9.7 TYPICAL PROTOCOL ENTITIES

Table 9-1 shows several examples of typical Protocol Entities used in space data systems. This table is representative, but does not include all of the available or applicable protocols for use in space data systems, and not all combinations of these protocols are valid. Protocols are normally associated with some layer defined in the ISO-BRM, but these layers designators are not provided here except by reference to protocol type. CCSDS Overview of Space Communication Protocols (OSCP) (reference [15]) can be consulted for more information about which combinations of protocols are recommended from Layer 4 down to the Physical Layer. The CCSDS Space Communication Cross Support Architecture Requirements Document (SCCS-ARD) (reference [18]) addresses both the currently available CCSDS standards and their appropriate uses in much more depth, as well as their recommended allocation to different components in a typical system.

Table 9-1: Typical Protocol Entities

Protocol Entities	Type	Description
Asynchronous Message Service (AMS)	Messaging	Provides message transfer services between functions.
CCSDS File Delivery Protocol (CFDP)	File transfer protocol	Transfers files over one or multiple space links.
File Transfer Protocol (FTP)	File transfer protocol	Transfers files over Internet protocols.
Audio and Video	Application Layer protocols	Provides end-to-end audio and video communications.
Transmission Control Protocol (TCP)	Transport protocol	Provides end-to-end communications in Internet.
Space Packet Protocol	Network protocol	Provides a path identifier through a set of one or more space and terrestrial links.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Protocol Entities	Type	Description
Bundle Protocol (BPv7)	Network protocol	Provides store-and-forward relay as a core element of a Delay (and disruption) Tolerant Network (DTN) involving a set of space links.
Bundle Protocol Security (BPsec)	Network Layer security	Provides mechanisms for securing the data sent end-to-end across a DTN network.
Internet Protocol	Network protocol	Provides routing through Internet
TM Space Data Link Protocol	Data link protocol	Provides communications from space to ground over a point-to-point space link.
TC Space Data Link Protocol	Data link protocol	Provides communications from ground to space over a point-to-point space link.
AOS Space Data Link Protocol	Data link protocol	Provides communications, space to ground, ground to space, or space to space, over a point-to-point space link.
Unified Space Data Link Protocol (USLP)	Data link protocol	Provides communications, space to ground, ground to space, space to space, or orbit to planet surface, over a point-to-point space link.
Space Data Link Security	Data Link Layer security	Provides mechanisms for securing the data sent across a single link.
TM Synchronization and Channel Coding	Channel coding	Provides mechanisms for data synchronization and error control.
TC Synchronization and Channel Coding	Channel coding	Provides mechanisms for data synchronization and Forward Error Correction (FEC).
Proximity-1 Space Link Protocol	Data link + physical protocol	Provides communications, orbit to planet surface, over a point-to-point space link.
WiFi and 3GPP	Data link protocol	SOIS Layer 2 'on-board' protocols, also suitable for planet LAN/WAN use.
1553 and CAN bus		SOIS Layer 1 and 2 onboard protocols.
CCSDS RF and Modulation	Physical protocol	Define physical RF frequency bands and power or Bandwidth Efficient Modulation (BWEM) to transmit and receive RF signals over a space link.
CCSDS Optical and Modulation	Physical protocol	Define physical optical frequency bands and modulation to transmit and receive optical signals over a space link.
Optical Communications Coding and Synchronization	Channel coding	Provides mechanisms for data synchronization and error control.

9.8 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH COMMUNICATIONS VIEWPOINT

9.8.1 OVERVIEW

Descriptions of Protocol entities may just focus on the features of an individual protocol layer, including the PDUs, the interfaces, and the behavior of the entity described as a state machine. Such descriptions may be used just to document a key part of a protocol stack that enables a certain kind of data transfer, such as across a space link. Or the end-to-end protocol stack may be shown, describing how sets of protocols providing capabilities, such as packet or file delivery, or delay and disruption tolerant space internetworking, are to be provided.

9.8.2 PROTOCOL STACK DIAGRAMS

A simple example of a Communications View is shown in figure 9-4, in which three sets of Protocol Entities are represented as stacks of rectangles. Each rectangle represents a specific Protocol Entity that implements services for its layer in the protocol stack. Each Protocol Entity offers services to the $N+1$ layer entity that is above it and uses the services of the $N-1$ layer that is below it. Each N -layer Entity participates in an exchange of PDUs with its peer N -layer Entity.

This figure is borrowed from the SCCS-ARD (reference [18]), which uses RASDSv2 as a framework to describe the services, and possible deployments, of more than 70 different CCSDS layer 1–7 standards.

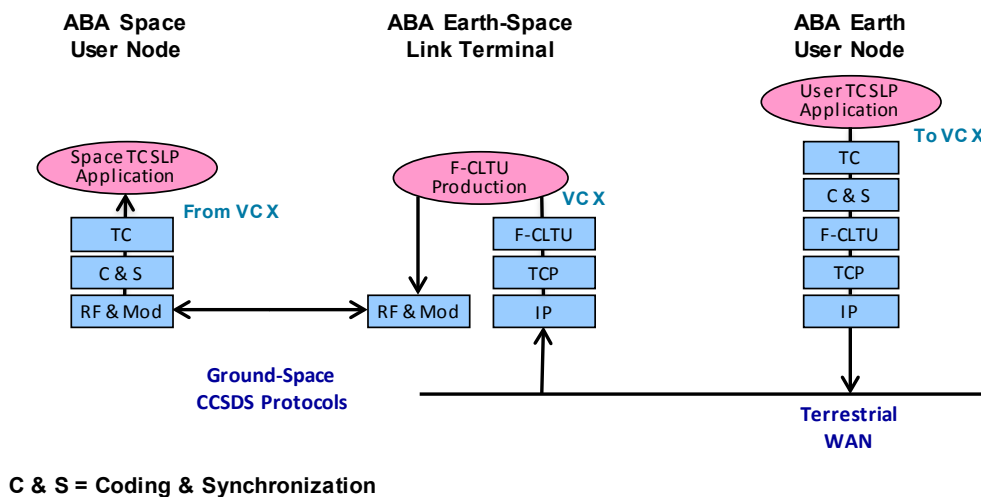


Figure 9-4: Simple Example of an End-to-End Communications View

Figure 9-4 represents the end-to-end flow of application data from a user node on Earth (on the right) to a user node in space (on the left), using a ground station (referred to as an Earth-Space Link Terminal [ESLT]) to provide the RF ground-to-space link. In this case the Telecommand (TC) protocol is used in the forward direction. And, as one might notice, the

protocol stacks on the terrestrial side and the space side of the ESLT are not fully symmetric. This is the result of the different features implemented in the ESLT by the Space Link Extension (SLE) Forward CLTU (F-CLTU) service protocol, which transfers the TC PDUs (as encoded CLTUs) over the terrestrial WAN versus how they are transferred over an RF ground-to-space link. The TC Data Link Layer protocol is carried end-to-end, but the underlying layers (and the SLE ‘tunnel’) are different between the Earth user to ESLT and the ESLT to Space User nodes.

Many of the protocols used in space exhibit these kinds of end-to-end asymmetries, which are often the result of the very constrained resources that are available on the typical spacecraft. The size, weight, and power limitations on the components on a spacecraft, compared to terrestrial resources, are always a driver on the forward and return data rates. Other drivers, such as SLE or Cross Support Transfer Service (CSTS) terrestrial services to access Telemetry, Tracking, and Command (TT&C) space links, and the kinds of services that will become necessary in remote environments such as CisLunar, will also result in asymmetric configurations.

9.8.3 END-TO-END PROTOCOL DIAGRAMS

Frequently it is necessary not just to show the end-to-end protocol stacks, but to also show them in conjunction with the physical nodes where the protocols are implemented and deployed. The physical nodes are referenced via correspondence from the Communications viewpoint.

Figure 9-5 shows an abstract view of an end-to-end protocol flow and overlays these protocol elements, by correspondence, on three nodes, two user nodes and a ground station, or ESLT. This figure is also borrowed from the SCCS-ARD (reference [18]), and it makes clearer just where these protocol stacks are to be deployed. Separate forward and return flows are often used because of the asymmetries just discussed. This figure just shows a return (spacecraft-to-Earth) flow.

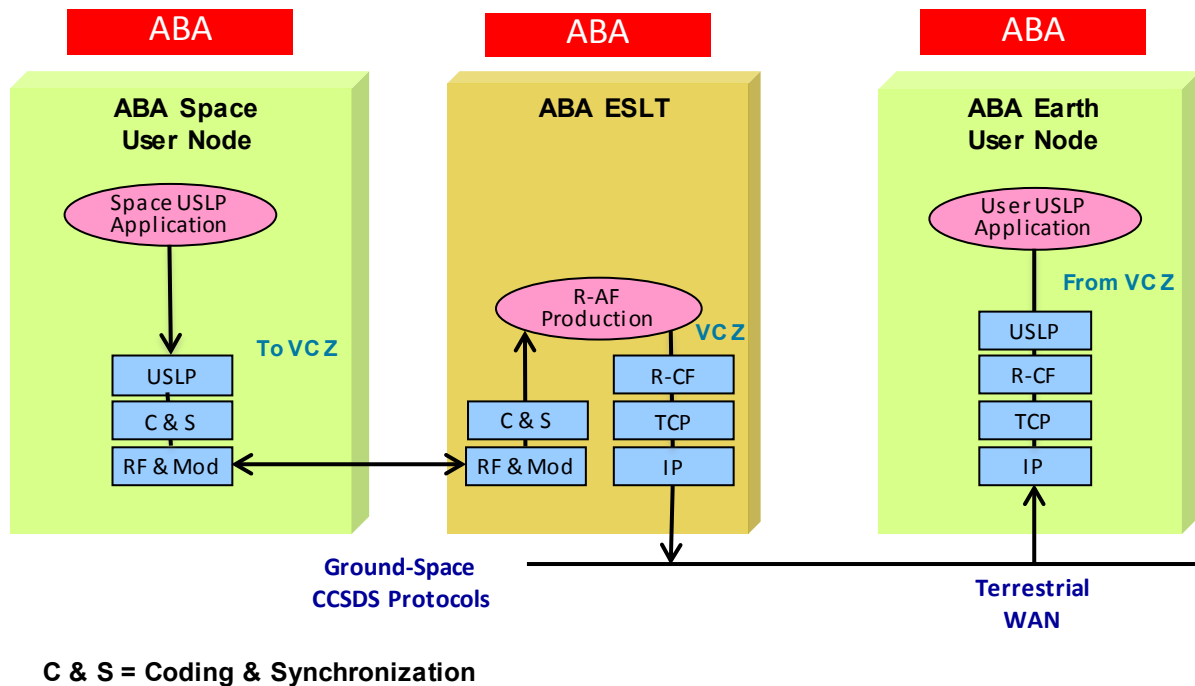


Figure 9-5: Example of a Return Communications View Showing Abstract Protocol Stack and Allocation to Nodes

Figure 9-6 shows a more complicated set of Data Link Layer and Network Layer Protocol Entities that support relayed communications between an Earth user node (on the right) and its space user node (on the far left). These two user nodes are shown communicating using the Delay Tolerant Network (DTN) protocols (reference [21]) end-to-end. This is an example of what is called a Solar System Internetwork (SSI) (reference [22]) deployment.

Intermediate relay nodes are also shown; the ESLT configured to support SSI supports both ‘standard’ Data Link Layer protocols and also hosts the features that support SSI networking. The other ‘support’ nodes for SSI end-to-end communications that are included in this view are the two space routing nodes that are placed out in a remote orbit around the Moon or Mars to relay data to the surface of the planet, and the space routing node MOC that controls these relay orbiters. End-to-end data traffic can flow through all these nodes, or, depending on how they are configured, DTN traffic may only flow through some subset of them.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

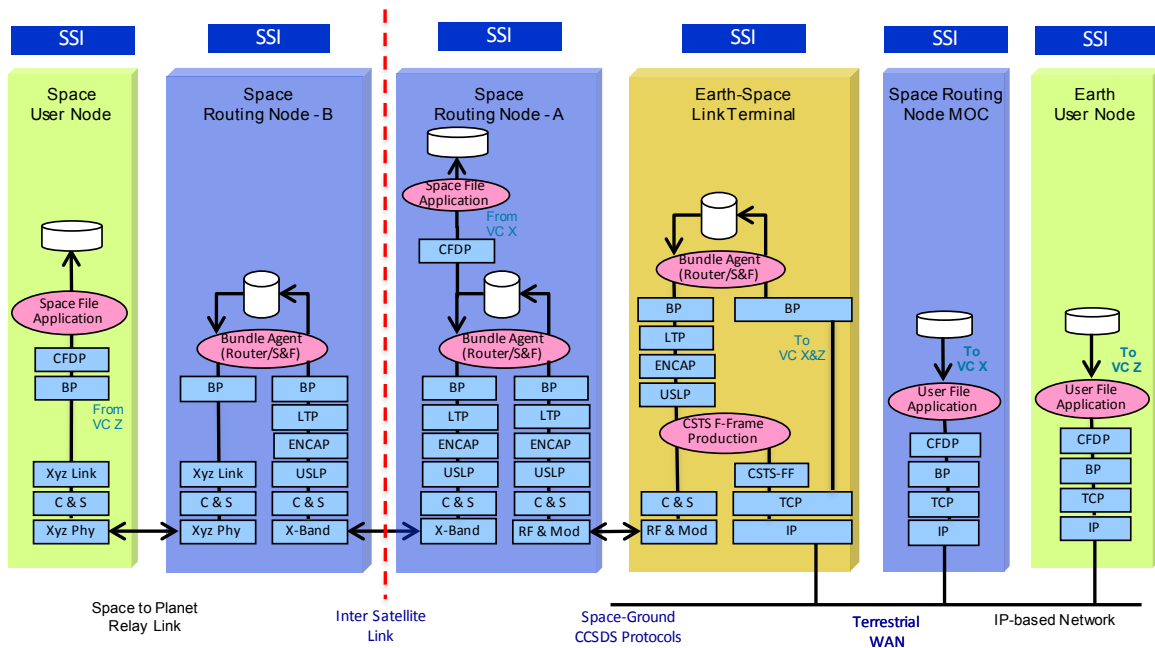


Figure 9-6: Example of SSI End-to-End Communications View Showing Nodes

9.8.4 PROTOCOL PDU AND BEHAVIOR DIAGRAMS

A Communications View may just show Protocol Entities in a ‘black box’ view, with only the SAPs and some representation of peer protocol entities indicated. But where required, more engineering details of the protocol specification may also be represented by showing the internal data flows, structure, and timing of PDUs, and even the internals of processing within the Protocol Entities.

In most CCSDS (and Internet) documents, a PDU is shown as a series of octets. Several different presentation styles may be used, and a specific representation for PDUs is not defined here. However, figure 9-7 provides a useful example from the CCSDS Space Packet Protocol (SPP) document (reference [23]). The Internet Engineering Task Force (IETF) Requests for Comment (RFCs) that document Internet protocols mandate a similar representation of PDUs, but they are expressed in an ASCII text form. However they are shown, the important aspect of this is to depict the exact data structures, ordering, sizes, and types of fields, and to identify the control and data elements in the PDU.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

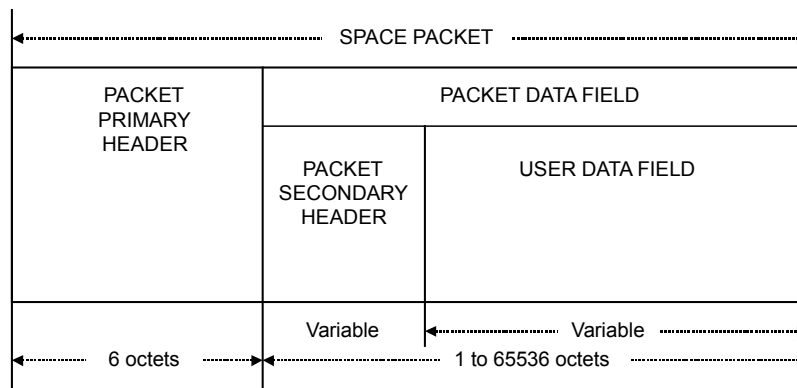


Figure 9-7: PDU Example, Space Packet Protocol

In addition to describing the sets of PDU structures, it is usually necessary to specify the set of states and transitions that describe the actions taken within a Protocol Entity when a particular type of PDU arrives. This may be shown diagrammatically using a state machine, as in figure 9-8, which is a simple one taken from the CCSDS SLE Return Channel Frames (RCF) document (reference [24]), or it may be described using a state table or even narrative text, as is done in several CCSDS and Internet documents.

RASDSv2 does not require use of any specific representation to describe Protocol Entity behavior, but a clear and concise state machine or state table specification is preferred, as these are typically more precise and more readily converted to code.

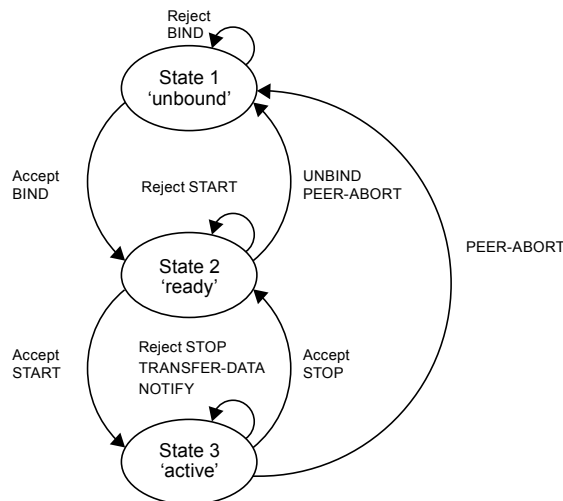


Figure 9-8: Example State Machine Diagram—SLE RCF

For interoperability, regardless of the representation used, the complete specification of a protocol must include both the PDU definitions (the data that are exchanged between peer Protocol Entities) and the behavior of the protocol state machine (action taken when a given PDU arrives).

The specification of the interface to a Protocol Entity, or SAP, may be required for application development, but agreement on a common API and language binding, while useful for portability of applications, is not essential for interoperability. Different protocol implementations, exposing different SAPs, with different APIs written in different languages, may be used in the two peer protocol entities at the same layer with no effect on interoperability as long as the protocol behavior and PDUs are correctly implemented.

These examples all use the canonical RASDSv2 drawing styles, which work well in document-based architectures. A related paper, ‘Model-based Protocol Specification’ (reference [33]) offers some more formalized approaches for protocol modeling. All these features can also be represented in SysML style drawings. A separate paper, “A Representative Application of a Layered Interface Modeling Pattern” (reference [28]), uses a pragmatic approach, and worked examples, for representing all these system and protocol features using SysML. (See annex C for more details on using MBSE approaches for this purpose.)

9.9 SECURITY TOPICS IN THE COMMUNICATIONS VIEWPOINT

Certain functions for implementing data system security may be allocated to protocols, and these will be addressed in the Communications Viewpoint. These will typically include Application Layer or Network Layer security protocols, plus authentication, access control, identity, and key management, Data Link Layer encryption and/or authentication, and they may include spread-spectrum or related Physical Layer jamming avoidance approaches.

Secure Data Link Layer and Network Layer standards for DTN and IP have been defined. [Future] New protocols for DTN network management, key management, and identity management are currently under development. The details of where and when to apply these approaches are described in the CCSDS Security Architecture (reference [4]) and Threat Assessment (reference [17]) documents, and in the Consultative Committee for International Telephony and Telegraphy (CCITT) Security Architecture for Open Systems Interconnection for applications, X.800 (reference [43]). The specific details of how to provide these capabilities are defined in the actual standards and specifications.

Figure 9-9 provides an example of an end-to-end Application and Network Layer security deployment. This SSI example is also borrowed from the SCCS-ARD (reference [18]), which uses RASDSv2 representations and viewpoints.

Security protocols are shown (as green protocol elements) being applied at two different layers in this diagram, Application (File Secure) and Network (BPsec). Application Layer security achieves either information security (hiding) or authentication (known sender) or both. The important function of ensuring knowledge of who sent some communique is addressed in the ITU Telecommunications (ITU-T) standard X.1250, *Baseline Capabilities for Enhanced Global Identity Management and Interoperability* (reference [44]). Using this technique the data in the file stream may be secured at the Application Layer, end-to-end, between the sender and the receiver. BPsec Network Layer security may also be used to provide similar information security features from end to end within the Network Layer itself.

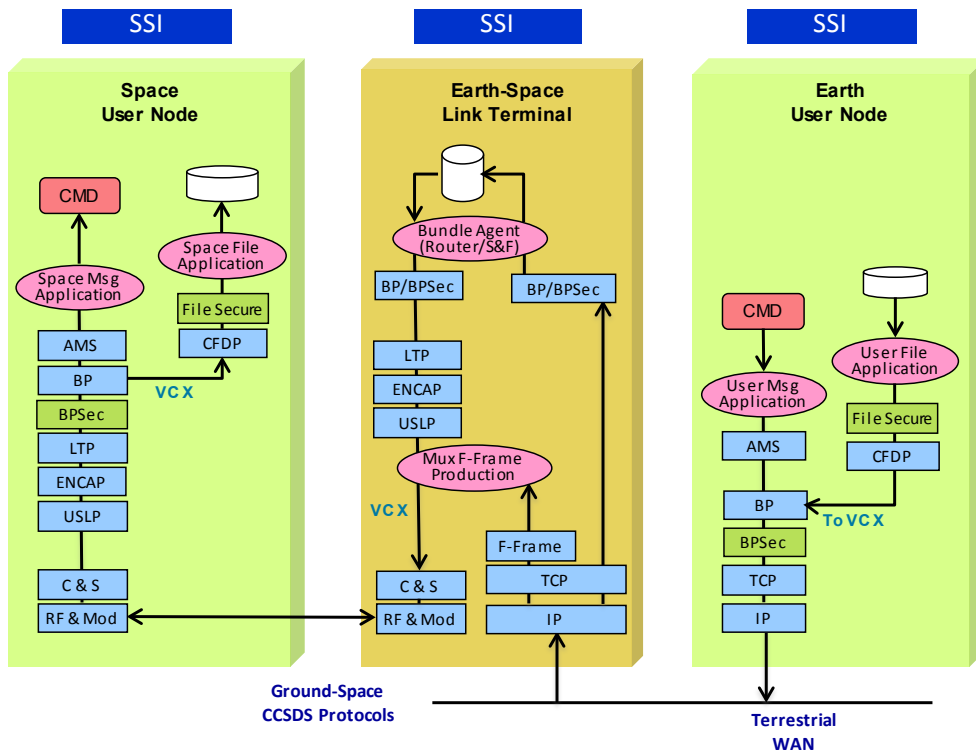


Figure 9-9: Example End-to-End Network (BPsec) and Application Layer (File Secure) Security Protocol Deployment

Examples of specific Communications Viewpoint security protocols:

- peer entity authentication;
- data origin encryption;
- data integrity;
- access control protocols (e.g., Security Assertion Markup Language [SAML], Open Authorization [OAuth], PKI);
- CCSDS Space Data Link Security (SDLS) and Key Management (KM);
- CCSDS BPsec;
- Two Factor Authentication (2FA) protocols;
- IP Security (IPsec) protocol;
- HyperText Transport Protocol Secure (HTTPS);
- secure Domain Name Service (DNS);
- Application Layer data encryption and authentication;
- secure application access;
- secure routing updates;
- secure device monitoring.

10 INFORMATION VIEWPOINT

10.1 OVERVIEW

The Information Viewpoint¹⁷ provides the detailed descriptions of the Data Objects that are passed among the elements in a system. These Data Objects may have different elements, structures, semantics, relationships, and policies. The Information Viewpoint is used to address the data architecture and data definition aspects of space systems. Representations of the Information Objects that fully defined in this viewpoint appear in other viewpoints by correspondence. They are managed (that is, stored, located, accessed, and distributed) by information infrastructure elements and are also shown as being passed among enterprise, functional, operations, and application entities. With the addition of carefully constructed relationships among Information Objects, knowledge may be represented as well, in the form of ontologies or knowledge graphs.

10.2 CONCERNS

Concerns are

- the structure and semantics of information and information management in a space system;
- the rules and constraints on information transformations and permanence in a space system;
- the relationships among Information Objects.

10.3 CONCEPTS FOR THE INFORMATION VIEWPOINT

The **Information Viewpoint** specification of a space system focuses on the information used by that system. This includes structural (syntactic) and semantic views of the information, the relationships among information elements, constraints on their use, rules for their management and transformation, and policies on access and persistence.

The Information Viewpoint looks at space systems from the perspective of the Information Objects and their relationships, separate from how they are implemented or used.

Information Objects are descriptions of data along with the necessary structure and syntax to allow interpretation and use of these objects. An Information Object may also have

¹⁷ The Information Viewpoint corresponds directly to the information viewpoint of RM-ODP, but without direct reference to the static and dynamic views of data and its transformations, which are handled in other viewpoints. This abstract view on the system is refined during implementation by developing concrete specifications that are bound to some particular language or framework.

associated metadata, and information views may define the relationships among Data Objects, rules for their use and transformation, and policies on access.

Information is any communication or representation of knowledge, such as facts, data, or opinions in any medium or form, including textual, numerical, graphic, cartographic, narrative, or audiovisual.

Knowledge is facts, information, and skills acquired by a person through experience or education; it is the theoretical or practical understanding of a subject.

Metadata is ‘data about data’, the information that describes content. It is information about the meaning of data, as well as the relationships among Data Objects, rules for their use and transformation, and policies on access.

An **Information Package** consists of a primary Information Object, with optional ancillary information, and any associated supporting information that is needed to use the Information Object. The Information Package has associated Packaging Information used to delimit and identify the primary Information Object and supporting information.

A **taxonomy** (or taxonomical classification) is a hierarchical classification or categorization system in which all the terms belong to a single hierarchical structure and have parent/child or broader/narrower relationships to other terms. Many taxonomies are hierarchies (and thus have an intrinsic tree structure), but not all are.

An **ontology** encompasses a representation, formal naming, and definitions of the categories, properties, and relations between the concepts, data, or entities that pertain to one, many, or all domains of discourse. More simply, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject.

This document uses an informal style of ontological representation to document all the objects in each viewpoint, their functions, the relationships among those in each viewpoint, and their relationships, via correspondence, to objects in other viewpoints.

10.4 CHARACTERISTICS OF INFORMATION OBJECTS

10.4.1 GENERAL

The characteristics of Information Objects¹⁸ are shown in figure 10-1. Unlike most other objects considered in this document, Information Objects are treated as static elements and are not represented with input or output interfaces. Information Objects will have some sort of schema that describes their structure, rules for use and transformation, and policies on access and permanence.¹⁹

Information Objects may have simple relationships represented in a data structure or schema, or they may be represented in a hierarchical taxonomy along with related terms, or in an ontology where the relationships among the different terms may be highly expressive and complex.

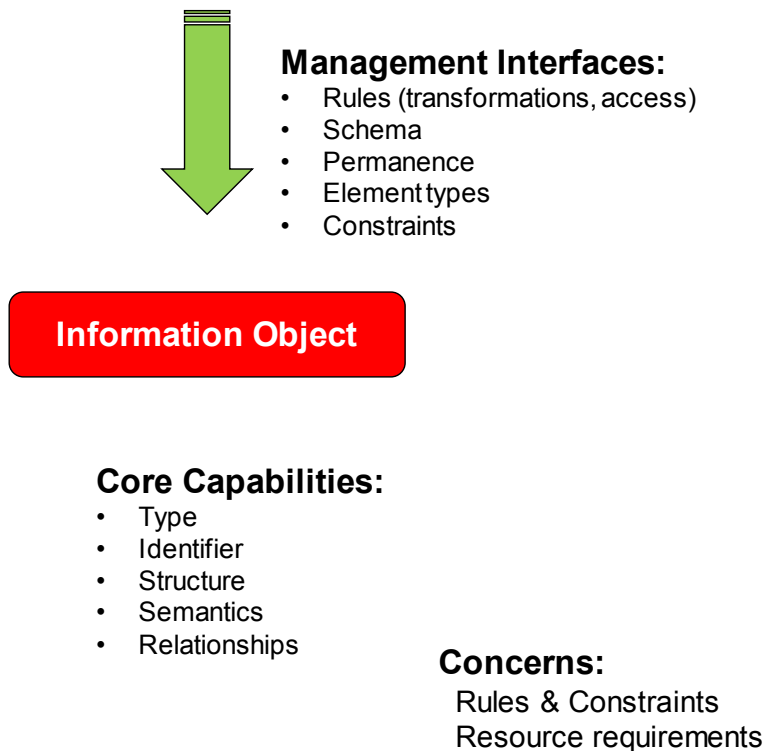


Figure 10-1: Overview of Information Objects

¹⁸ Detailed descriptions of Information Objects, and the means for managing and operating on them, may be found in a separate document, the *Reference Architecture for Space Information Management* (reference [5]).

¹⁹ RM-ODP describes static, dynamic, and snapshot aspects in their Information Viewpoint. In RASDSv2 only the static view of information structures and descriptions is treated. Any dynamic aspects of information transformation are to be handled by the corresponding representations of Information Objects that appear in the Functional and Connectivity Viewpoints.

10.4.2 KEY OBJECTS AND RELATIONSHIPS

The following elements may appear in the Information Viewpoint:

- Information Objects (abstract definitions of information elements, structures, semantics, schema):
 - types: data, metadata, information, package, schema, model, metamodel;
 - attributes: name, type, length, structure, syntax, semantics, permanence, provenance, realized by, rules, policies;
- relationships (Information Object aggregates, transformations);
- constraints (type checking rules, permanence, policies).

10.4.3 ONTOLOGY OF INFORMATION OBJECTS

The ontology of Information Objects is shown in figure 10-2. Functions produce and consume information. Data artifacts are realizations of information within deployed systems.

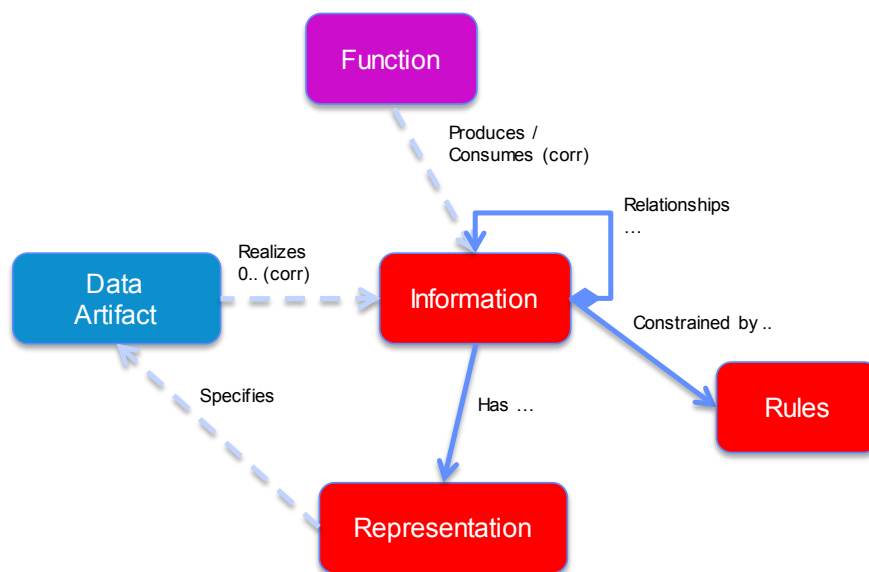


Figure 10-2: Information Object Ontology

10.4.4 REPRESENTATION OF INFORMATION OBJECTS

The representation of Information Objects recommended in this document is shown in figure 10-3. The representational style and relationships are based on UML (reference [6]) notation.

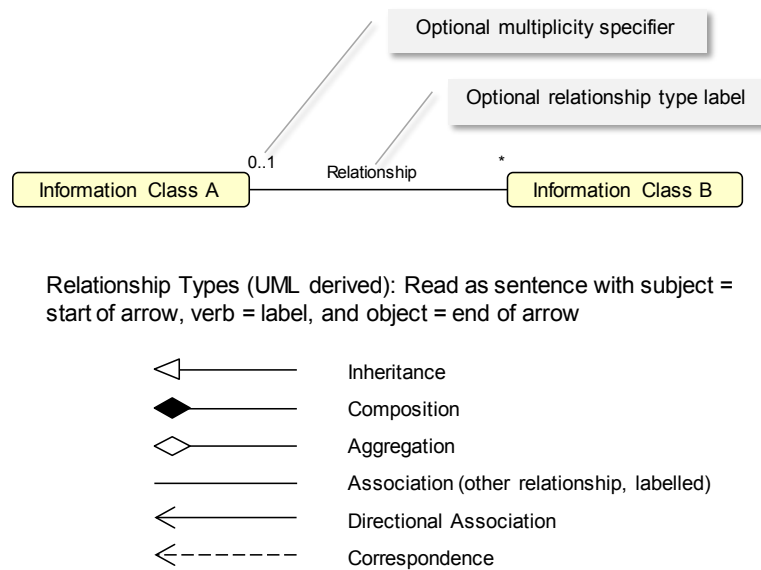


Figure 10-3: Representation of Information Objects

10.5 TERMS FOR THE INFORMATION VIEWPOINT

10.5.1 ATTRIBUTES

Data is a representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means.

Data element is a basic unit of information that has a unique meaning and subcategories (data items) of distinct value. Examples of data elements include gender, race, and geographic location.

A **data structure** is a data organization, management, and storage format that is usually chosen for efficient access to data.

Data semantics is information that defines the meaning rather than the physical representation of data. Semantics potentially cover a very large domain, from the simple domain, such as the units of one data entity, to a more complex one, such as the relationship between a data entity and another.

Relationship is the way in which two or more entities can be associated with one another.

10.5.2 OTHER TERMS

Artifact is any tangible thing made, modified, or used by people, or produced during system design, development, testing, or operations.

Data Objects are the basic Information Objects, either physical or digital.

A **data model** is the schema and structure definitions of information in a system.

A **metamodel** is an explicit model of the constructs and rules needed to build specific models within a domain of interest.

Abstract data architecture metamodels are models for specification and standardization of data elements (e.g., ISO/IEC 11179, DEDSL).

Data architecture is a model of the structure and relationships among the data elements used within a system.

A **knowledge model** is a representation of knowledge in a form that can be interpreted by both humans and machines and is used in knowledge-based systems.

A **schema** is an information model defined in a document or a database. The universe of objects that can be described is defined in the schema. For each object class, the schema defines what attributes an instance of the class must have, what additional attributes it may have, and what object class can be a parent of the current object base.

Instantiation is the creation of an instance of some abstract element, achieved by an action of an object in the model. The element can be anything that can be instantiated, in particular objects and interfaces. Data models must be instantiated as real Information Objects to participate in system activities.

Realization is the act or the condition of becoming real. Abstract data architecture elements must be **realized** as data models and stored in some sort of repository.

The **provenance** of an Information Object documents its place of origin, proof of authenticity, or record of previous processing. These are valuable pieces of information in the history of an object.

10.6 INFORMATION OBJECT TRANSFORMATIONS

The Information Viewpoint includes descriptions of Information Objects (their structure and syntax), information about the meaning and use of these objects (contents and semantics), the relationships among Information Objects (the data model), rules that define constraints on their use, transformation, and retention, and policies on access.

The basic semantics of Information Objects are shown in the figure 10-4.

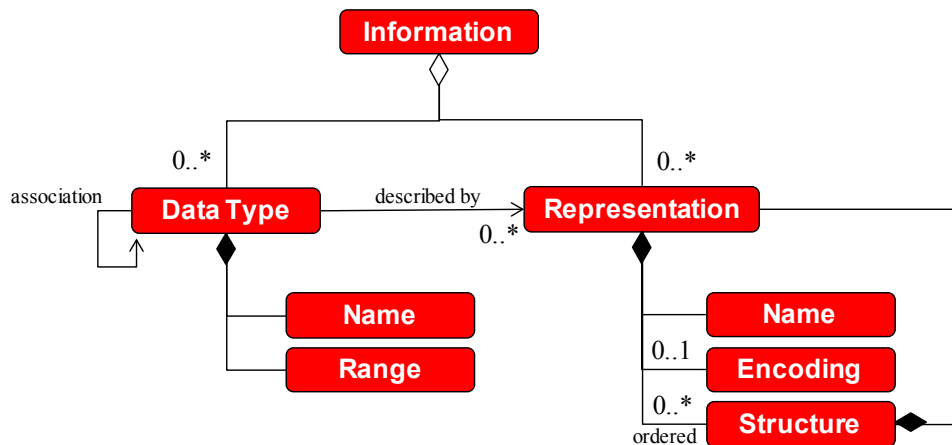


Figure 10-4: Example of Information View Showing the Basic Object Semantics

Information Objects may be represented in a systems architecture in several different ways, ranging from very abstract views to quite concrete ones. The Information Viewpoint primarily addresses the abstract specifications of data and provides a language for describing data transformations from the abstract to the concrete. The relationships among these different views are shown in figure 10-5.

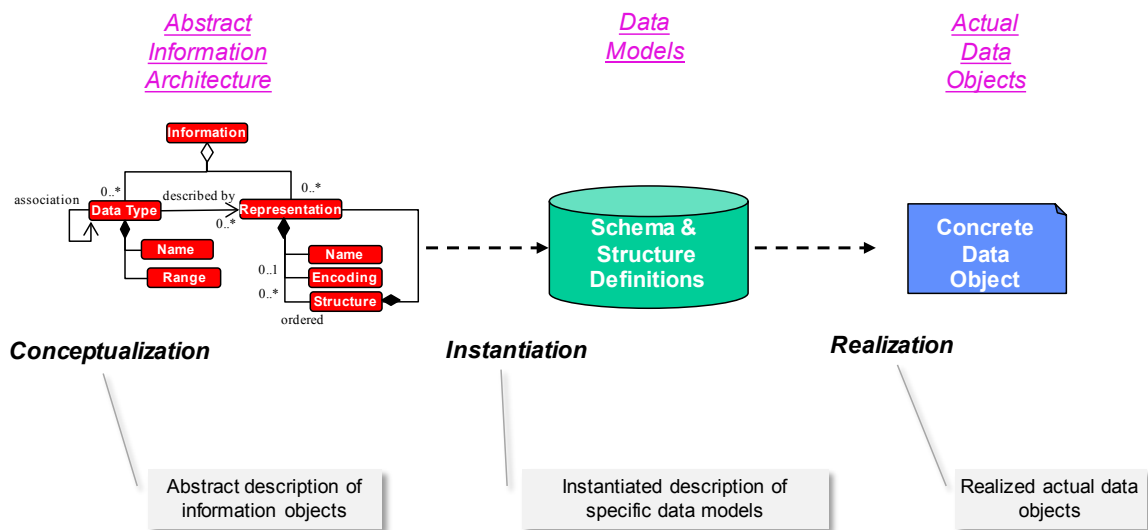


Figure 10-5: Information Object Transformations

These views may include the data element definitions, the data schema, which specifies the set of data types and order contained within the object, and the relationships among different Information Objects that are defined within the system. There will also be more concrete representations of Information Objects as they are implemented within the system. These are shown in RASDSv2 as correspondences in other views, such as the Enterprise or Functional Views.

The most concrete representations of Information Objects are the actual Data Objects, or the sets of bits or bytes of data, that are used to store information in memory or to exchange it across a communications link. If an Information Object is ‘self descriptive’, it may contain within it both the semantic content and a description of the syntax.

Often a separate description of an Information Object may be required to interpret it (although there are also self-describing Information Objects). This data model or metadata may be in the form of structure definitions within a program, schema definitions in a database or external document, or metadata stored in some other form.

A further level of abstraction that may be part of the Information Viewpoint is the data architecture, a design artifact that describes all of the different data elements and their relationships. This may be stored in a machine-accessible format, or it may be defined in a document.

In a more generalized way, relationships among Information Objects may also be defined with an ontology, which describes in more detail the relationships among a broad set of Information Objects, that is, is related to, is part of, or is used by. Increasingly, formal information description mechanisms, such as an ontology, are being used to permit machine access to all these levels of abstraction. Ontological representations, created using the rules from this viewpoint, are used throughout this document to describe the objects modeled in each viewpoint.

The Information Viewpoint is primarily concerned with the abstract data architecture representation of information within a system. Representations of this abstract data architecture, in the form of instantiated schema and data models, and representations of concrete Data Objects, developed as the system is engineered, may appear in Functional or Connectivity Views. Other representations of abstract Data Objects may appear in the Enterprise and Functional Views, and actual concrete Data Objects appear in other engineering views as the system enters detailed design.

10.7 EXAMPLE OF OBJECTS FOR THE INFORMATION VIEWPOINT

Figure 10-6 shows the relationships among some typical space system Functional Objects and the information that they exchange. This example shows a mission planning flow, where the green objects are the Functional Objects and the blue ‘narrow rounded rectangle’ objects are the actual (fully realized) Information Objects they exchange. This is a Functional View on a system showing representations of Information Objects, which would themselves be fully described in an Information View. Another way to think of this is that the structure and meaning of the data are defined in an Information View, but how these data are used and transformed is represented in a Functional View.

The Reference Architecture for Space Information Management (RASIM) (reference [5]) has a much more complete treatment of the definition and use of Information Objects. The interested reader is directed to that document. It also provides a much more complete description of the information management Functional Objects introduced in 5.8.

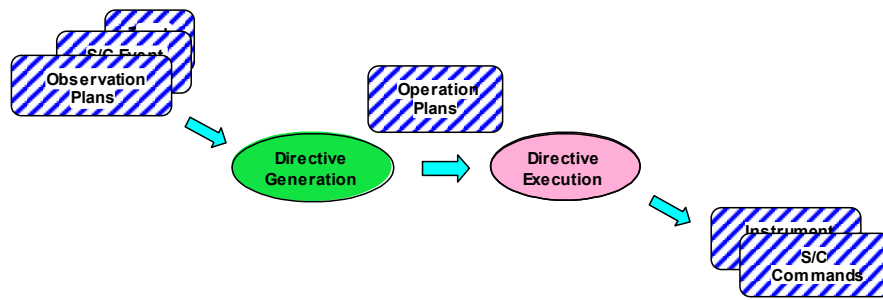


Figure 10-6: Example of Functional View with Representation of Information Objects

Figure 10-7 shows an example of an Information Object showing the use of various relationships. The relationships are represented and labelled for clarity. This figure is borrowed from the ASL document (reference [27]).

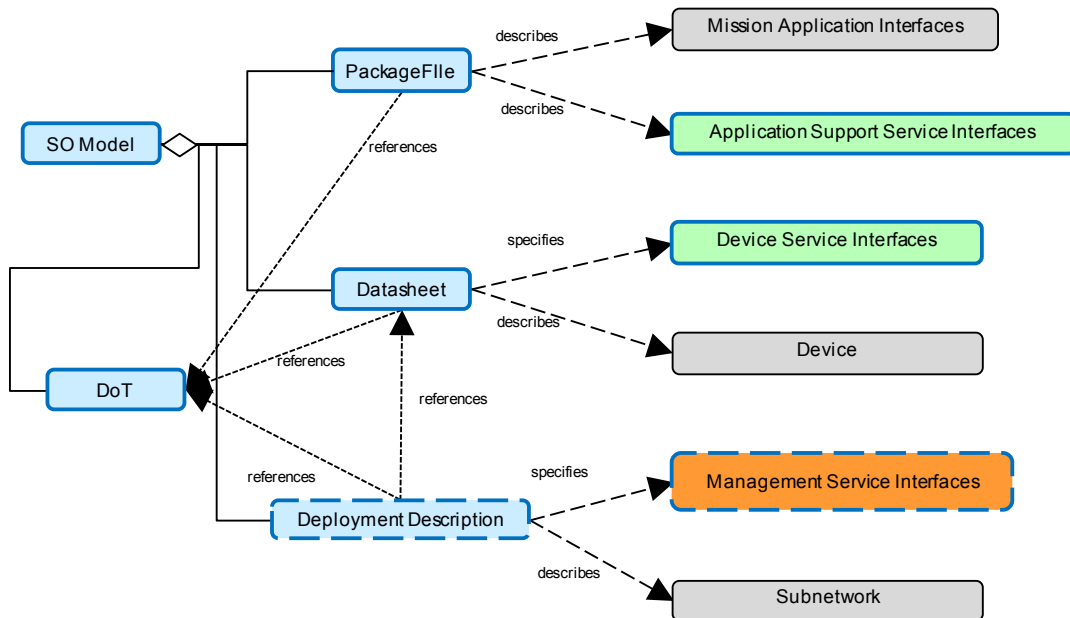


Figure 10-7: Example Information Model—Spacecraft Onboard Services

Figure 10-8 shows the Information model for the Spacecraft Onboard Dictionary of Terms (DoT) (reference [39]). This model is rendered as an ontology in the Ontology Web Language (OWL) (reference [40]) using a tool called Protégé (reference [41]), but other tools can also create and manipulate OWL and related forms. This looks a little like a taxonomy, with a hierarchical set of definitions, but the top-level elements in the ontology are classes and subclasses, and the elements may be typed by their domain and range. Within the information modeling community, the term ‘domain’ is specifically defined as the class to which the subject defining a given property belongs (the set of all possible values), and the range is the set of the values that are obtained in the form of the answers to the relation.

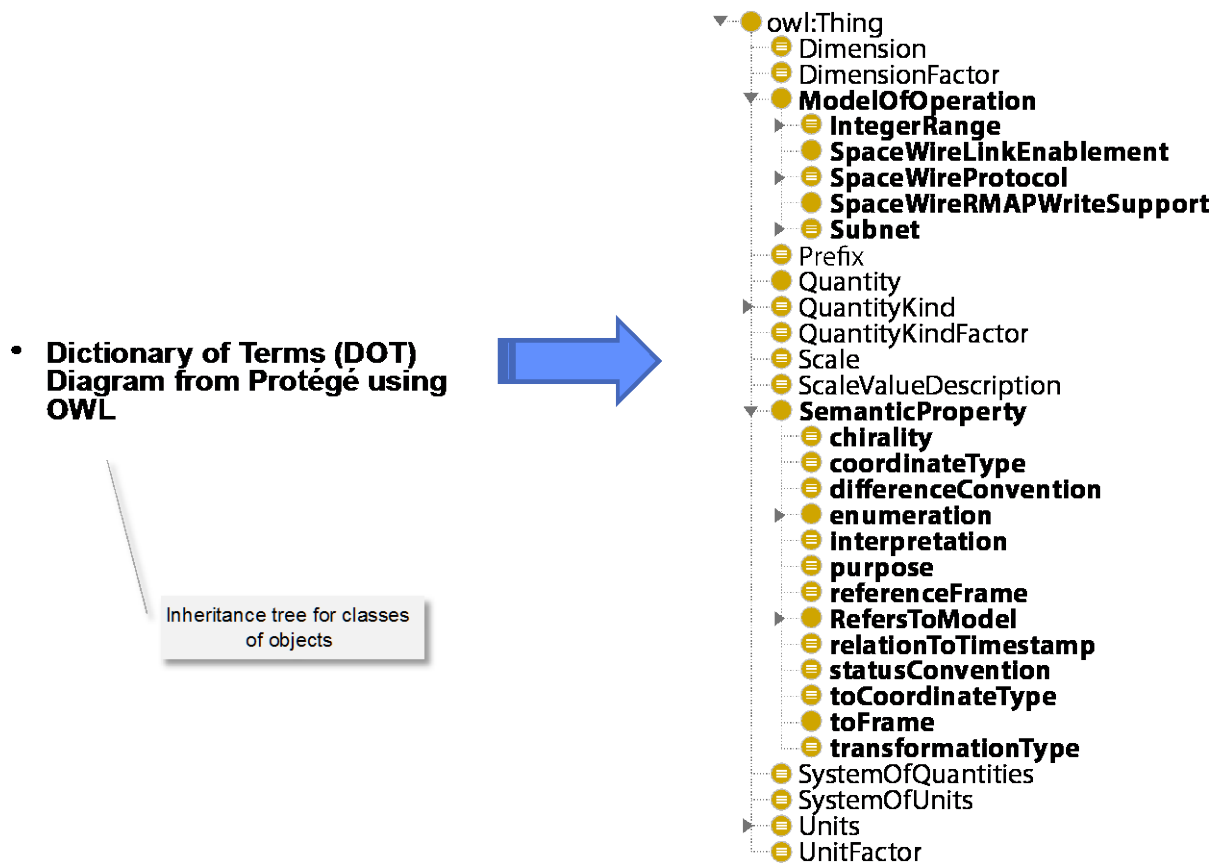


Figure 10-8: Information Object Formal Models

Formal models of information, and even more expressive knowledge models, are increasingly being used in many space information systems.

10.8 SECURITY TOPICS IN THE INFORMATION VIEWPOINT

The techniques for ensuring confidentiality and integrity are very often applied to Information Objects as part of managing system security. These techniques may be applied to data at rest or to data in motion. Data may be encrypted, or digitally signed, or both when it is at rest. And the same kinds of techniques may be applied, at different protocol stack or system layers, as data is manipulated or transported.

Examples of specific Information Viewpoint security elements:

- security keys;
- passwords;
- risk models;
- threat models;

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

- user device database;
- identity management/vetted identities/ICAM registry;
- application database;
- network device database;
- user certificates;
- encrypted data at rest/in transit.

11 SERVICE VIEWPOINT

11.1 OVERVIEW

The Service Viewpoint is used to describe the exposed services offered by a space system, as formalized by functions, interfaces, and exchanged Data Objects. It describes the service interfaces of systems entities (hardware, software, people, and/or procedures), how to request and provide services, the operations that the services provide, and the interface bindings used to access them.

The Service Viewpoint is a composite in that the technical details of service protocols, exchanged data, and the systems elements that offer services are all rendered by using the defined representations for each of those object classes. Furthermore, enterprise services will often appear in the context of descriptions of enterprise capabilities (see figures 4-7 and 4-8).

11.2 CONCERNS

The Service Viewpoint addresses the following stakeholder concerns.

Concerns

- the services provided by the system (H/W, SW, people);
- the interfaces, access points, and protocols for the system;
- the requirements and constraints on the services;
- the required and provided data for the services;
- any contractual arrangements or agreements for services; and
- how the system provides for confidentiality, integrity, and availability.

11.3 CONCEPTS FOR THE SERVICES VIEWPOINT

11.3.1 OVERVIEW

Consistent with other viewpoints, the Services Viewpoint is based on Service Objects with stated formal relationships among terms. Characteristics of the Services Viewpoint may vary in detail from architecture level to higher or lower layers of space system decomposition as they are realized in Service Views. Service descriptions are themselves dependent upon Functional, Communication, and Information Views.

11.3.2 SERVICE PROPERTIES

A service has four properties according to one of many definitions of Service Oriented Architecture (SOA).²⁰

- It logically represents a business activity with a specified outcome.
- It is self-contained.
- It is a black box for its consumers, meaning the consumer does not have to be aware of the service’s inner workings.
- It may consist of other underlying services.

The RASDSv2 definition of ‘service’ is consistent with these definitions.

11.4 CHARACTERISTICS OF SERVICE OBJECTS

11.4.1 GENERAL

types: service types, both formal and informal (systems, people, software, etc.), data types, and resources (access to elements having service roles);

attributes: service name, type, operations, point of contact, interfaces, interface binding signature, and data, interaction modes, policies, constraints;

Domains (boundaries of responsibility or ownership);

Relationships (ownership, membership, participation, roles, contractual);

Information (defined instances of services, functions, interface binding signature, and data specifications, along with documents, agreements, contracts, policies, requirements, objectives, goals, scenarios, membership lists, where formal specifications of all the data to be exchanged are found in Information Views).

In the context of figure 11-1, the interfaces of Service Objects may involve an ‘app’ (thick or thin client) or expose a formalized service protocol, they may have their own user interface (UI), or they may be deployed in the cloud and be accessed using HTTP with REpresentational State Transfer (REST) and a Web browser. Service Objects may also be implemented by people, and accessed via phone, text, or Internet requests, or even, in increasingly rare cases, in person.

²⁰ Definitions taken from https://en.wikipedia.org/wiki/Service-oriented_architecture.

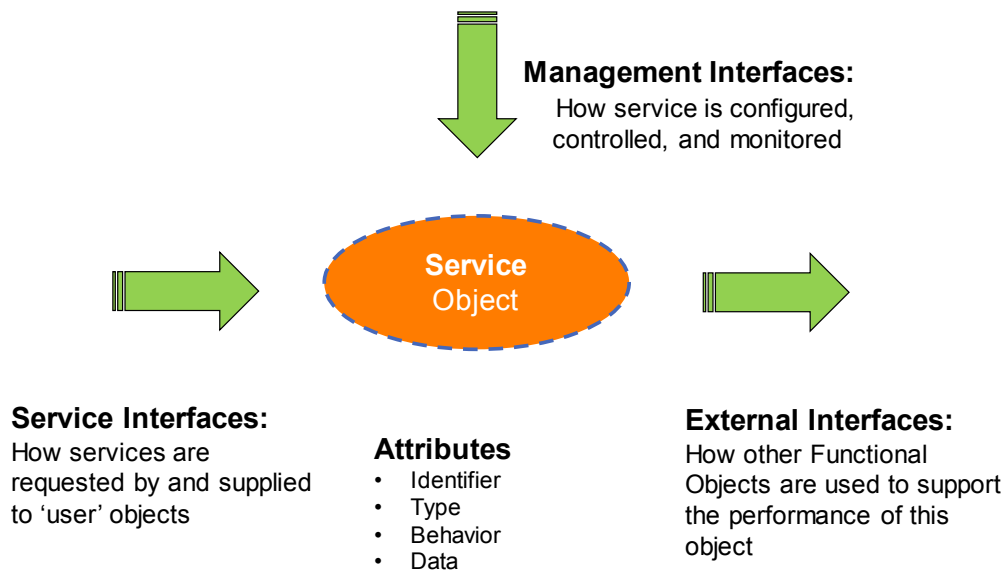


Figure 11-1: Service Viewpoint Overview

The external interfaces of Service Objects may be similarly varied. The Service Object can have interfaces such as message bus, Web, cloud, or AMS service types. The ISO-BRM treats all of this as 'Application Layer'. Server, Web/cloud, or virtualized deployments all fit this same pattern.

11.4.2 KEY OBJECTS AND RELATIONSHIPS

11.4.2.1 Ontology of Service Objects

Figure 11-2 shows the ontology of Service Objects.

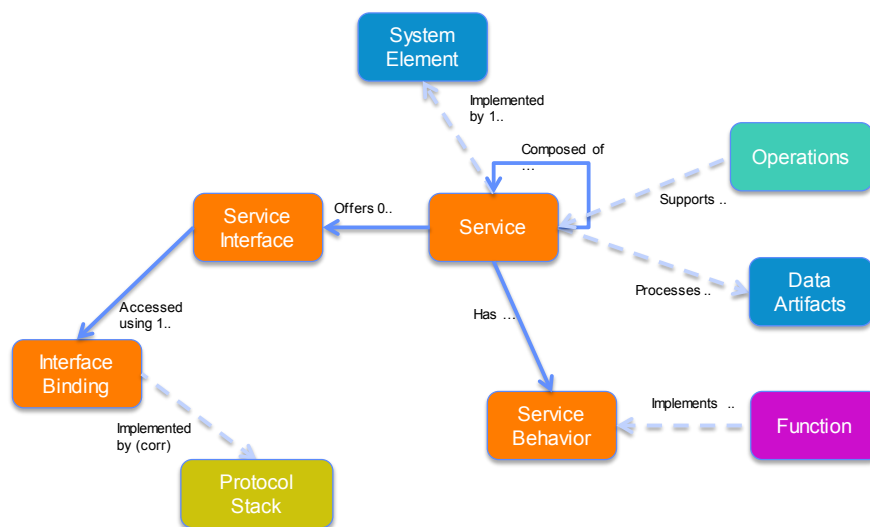
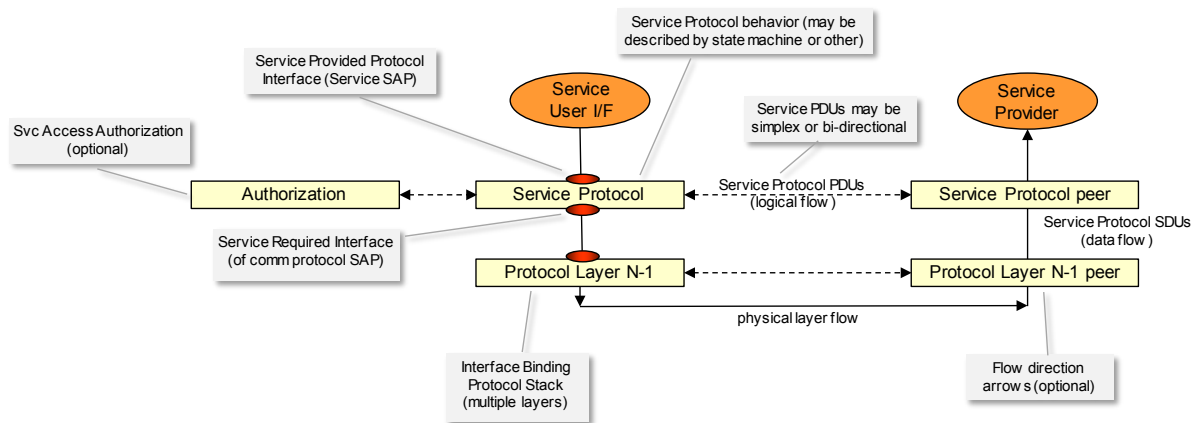


Figure 11-2: Ontology of Service Viewpoint

In figure 11-2 the interaction of Service and System Objects is formalized. A system is a collection of interacting components organized to accomplish a specific function or set of functions. A system is composed of elements, which may be any of: hardware, software, person, and procedures.

11.4.2.2 Representation of Service Objects



Specific and Generic Object Types and Containment:






-  Function that offers a service
-  Denotes a (service or optional authorization) protocol layer (typically defined in a standard)
-  Denotes a Service Access Point (SAP, required, usually defined in a standard or ICD)
-  Denotes the actual flow of wrapped PDUs (as Service Data Units) between two layers
-  Denotes a logical, peer to peer, protocol flow within a layer (Protocol Data Units)

Figure 11-3: Service Interface Representation

The service protocol, which implements the defined, exposed interface, must have defined behavior, PDUs, and Required/Provided Interfaces. Behavior may be defined in a state machine, state table, or narrative form. The specification usually only defines behavior of the sending side but may define both ends. A service protocol may be layered upon a standard, Layer $N-1$, protocol such as HTTP/REST, or use a message protocol like AMS, or be accessed via a defined API with some bespoke protocol. The protocol stacks underlying the service protocol follow normal Communication View rules.

Services may use externally supplied authentication mechanisms. They may also rely upon security mechanisms (encryption, authentication of data payloads) implemented within the service layer or in underlying layers.

11.5 TERMS FOR SERVICE VIEWPOINT

11.5.1 ATTRIBUTES

Service type may be any one of: network service, Data Link Layer service, cross support service, mission operations service, Web service, name service, or any of many other types of defined services.

Behavior of a service is a generic term for the kinds of behavior that a specific service exhibits. There are many different kinds of behaviors.

Service data is a generic term for the kinds of data provided by a specific service. There are many different kinds of service data, which may be a discrete Data Object, a stream of Data Objects (that could be turned into audio or video), or other forms such as a file or a message.

11.5.2 OTHER TERMS

Service is the provision of an interface of an object to support actions of another object.

A **service system** is the set of hardware and software components used to implement a service in a real system. Service systems may be implemented using one or more hardware and software components.

A **cross support service** is a function provided by one space agency to support operations of a space mission of another space agency.

A **service provider** is the role played by a physical, functional, or organizational entity that provides a cross support (or other) service for a service user.

A **service user** is the role played by a physical, functional, or organizational entity that uses a cross support (or other) service provided by a service provider.

A **Web service** is a software component or system designed to support interoperable machine- or application-oriented interaction over a network. A Web service has an interface described in a machine-processable format, specifically Web Services Description Language (WSDL).

An **SLE service** is the set of services that extend one of the CCSDS Space Link Subnetwork services, providing access to the ground termination of that service from a remote ground-based system. An SLE service supplies or consumes one or more channels of the same space data channel type.

Mission operations service is a suite of end-to-end application-level services that constitute an SOA for space mission operations.

11.6 TYPICAL SERVICE TYPES

Table 11-1 shows examples of typical service types that appear in space data systems.

Table 11-1: Examples of Typical Service Entities

Service Entities	Type	Description
Cross Support ground station	SLE Service	A family of Data Link Layer cross support services delivering different kinds of spacecraft data between ground station and user.
Cross Support ground station	Cross Support Transfer Service (CSTS)	A family of cross support services delivering different kinds of spacecraft data between ground station and user.
Mission Operations Center	Mission Operations Service (MOS)	Provides a framework for mission operations services.
Mission Operations Center	Message Transfer Service (MTS)	Provides topic driven message transfer services among distributed elements in space and ground.
Mission Operations Center	File delivery service	Provides file transfer services among distributed elements in space and on the ground.
Cross Support ground station	Virtual Channel Service	Provides delivery services of Data Link Layer virtual channel data between distributed elements in space and on the ground.
Space relay service provider	Delay Tolerant Network (DTN) Service	Provides delivery services of DTN network data (bundles) from distributed elements in space to the ground.
Network service provider	Domain Name Service (DNS)	Provides a service that maps network entity names to Internet addresses.
Identity service provider	Identity Service	Provides the ability to assign and verify the identity of an entity using a secure token or other means.
Audio service provider (might be Mission Operations Center)	Audio service	Provides the ability to deliver audio data in a stream form (where RTLTL permits) or as an audio recording.
SaaS provider (cloud services)	Software as a Service (SaaS)	Allows users to connect to and use cloud-based apps over the Internet. Common examples are email, calendaring, file storage, and office tools.
Communication Service Providers	Provide end-to-end data delivery (and networking) services	Commercial entities that offer end-to-end (from user MOC to user spacecraft) communications services, at link or Network Layer.

11.7 EXAMPLES OF SPACE SYSTEMS DESCRIBED WITH SERVICES VIEWS

11.7.1 GENERAL

Figure 11-4 provides a simple, abstract service protocol borrowed from the Monitor Data—Cross Support Transfer Service (MD-CSTS) (reference [25]). Standard network and transport protocols are used to provide connectivity, and the details are abstracted away. The focus in this diagram is on the layering within the service layer, which involves a user/provider interface pair and uses a separately defined CSTS message layer. The end-to-end service is between the service provider and the service user.

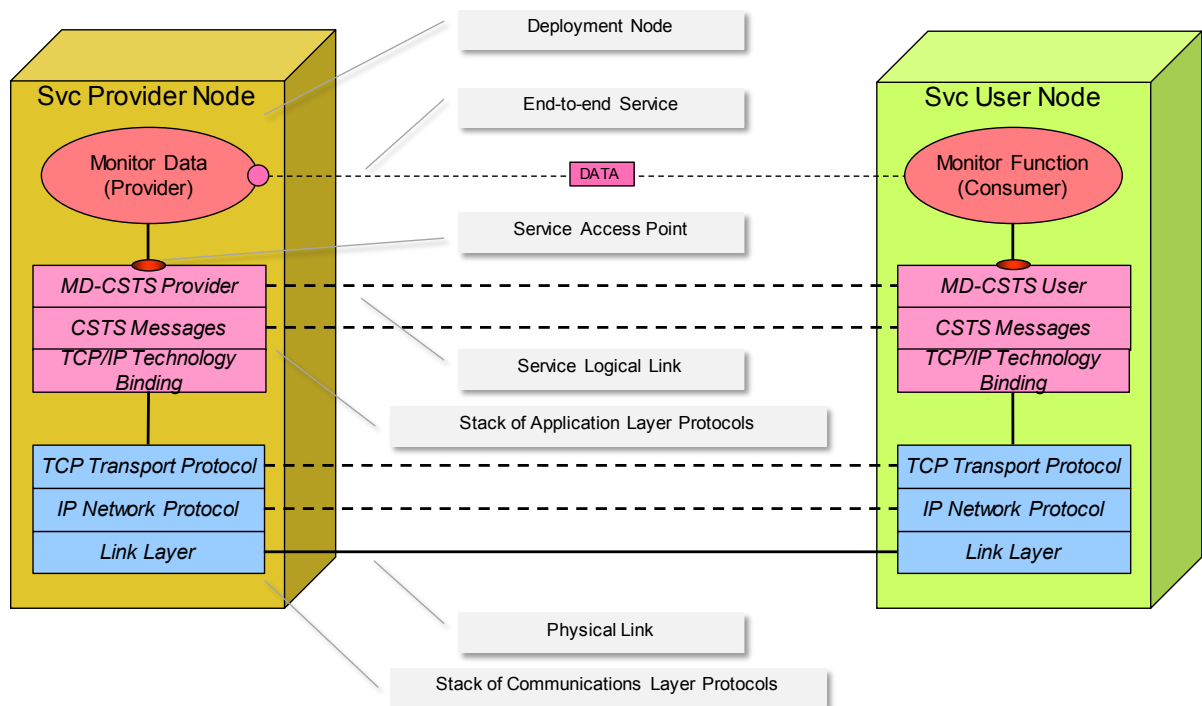


Figure 11-4: Abstract Example of a Cross Support Service: Monitor Data

This service protocol example is shown as being layered directly on TCP/IP via a technology binding. Other kinds of bindings could be shown, such as Web-based services using HTTPS, or even cloud-based deployments using some vendor’s cloud service interfaces. Regardless of the specific deployment patterns chosen, at bottom there will be kinds of layered protocol stacks between the user and the provider, wherever, and however, those components are deployed.

Figure 11-5 shows an example of a Service View that incorporates all four major aspects of a service as defined within RASDSv2.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

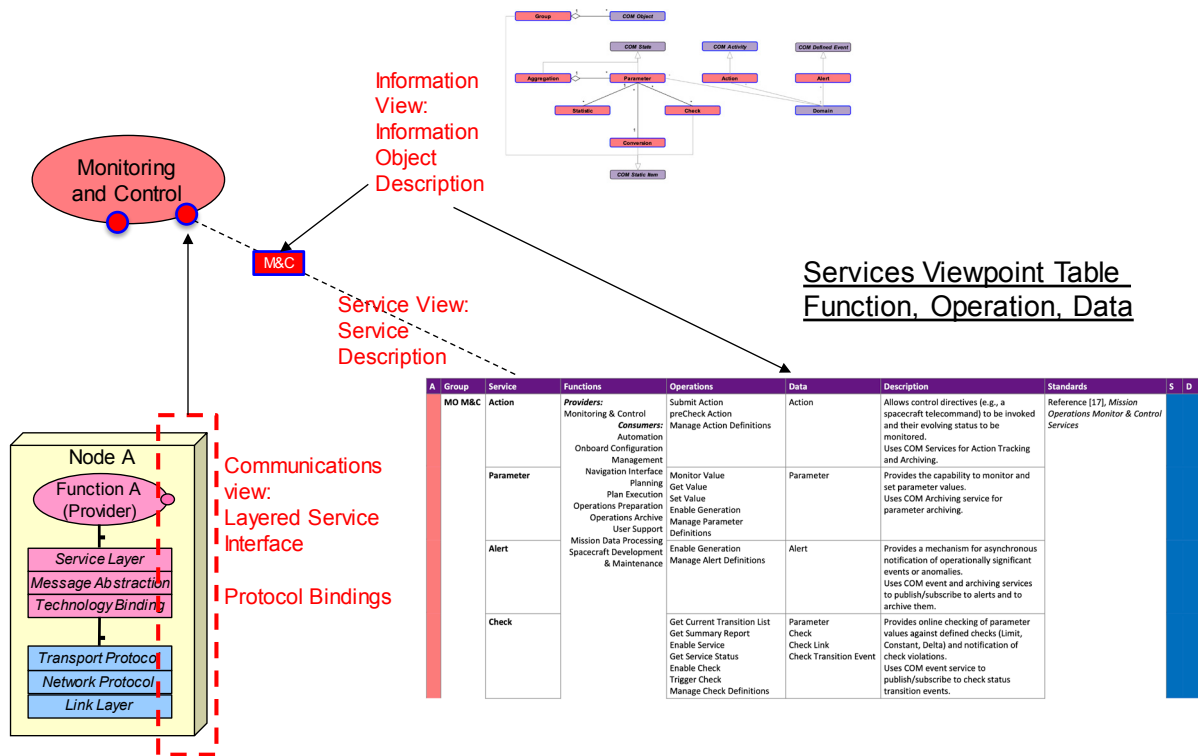


Figure 11-5: Aspects of ASL-Style Service: Mission Control

The four elements of a full service description, as shown in figure 11-5, are:

- named service and interface: Monitor and Control;
- the Service View description, showing service names, functions, operations, and the kind of data handled;
- Service interface binding, describing the deployment mode and pointers to the protocol stack defined at the interface, including the pointers to the specifics of the service protocol;
- the Information Objects that are exchanged, with pointers to the information view where they are defined.

All services essentially follow this same pattern; what tends to differ are the characteristics of how and where the various components are deployed, the protocols that are used, and the granularity of the offered services.

11.7.2 RELATIONSHIP OF SERVICES TO ENTERPRISE AND OPERATIONS

Enterprises may work with other enterprises through provision of services. While distinct from service descriptions such as Service Oriented Architecture Modeling Language (SOAML) (reference [26]), Service Objects and the formalized relationships among terms

are still fundamental to Service Views in enterprise architecture. Functional, physical, and operational views in enterprise descriptions depend on services.

11.7.3 RELATIONSHIP OF SERVICES TO MESSAGE BUS, WEB OR CLOUD, OR MESSAGE PROTOCOLS

From a protocol/interface binding perspective, services may be deployed in many different ways, built upon different underlying protocol stacks: message bus, Web services, ‘cloud’, or other messaging protocols.

- Message bus: typically uses a bespoke protocol implementation, possibly in an on-board context, or using something like a Pub/Sub message bus. It often uses an API to hide implementation details which may not provide an interoperability specification, but may be layered on underlying Transport/Network Layer like TCP/IP or a Data Link Layer.
- Web or cloud: typically uses HTTP(S)/REST or related protocols on top of a Network/Transport Layer. Server may be on local hardware, or at a data center, or in some ‘cloud’ deployment. The actual end-to-end protocol stack may involve intermediate caching systems for performance.
- Message protocols: may use an interoperable message protocol specification like AMS and layer upon TCP/IP or DTN Network Layer.

Services come in many different deployment ‘flavors’, including: fat client, thin client, browser ‘app’, virtualized, SaaS, Platform as a Service (PaaS), Infrastructure as a Service (IaaS), micro-service, and so on. All use one flavor or another of application and underlying communications protocol stacks with different deployment, granularity, and ownership models.

11.8 SECURITY TOPICS IN THE SERVICES VIEWPOINT

Examples of specific services security topics:

- confidentiality services (encryption);
- integrity services (authentication, verification);
- availability services (redundancy, resilience, Distributed Denial Of Service [DDOS] protection);
- access control services (authentication and authorization);
- procured services, supplier;
- locally built services, supply chain;
- service continuity and recovery;

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

- development services, supply chain verification and validation;
- operations services, source, vetting;
- SaaS, same as procured;
- network (e.g., use Internet Service Provider [ISP] VPN), power conditioning and UPS, heating, ventilation, and air conditioning (HVAC), and other services, integrity, reliability, access control.

12 OPERATIONS VIEWPOINT

12.1 OVERVIEW

The Operations Viewpoint defines the objects, relationships, and rules necessary to describe various space mission operational scenarios. Operational views address specific concerns and represent activities, processes, and behaviors present in mission operations.

12.2 CONCERNS FOR THE OPERATIONS VIEWPOINT

The Operations Viewpoint addresses the following stakeholder concerns:

- the operations enabled by using the system;
- the activities carried out using the system;
- the effectiveness or efficiency of carrying out operations;
- the requirements on operations and activities (from Enterprise Viewpoint);
- responding to constraints (e.g., cost, performance, policies) on operations and activities.

12.3 CONCEPTS FOR THE OPERATIONS VIEWPOINT

Operations Objects are tasks, treated like functions, as abstract representations of behavior. For this purpose they are shown within some system context (swimlanes). Operations Objects may themselves be decomposed into lower-level operations.

Operations views are intended to describe temporal flows among different systems elements as described earlier. Such views may explicitly show timing and/or duration, and are suitable for representing instances, durative events, and sequences of events. They may explicitly identify the Information Objects that are exchanged (tied by correspondence to the Information View where they are defined).

Consistent with other viewpoints, the Operations Viewpoint is based on Operations Objects with a formal relationship among terms. Operations Objects may be operational processes, activities, tasks, and relationships that are represented as using the functions of systems. Operations Objects are typically represented using UML activity diagrams, but optionally Business Process Management Notation (BPMN) (reference [42]) may be used as well.

The Operational Viewpoint may describe any of the following:

- a) activity, procedure, and task all may have stated temporal aspects:
 - 1) may have start time, stop time, or duration specified,
 - 2) may be used to represent both planning views and actual ‘as executed’ views;

- b) action:
 - 1) an action may be instantaneous or have some finite duration,
 - 2) an action may have both planning estimates and actual ‘as executed’ observations;
- c) event:
 - 1) any observable system or natural occurrence,
 - 2) the fundamental entity of observed physical reality represented by a point designated by three coordinates of place and one of time in the space-time continuum postulated by the theory of relativity;
- d) Sequence of Events (SoE):
 - 1) a number of events or activities that come one after another in a particular order;
 - 2) the predicted order of events during spacecraft operations, and also the observed order of events during spacecraft operations.

Operations processes and activities are often described in the context of some enterprise-defined scenario.

A **scenario** is a sequential, narrative account of a hypothetical enterprise concern that provides the catalyst for the exercise and is intended to introduce situations that will inspire responses and thus allow demonstration of the exercise objectives.

A **scenario** may also be a postulated sequence or development of events, or a description of how things might happen in the future under certain circumstances.

12.4 CHARACTERISTICS OF OPERATIONS OBJECT

12.4.1 GENERAL

In the context of figure 12-1, the service and external interfaces of Operations Objects may involve some formalized service protocol, or they may be initiated via email or file transfer requests. Operations Objects may be implemented by people and accessed via phone, text, or Internet requests. They may also be triggered by events that occur inside the system.

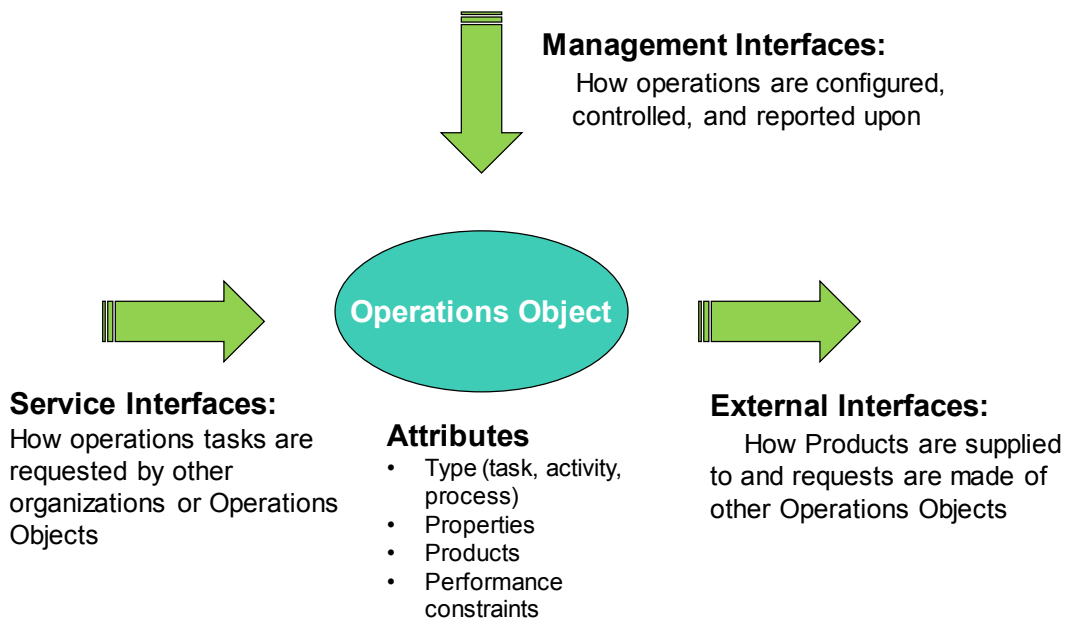


Figure 12-1: Service Object

12.4.2 KEY OBJECTS AND RELATIONSHIPS

12.4.2.1 Ontology of Operations Objects

Figure 12-2 shows the ontology of Operations Objects.

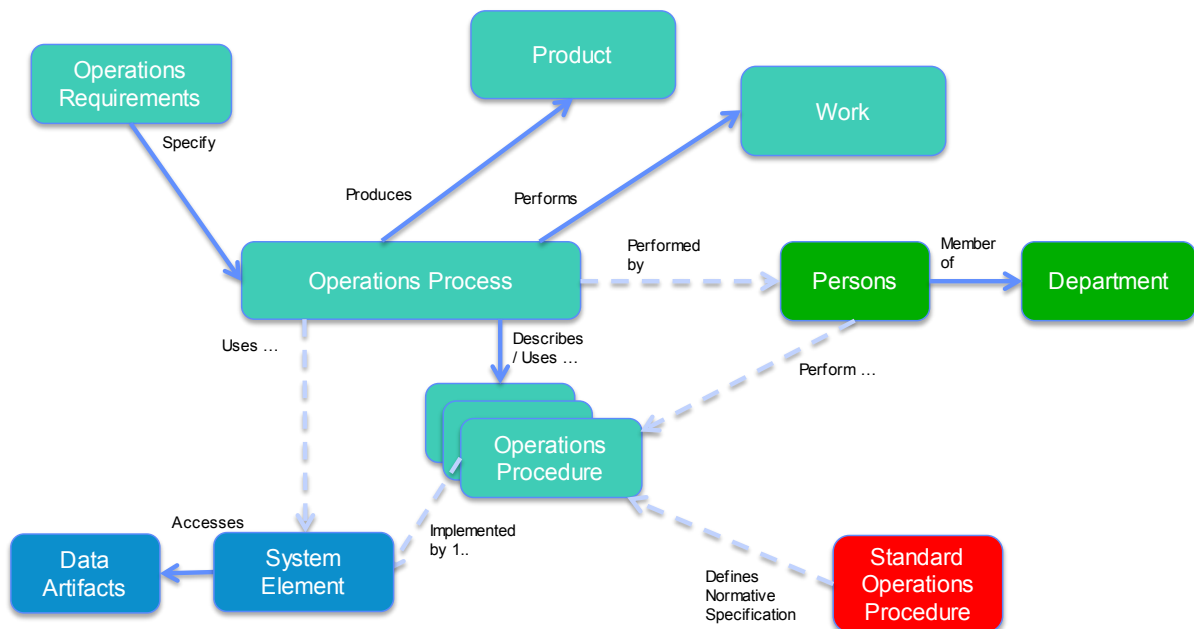
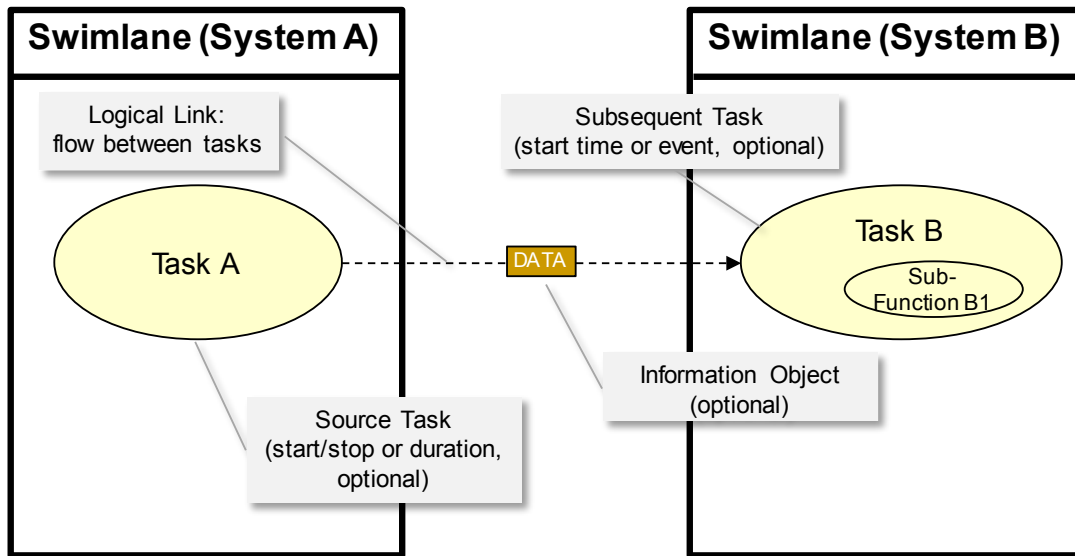


Figure 12-2: Ontology of Operations Viewpoint

In figure 12-2 the relationships among Operations Objects, and those of systems and enterprises is formalized. Operations procedures may be defined by standard operations procedures, and they may be decomposed into lower-level activities and tasks as shown in figure 12-4.

12.4.2.2 Representation of Operations Objects

The preferred representation of Operations Objects is shown in figure 12-3. The representation formalism is borrowed from SysML activity diagrams.



Specific and Generic Object Types and Containment:

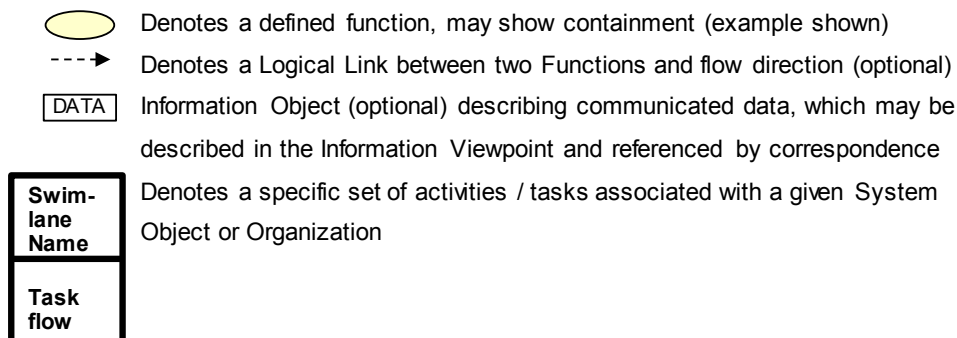


Figure 12-3: Operations Object Representation

Each swimlane is typically associated with a single function group, system/sub-system, or organization and/or team.

12.5 TERMS FOR THE OPERATIONS VIEWPOINT

12.5.1 ATTRIBUTES

There are many different **types** of operations tasks. They may involve mission lifecycle design, strategic and tactical planning, forecasting, commanding and monitoring spacecraft execution, data management and processing, analysis of events, and anomaly handling.

Flows of data are shown using named Data Objects. The formal definitions will be found in Information Views.

There may be different **interaction modes** used among operations elements. These may include request and immediate response, or request with immediate acknowledgement (and later response), or even a request followed some time later with a response or acknowledgement.

A **state** is a condition or situation that determines the set of all sequences of actions in which the object can take part.

A **constraint** is a limitation or implied requirement that limits the design solution or implementation, is not changeable by the enterprise, and is generally non-allocable.

12.5.2 OPERATIONS VIEWPOINT SELECTED TERMS

Operations Objects may be any of the following types:

A **process** is a set of interrelated or interacting activities which transforms inputs into outputs.

An **activity** is a set of cohesive tasks of a process.

A **procedure** is an ordered set of tasks for performing some action.

A **Standard Operations Procedure (SOP)** is a set of instructions used to describe a process or procedure that performs an explicit task or explicit reaction to a given event.

A **task** is a specific defined piece of work that, combined with other identified tasks, composes the work in a specific specialty area or work role.

An **action** is something that happens within an object, either with or without participation of another object. An interaction is an action performed by an object with participation of another object or with its environment.

A **product** is the result of a process.

NOTE – A system as a ‘product’ is what is delivered by systems engineering.

Optionally the set of relationships described in figure 12-4 may have a temporal aspect (i.e., start and stop times, duration) which should be interpreted as an extension of the core model.

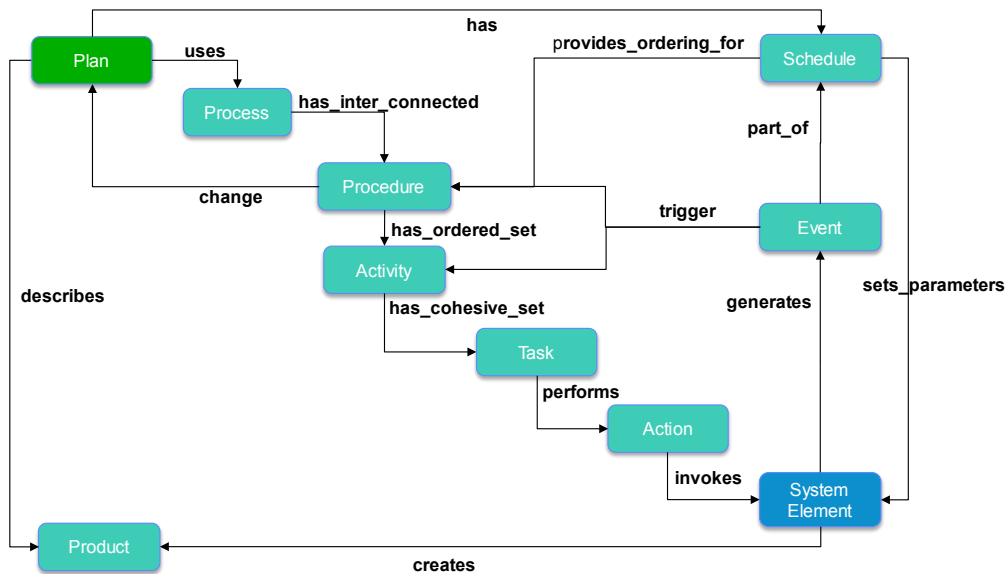


Figure 12-4: Operations Viewpoint Core Model with Temporal Aspects ('Schedule')

Temporal aspects extend the core operations model by introducing the following constructs (see figure 12-4):

- a) plan: the (acceptable) balance of risk vs. result in generating an operational product (e.g., nominal or off-nominal plans);
- b) schedule: the temporal context for plan. (e.g., keeps track of SoE [anomaly vs. 'as predicted/expected']);
- c) process: what needs to happen in the plan to generate an operational product.

12.6 EXAMPLES OF OPERATIONS VIEWS

12.6.1 GENERAL

The technical details of the system elements that implement operations will typically be defined in separate views. An activity that is part of a operations process may be implemented by people, or by software and hardware components that are invoked or monitored by people. In its simplest form, an Operations View may be described by an activity or sequence diagram showing operations and data within a single system element, or it may include flows among different system elements.

An Operations View may just be a simple ordering of activities, or it may be designed to orchestrate a carefully timed sequence of events, potentially tied to external or natural phenomena. It may require specification of start and stop times, or of event times. And it is often the case that the Operations Views need to be able to describe the timing of planned

activities and to support the comparison of what was planned to the actual timing that occurred.

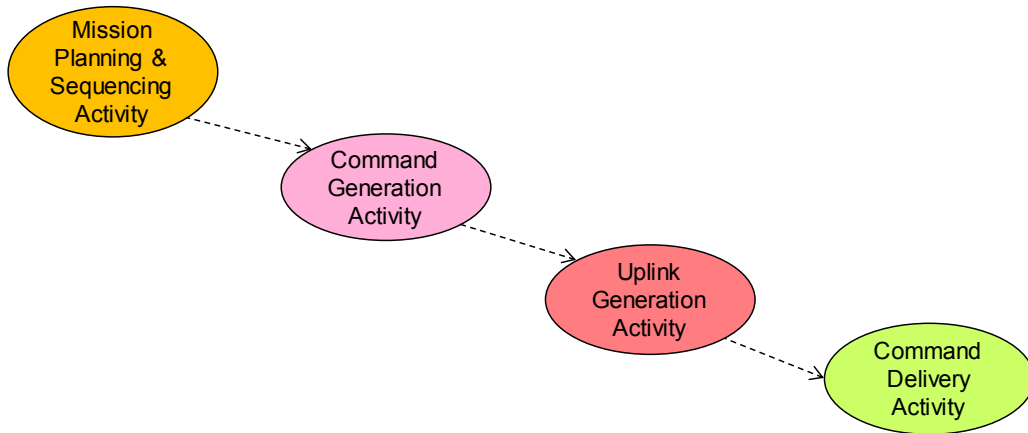


Figure 12-5: Simple Example of an Operations View

A simple example of an Operations View is shown in figure 12-5, in which the activities involved in mission planning, command generation, and delivery are represented.

Swimlanes, as shown in the more complex figure 12-6, are typically associated with function groups, systems/subsystems, and other organizations and/or teams. This includes identification of Data Objects that flow among activities.

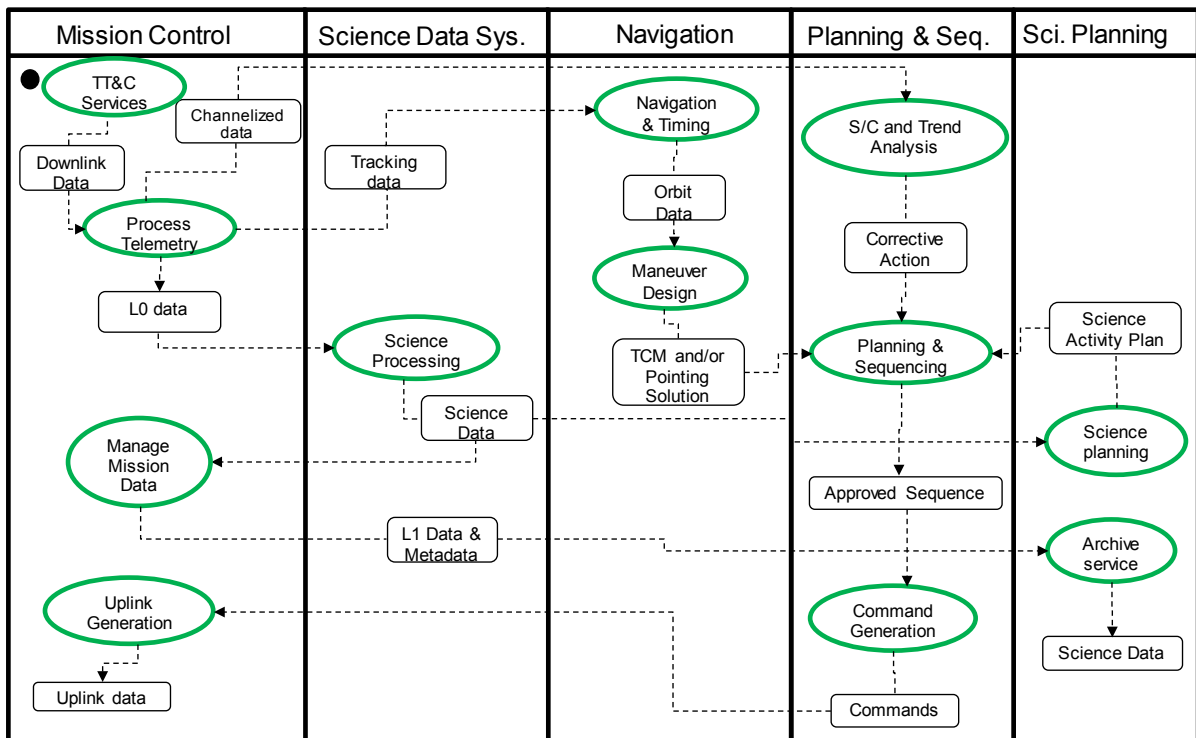


Figure 12-6: Operations Viewpoint Mission System Tasks and Data Flows

12.6.2 RELATIONSHIP TO FUNCTIONAL VIEWPOINT

The Functional Viewpoint documents functions (and associated functional groups) along with any defined abstract interfaces. It also may document any data being exchanged, which should be carefully defined in separate Information Views. However, the Functional Viewpoint does not show any temporal aspects or recursion.

The Operations Viewpoint describes how to perform a process (a set of functions or activities) to accomplish tasks and associates tasks with system elements (vertical or horizontal swimlanes).

An Operations Viewpoint may show temporal ordering of flows (starting upper left in this case) between activities (or tasks), including recursion. It may also capture timing (start/stop, duration, or events) of activities (or tasks) and names the Data Objects that are exchanged. As a byproduct of temporal representation, Operations Views may also represent both predicted and actual observed sequences of events.

12.6.3 RELATIONSHIP OF OPERATIONS TO ENTERPRISE, SYSTEMS, AND INFORMATION

Operations gets guidance from the Enterprise in the form of requirements, objectives, people, and policies, and its interfaces with actual systems are described using the Connectivity Viewpoint. Correspondence to information is described by referencing defined instances of documents, agreements, contracts, policies, requirements, objectives, goals, and operational scenarios.

Boundaries of responsibility or ownership related to enterprise are described using domain definitions.

12.7 SECURITY TOPICS IN THE OPERATIONS VIEWPOINT

Examples of specific operations security topics:

- establishing Service Level Agreements (SLAs);
- managing access control to facilities;
- assigning roles and permissions;
- establishing operations continuity and recovery processes;
- acquiring operations services, sources, vetting;
- processes for handling operational data privacy and authenticity.

13 DERIVING OTHER VIEWS FROM THE BASIC VIEWPOINTS

RASDSv2 provides a general set of viewpoints that may be used to describe many aspects of space system architectures. However, depending on the set of concerns that need to be addressed during design of a particular space system, other views may need to be constructed from the objects and relationships described in the nine basic viewpoints. This may be done either by defining correspondences to objects defined in two or three existing viewpoints, as was done to create the Service Viewpoint, or by defining entirely new classes of objects or relationships using the RASDSv2 object and ontology rules to create a new RASDSv2-style viewpoint.

Any new viewpoint would ideally define the primary objects being addressed, their relationships to other object types, the concerns being addressed, the selected representation, and some discussion of how and where it might be used.

As has been demonstrated, providing adequate visibility into particular design concerns can often be accomplished by showing two related views on one diagram and explicitly describing the correspondences between them. In some cases it may be necessary to define new constructs, but this should be done with care, to avoid cluttering diagrams with new elements that then need to be defined and explained.

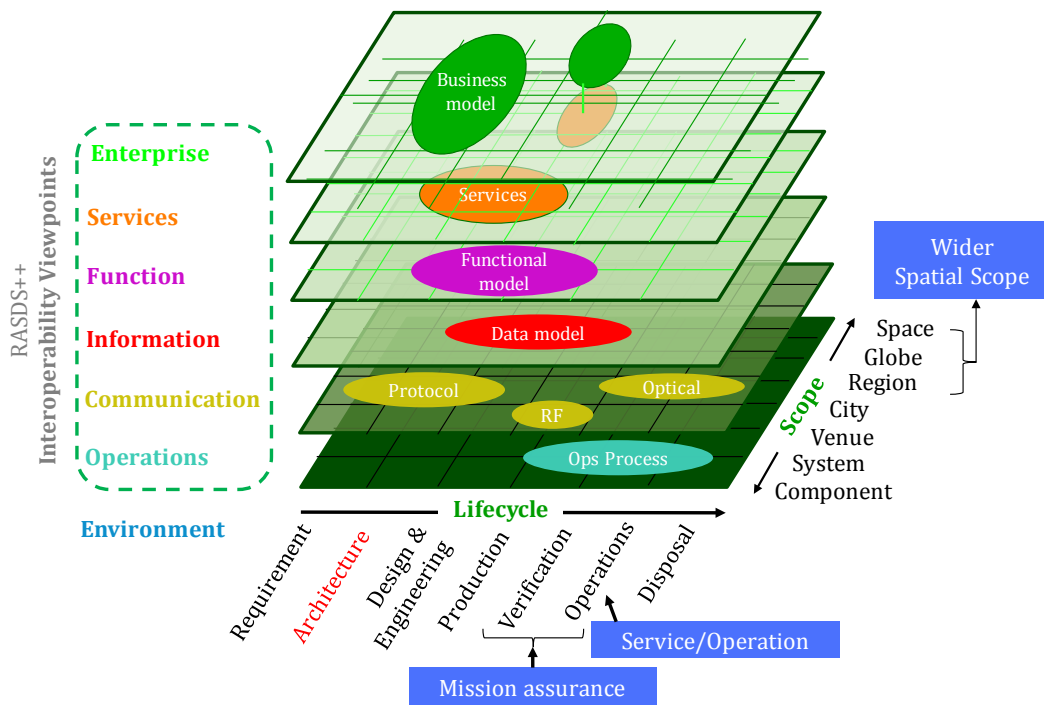


Figure 13-1: RASDSv2 in Wider Social Context²¹

²¹ Source: Koki Asari, Japan Space Systems, ISO TC 20/SC 14/WG8.

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Figure 13-1 is an example of adopting large parts of the RASDSv2 viewpoint structures and adding two new dimensions to reflect lifecycle stages or social considerations that may occur in systems engineering projects. This diagram makes explicit that RASDSv2 may be applied at different scoping levels, from component, to city or regional scale, and out to space. The other dimension explicitly identified in this figure relates to project lifecycle stages, from requirements, to architecture, and through development, to operations, and disposal.

There is nothing inherent in the RASDSv2 framework that prevents it from being used at these different scales and lifecycle phases. The object types and relationships are clear, but generalized, so that they can be specialized as needed for different purposes.

ANNEX A

NOTES ON USE OF RASDSv2 TO DO SYSTEMS ARCHITECTURE

(INFORMATIVE)

A1 INTRODUCTION

A detailed treatise on how to *do* system architecture is far too complex to attempt in these pages. The interested reader who wishes to learn more is directed to any of the available texts that address this topic, Rehtin and Maier, is very well regarded (Mark W. Maier and Eberhardt Rehtin, *The Art of Systems Architecting*, 2nd edition, CRC Press, 2000).

The RASDSv2 methodology may be applied in many different projects and scopes as part of a system architecture description or design process. It has been applied to individual components and protocols, to systems, and to systems of systems. Not all the viewpoints of RASDSv2 need to be used for all problems. Regardless of the scale of the project, the first questions that are asked are usually:

- Who are the stakeholders?
- What are their concerns and which views address them?
- What is to be described with this set of architectural views?
- What is the right level of detail to expose during the process?

In many applications of RASDSv2 only two or three viewpoints may be needed, and only one or two views may be needed to address different concerns in each viewpoint.

Each viewpoint specification in RASDSv2 defines the typical stakeholders and concerns, and defines the kinds of objects, their description, and relationships that may appear in any view on a system. A set of recommended representations is also provided for each viewpoint. Different representations may be selected instead of those offered here, but the recommendation is that whatever choices are made that they be clearly documented and consistently used. The same is true of any color codes that might be adopted. ISO 42010 provides a Legend element of an Architecture Description Framework (ADF) as a place to display such aspects.

During development of system views all these constructs should be treated as constraints on what may be represented in any given view. For each element in any given viewpoint a set of attributes are specified in sections 4–12. Not all attributes are needed in all views for any given viewpoint, and not all attributes are relevant for all objects in a view. Choose the ones that are going to be most effective for the stakeholders that each view is intended to address.

Furthermore, RASDSv2 does not provide any single method for capturing these attributes in each view. In some cases they may be shown as notes, in others as tagged values associated

with any element, or as implicit or explicit correspondence relationships. In still others they may be shown in a separate table. There are formal methods of capturing these views, as is discussed in the other annexes, that offer suitable means for capturing these object attributes.

The following subsections provide some brief examples of the sorts of heuristics that one can apply while using RASDSv2 to produce a Functional View on a system. Similar heuristics would be applied for each of the other views that need to be generated, possibly at different levels of detail. So a high-level Connectivity View showing the major nodes and links in a system may be accompanied by one or more detailed views that drill into the internal composition of those nodes and the connectivity approaches adopted among all of the lower level components.

A2 EXAMPLE METHODOLOGY FOR CONSTRUCTING FUNCTIONAL VIEWS:

- For each stakeholder concern, the Functional Objects and interactions relevant to the concern are identified.
- For each Functional Object, the services provided to other Functional Objects are identified.
- For each Functional Object, the services used from other Functional Objects, if any, are identified.
- The cooperative actions performed by multiple Functional Objects, if any, are described.
- The resulting view is checked against the structuring rules.
- At least the abstract data types that are exchanged across each interface are identified.
- Constraints on interactions among Functional Objects are identified.

A3 EXAMPLE STRUCTURING RULES FOR THE FUNCTIONAL VIEW

- Each Functional Object that is inside the system has at least one logical link.
- Each logical link is connected to at least one Functional Object inside the system and to at least one more Functional Object either inside or outside the system.
- Each Functional Object or logical link has a unique name.
- Each Functional Object provides at least one of: generation of data, transformation of data, initiation of action, or response to stimulus function.
- Each Functional Object has a defined set of interfaces.
- Each Functional Object has a defined set of behaviors and actions.
- Information is available:

- To explain the Functional Objects;
 - To explain the logical links;
 - To indicate whether a Functional Object is inside or outside the system.
- Taken in total, the Functional Objects, logical links, and attached notes completely address the concerns of the functional viewpoint at the level of detail appropriate for the audience.

A4 RELATED EXAMPLE METHODOLOGY FOR CONNECTIVITY VIEW

A methodology similar to that of the Functional View is followed for the Connectivity View. All the engineering objects that need to be represented to capture the breadth of the system implementation design, and enough of its required context, need to be identified. Mappings should be made from identified Functional Objects to engineering objects in the Connectivity View.

- Each Functional Object should be mapped to at least one engineering object in a connectivity view.
- For each logical link in a functional view it should be clear which physical links in a connectivity view support the actual communications.
- The performance envelope required by the assembled set of engineering objects should be described, and whether the capabilities provided by the nodes and links are adequate to meet requirements should be evaluated and documented.
- The interactions of the engineered objects with one another, and with the environment, should be documented.
- The ability of the engineered elements of the system to meet the performance requirements should be evaluated.
- The resulting views should be checked against the Connectivity View structuring rules and cross checked with the Functional View for completeness.
- The steps above may be iterated if necessary.
- Trade studies may be supported by creating more than one mapping of Functional Objects to engineering objects, and one of more approaches for creating those engineering objects may be evaluated.

ANNEX B

FORMAL METHODS AND TOOLS

(INFORMATIVE)

As noted in the Introduction to this document, one of the primary motivations for the RASDSv2 is to provide a system architecture methodology that domain experts can use to describe and construct many different specific space system architectures for complex systems. The RASDSv2 methodology can be used very effectively in its current form to describe systems architectures, provide viewpoints from which to examine them, related representations of architectures, guidelines for concerns to be addressed, and issues to consider at each viewpoint. Even in the absence of a more formal notation, tools to support design, or formal models the consistent use of these concepts, methodology, and representational formalisms can help enormously in clarifying architectural descriptions and design.

This methodology has been successfully validated through use in describing real space systems. Several different missions and projects have used RASDSv2 to describe their high-level architectures, and several CCSDS working groups have successfully used RASDSv2 to document individual standards and complex reference architectures involving many tens of standards. Feedback from all these activities has helped to refine the present document.

Although RASDSv2 is useful even if it is only used to guide selection of useful views and their contents in a set of ‘design drawings’, for RASDSv2 to be most useful for large scale systems design, tools are required that will permit the ready creation of system descriptions and that will automatically maintain the complex relationships between objects as seen from different viewpoints. This requires both a mapping into some more formalized methodology and tools that implement it.

Architectures for several projects that use RASDSv2 viewpoints and models have been developed using SysML formalisms. These have been effective, and have shown the strengths and limitations of current modeling environments. SysML, which is based upon UML 2.0, has been used because it already provides formalisms for requirements, verification, viewpoints, describing hardware and software objects, and handling of discrete and continuous data flows.

SysML provides a set of techniques and formalisms for modeling systems and software. It brings a rich set of formalized modeling constructs that permit description of systems from several different aspects. These aspects are represented by a set of diagram types, each of which maps to some underlying constructs in the model that the implementing tools maintain. There are two very important concepts to keep clear about:

- a) In SysML the model is in the tool, the drawings are just external representations of the model.

- b) In order to adequately represent stakeholder viewpoints, diagrams must be constructed carefully, with defined objects and semantics used for each view.

The SysML methodology extends UML 2.0 by adding requirements, verification, and parametrics to the UML suite of diagram types. The SysML specification also supports modeling semantics for continuous as well as discrete behavior. This provides good support in a general way for many front-end systems engineering and architecting processes.

SysML has incorporated the concepts of viewpoint and view, but it does not define any specific instances of these. As part of the earlier CCSDS SAWG studies an analysis was done of the capabilities defined within SysML and how it would map to the kinds of constructs needed by RASDSv2. The biggest realization was that many of these SysML diagram types could be used for more than one viewpoint, and that what differentiated the different viewpoint diagrams were the object types that were represented in any given viewpoint and the nature of the relationships that would be depicted.

The following description of the relationships between RASDSv2 viewpoints and SysML diagram types, while not complete, is intended to provide guidance to any group that wishes to use SysML to represent RASDSv2 viewpoints. The SysML constructs suggested for views in each viewpoint are underlined:

- Enterprise Viewpoint:
 - Organizational structure and behavior diagrams;
 - Organizational use case, activity, and sequence diagrams;
 - Requirements and constraints for rules, policies and agreements;
- Functional Viewpoint:
 - logical structure, behavior and package diagrams;
 - functional activity, state chart, parametric, and sequence diagrams;
- Connectivity Viewpoint:
 - physical block definition, composition, behavior and class diagrams;
 - parametric diagram for performance and physical link characterization;
 - UML deployment diagrams needed for allocation views;
- Information Viewpoint:
 - information block, package and parametric diagrams;
- Communication Viewpoint:
 - protocol structure and behavior diagrams;
 - Protocol state machine, PDU sequence, and activity diagrams.

Clearly there is not just a one-for-one mapping between SysML diagram types and the kinds of constructs needed for RASDSv2 viewpoints. However, there is a sensible mapping for all the RASDSv2 constructs into at least one of the SysML types once the stereotyping for the particular RASDSv2 object and relationship classes has been made. The fact that there is much more precision of expression possible with SysML models, instead of PowerPoint drawings, means that additional information can be conveyed in any models developed within these frameworks.

The fact that model element attributes can be specified directly within the modeling environment, and that models can be checked for validity and completeness, makes this a particularly attractive approach for describing complex space system architectures.

Annex C provides some worked examples of using SysML tools to describe complex systems architectures.

ANNEX C

RASDSv2 AND SYSML EXAMPLES

(INFORMATIVE)

C1 INTRODUCTION

Since it was first published RASDSv2 has been used within CCSDS to describe protocol standards, various frameworks, and systems architectures involving the use of many standards. It has also been used for a number of projects that have been written about and published. Some of these have involved the use of SysML to provide accurate models of systems and to take advantage of the power and functionality of SysML modeling environments.

This annex uses some of the materials that have been published elsewhere to provide some worked examples of RASDSv2 models developed using SysML. They can provide useful examples of patterns that might be adopted at various scales to model complex systems architectures. These figures have been annotated to show the mapping from SysML diagram types to RASDSv2 views.

C2 CREATE A PROFILE

One of the first useful steps in modeling a complex system is to define a SysML profile and a decomposition hierarchy for the elements to be used in the model. This creates a sort of meta-framework for the model and defines the sets of terms and relationships to be used in the model.

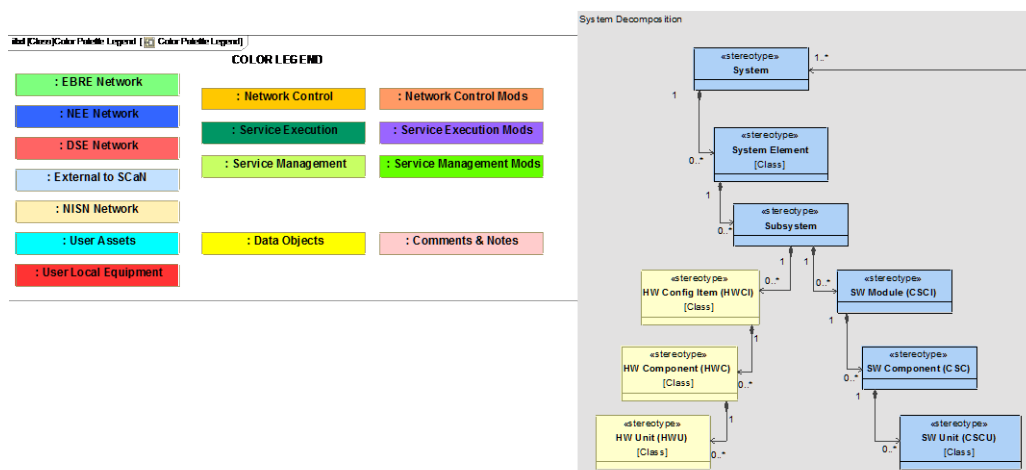


Figure C-1: Example SysML Profile

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

This and several of the following figures are borrowed from a SpaceOps 2012 paper titled “NASA Integrated Network Monitor and Control Software Architecture” (reference [29]). Figure C-1 defines a set of system decomposition stereotypes and also a set of color codes that were used in this particular model. These are used consistently in the following figures, but there is nothing in this section that should be taken as required; they are just offered as a set of coherent, real world, examples.

Within a specific model these stereotypes and color codes should be defined to have meaning in the modeling context, which in this case was a systems of systems model. The stereotypes that are defined may have as many levels as are needed, and will define the specific terms adopted for the system decomposition. It may be noted that in this figure the top three levels of decomposition are all different levels of ‘systems’ decomposition and that below that level the components are distinguished as either software or hardware.

C3 DEFINE THE TOP LEVEL DECOMPOSITION

A next step in the SysML model process may be to define the top level decomposition of the model, using the newly defined stereotypes. Since this example reflects the decomposition of a space communications system it offers an example of a RASDSv2 Connectivity Viewpoint, but one focused on the decomposition of the system into parts rather than just on the connections among them. This is one of several possible Connectivity views that might be used.

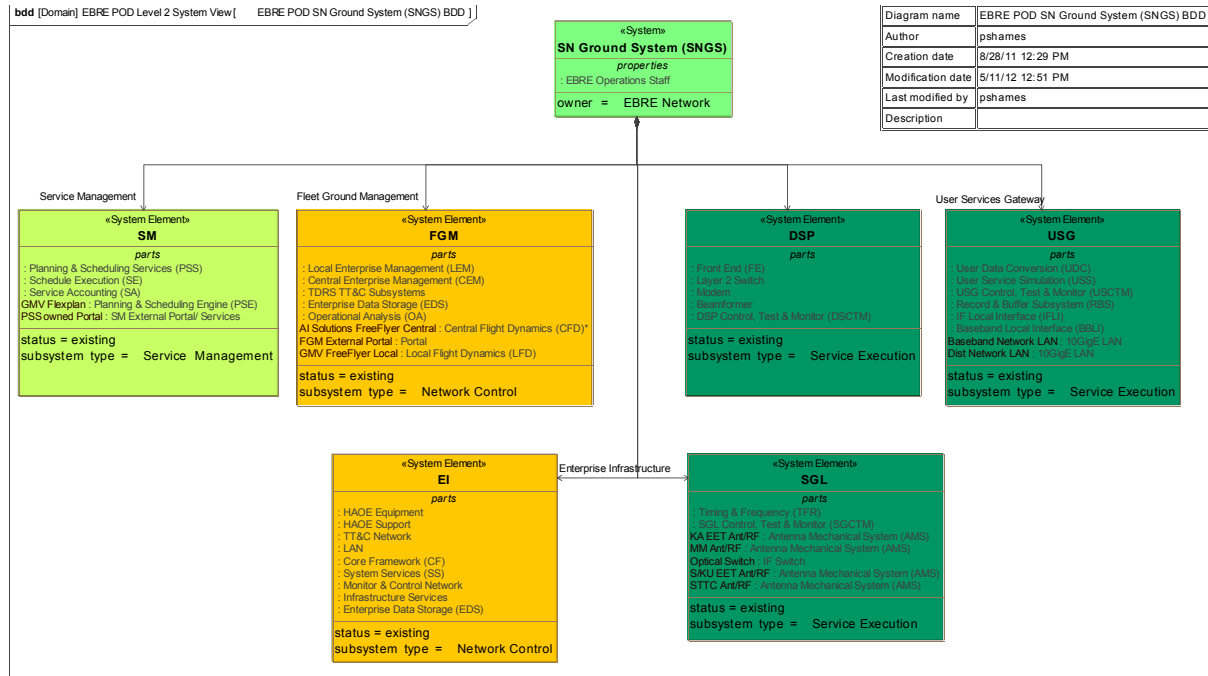


Figure C-2: Example of RASDSv2 Connectivity View Showing Composition of Abstract System Objects

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Figure C-2 uses the stereotypes and also the color codes defined in figure C-1. The top level object is one of the defined <<System>> objects and the other components are <<System Element>> and adopt the color codes used to distinguish the different kinds of elements shown on the related diagrams. This particular example also shows, within each <<System Element>> the parts that compose each one, which may be elaborated on lower-level diagrams.

Now that these components have been defined, they can be used in other Connectivity diagrams showing the communications connections among the different parts, and other details of the systems and communications structures, as needed.

C4 DESCRIBE THE DECOMPOSITION IN A CONNECTIVITY VIEW

Each of the objects defined in the top-level decomposition will have its own decomposition and internal structures. Figure C-3 shows these features for the top level <<System>> named SNGS.

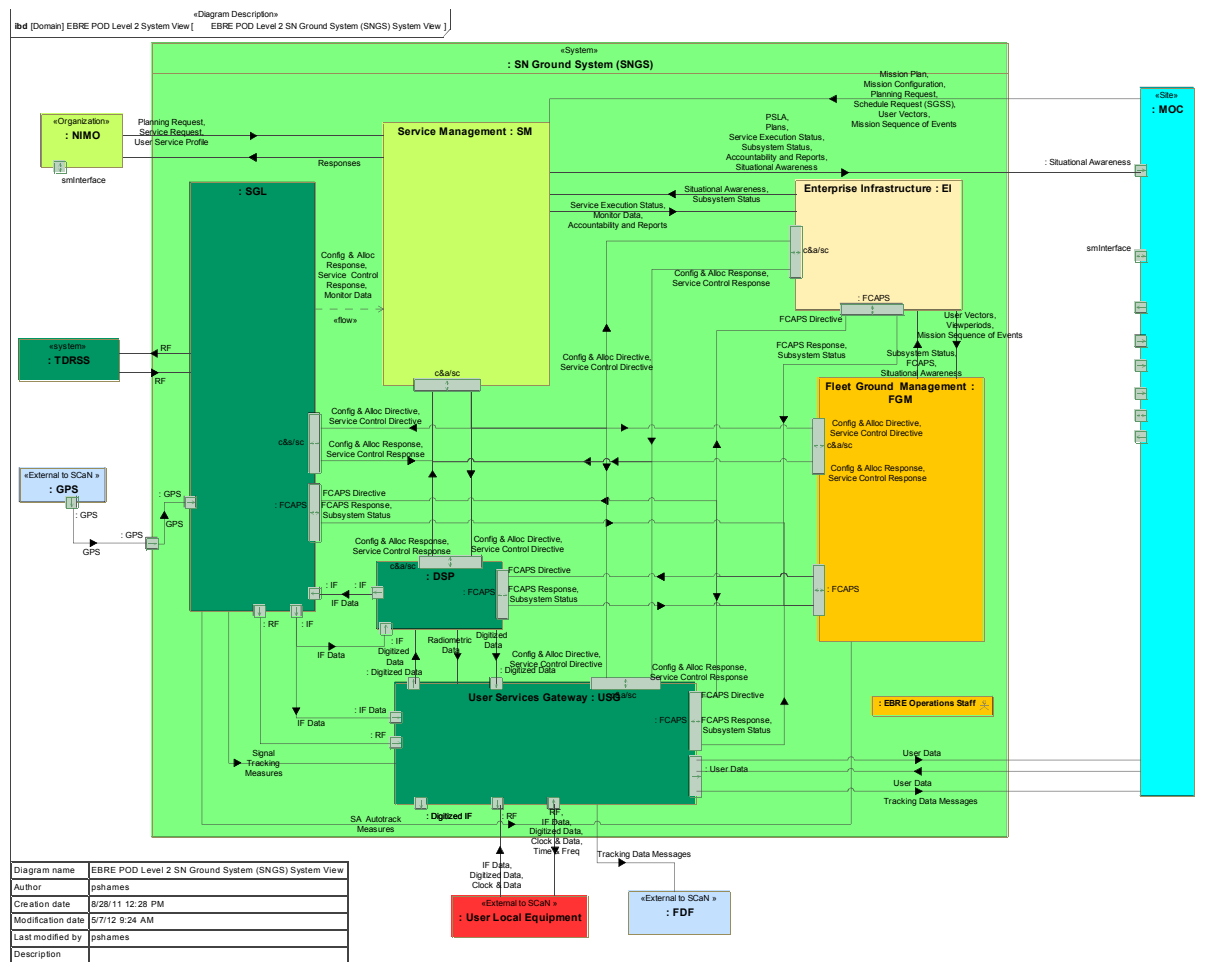


Figure C-3: Example of RASDSv2 Connectivity View Showing Abstract Interfaces, Communications Links, and Data Flows among Connectivity Objects

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Figure C-3 uses the same decomposition hierarchy and color coding as defined in the Profile. This is a Connectivity View showing the major <<System Elements>>, their interfaces, and data flows. These interfaces and flows are named in this view, but the details are defined, by correspondence, in other views.

Several variants on these kinds of views, as shown in figures C-2 and C-3, may be developed as part of trade studies intended to determine the best way to partition the functions in a system.

C5 DESCRIBE THE INTERNAL STRUCTURE IN A CONNECTIVITY VIEW

Each of the objects defined at any level of decomposition may have their own diagram(s) showing their internal functional structures. Of course, this process of successive decomposition is a familiar one, and it may be carried out, in these architecture diagrams, down to whatever level is useful. Figure C-4 is an example Connectivity diagram showing the internal structures of a <<SW Subsystem>>, one that is largely software, named Service Management.

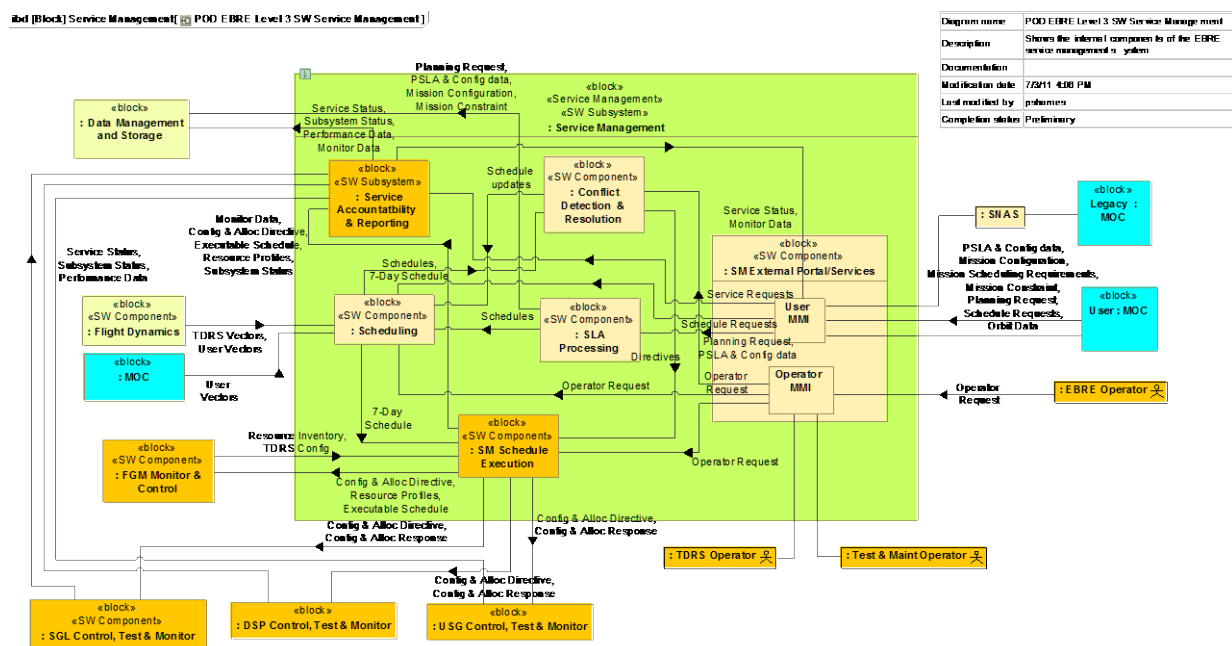


Figure C-4: Example of RASDSv2 Connectivity View Showing Communication Flows among Connectivity Objects

Whereas in figure C-3 the interfaces among <<System Elements>> were explicitly identified in this case they are not. This detail is a representation choice and these interfaces can be explicit, or left off, at either level. Choosing to show them on figure C-3 was an acknowledgement that interface control among major <<System Element>> is critical to being able to control the interactions among these elements.

This decomposition process may be carried out down to whatever level of detail is appropriate for the given project. This entire modeling framework may be built upon, and further elaborated, as the development transitions from architecture into design.

C6 DESCRIBE THE SYSTEM DEPLOYMENT IN A CONNECTIVITY VIEW

Having defined the decomposition of some set of system elements it is often useful, in a different kind of Connectivity View, to describe how and where the different elements are to be deployed. As noted in figure C-4, more than one deployment diagram may be developed as part of trade studies to determine the best allocation of functions to different sites.

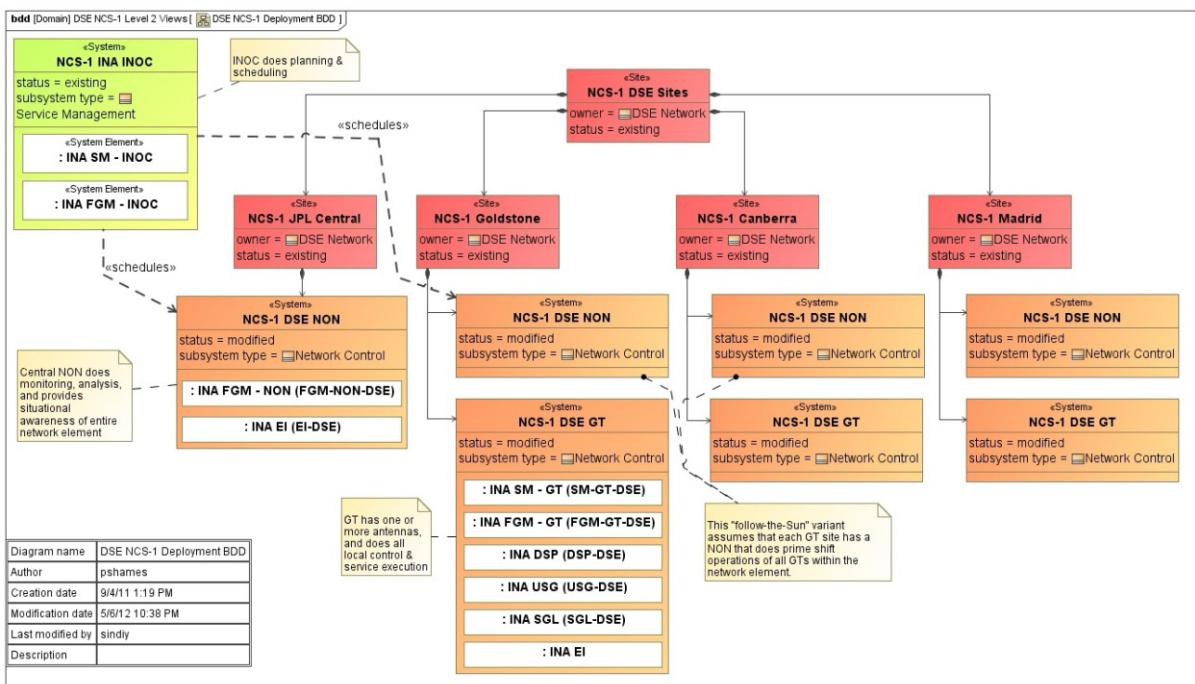


Figure C-5: Example of RASDSv2 Connectivity View Showing Allocation of System Elements to Different Physical Sites

Figure C-5 is a Connectivity View that explicitly shows the allocation of different <<System>> components to a separate set of <<Site>> objects. Within the modeling tool the appearance of the Blocks are identical, but the use of stereotypes allows different kinds of objects to be clearly identified.

Figure C-5 also adopts another other useful feature, using correspondence to show the relationship between the separate <<System>>, called the Integrated Network Operations Center (INOC), which hosts the Service Management <<Subsystem>> and the Network Control <<Subsystem>> allocated at each <<Site>>.

C7 DESCRIBE THE SYSTEM INTERFACES

Where such details are needed, various kinds of interfaces may also be modeling in SysML. The following figures are borrowed from INCOSE 2016 paper titled “A Representative Application of a Layered Interface Modeling Pattern” (reference [28]) that provides a consistent set of examples for how to model the interfaces of systems components (both hardware and software), the protocols and their behavior, and the Information Objects that are passed from one system element to another. This set of figures are different in style, and they do not explicitly define either a profile nor a set of color codes, but do name the different component types.

The value of these diagrams is that they describe how to model the details of the interfaces identified in figure C-3 and the data flows and Data Objects shown in figure C-4, as well as Protocol Entity interfaces and behaviors.

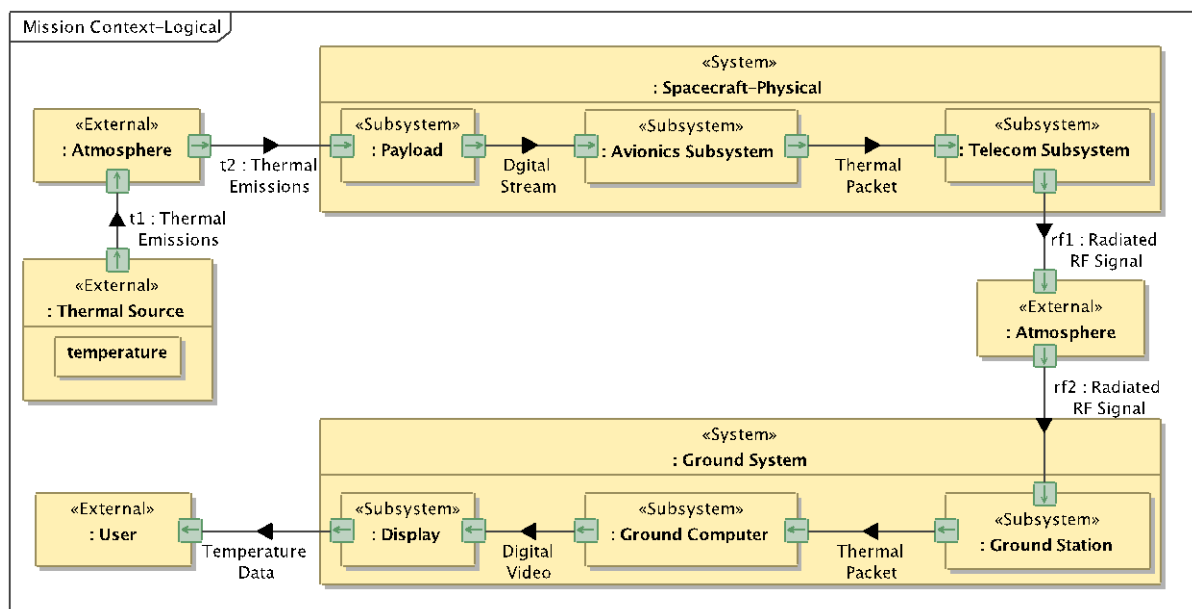


Figure C-6: Example of RASDSv2 High-level Connectivity View Providing Context for Interface Modeling²²

Figure C-6 is a top level RASDSv2 Connectivity View showing a set of major <<System>> elements as well as <<External>> physical elements like a Thermal Source and the Atmosphere that RF signals radiate through. This is the base model for the following set of diagrams, and it roughly corresponds to the top level Connectivity View shown figure C-3. The following subsections will dive down into the technical details of how to formally model these interfaces, protocols, and Data Objects.

²² Copyright INCOSE IS 2016 by Peter Shames, Sandy Freidenthal, Marc Sarrel. Used with permission.

C8 DESCRIBE THE COMPONENT INTERFACES USING A COMMUNICATIONS VIEW

In figure C-7 a RASDSv2 Communication View describes, in more detail, the protocol stacks and interfaces by which two of the <<Subsystem>> elements introduced in figure C-6 actually communicate. This level of detail is not always needed in high level systems architectures, but for subsystem level systems architecture descriptions it is usually required.

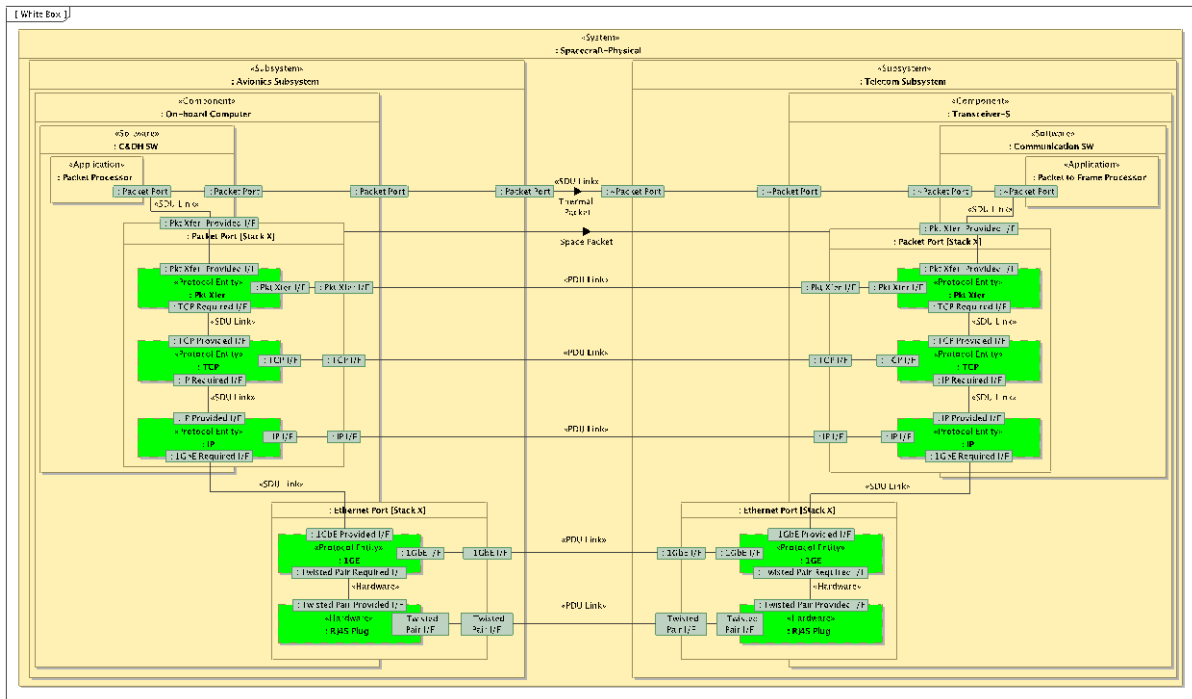


Figure C-7: Example of RASDSv2 Communications View for Component Interface Modeling

What is shown in figure C-7 are two of the <<Subsystem>> from figure C-6, the Avionics Subsystem and the Telecom Subsystem. These components are referenced by correspondence, and are included to provide context, because the real focus in this Communications View diagram are the protocol details describing the interfaces of these two components.

Two levels of detail of these <<Component>> interfaces are shown, both the software based ‘Packet Port’ on the two C&DH SW and Communications SW components and the lowest level hardware based ‘Ethernet Port’ on the edge of each of the two <<Component>>. These ports map to ISO BRM layers 7 (application packets), 4 (TCP), 3 (IP), 2 (1 Gb Ethernet), and 1 (RJ-45 plug). The top three layers are software, the bottom two are hardware, and these hardware ports are literally shown at the edge of the Computer and Transceiver <<Component>>.

The <<Protocol Entity>> elements themselves each show the interfaces that are identified in RASDSv2 Communication Views for Protocol elements, named as provided, required, and

peer protocol interfaces. Drilling down further, figure C-8 shows an abstract representation of a <<Protocol Entity>>, showing all three interfaces of the <<Protocol Entity>> and an abstract view of the behavior of one of these entities that implements a specific protocol layer, TCP (reference [47]) in this case.

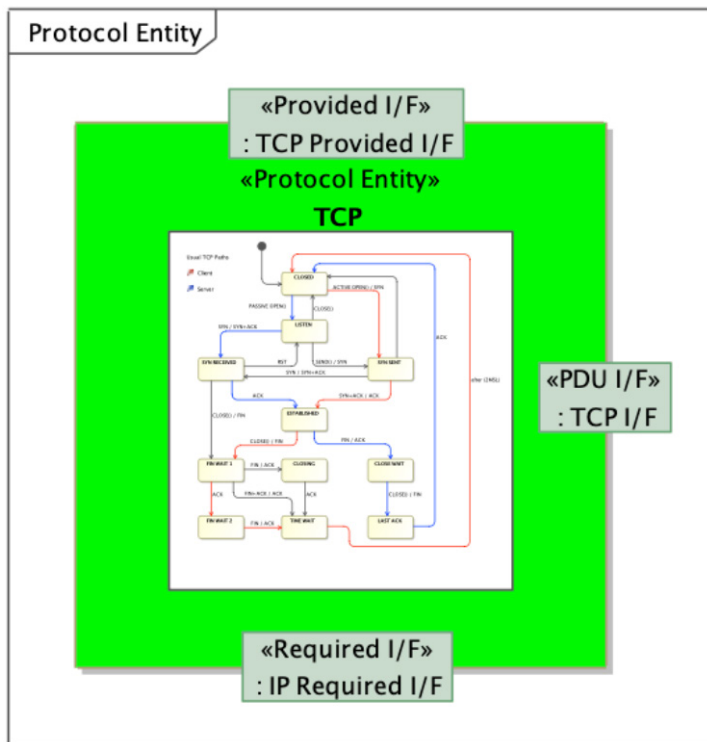


Figure C-8: RASDSv2 Abstract Model of a Protocol Entity Showing Behavior and Interfaces

Every Protocol Entity has similar features to those shown in figure C-8. A typical protocol standard will formally define the PDUs that get exchanged across the peer-to-peer interfaces and also the behavior of the protocol, which is abstractly shown inside the <<Protocol Entity>>. In many protocol standards the Required and Provided Interfaces will also be documented formally, but this is not always done.

A formal model of the behavior inside this Protocol Entity is shown in figure C-9. In this example the behavior of this Communications Viewpoint element is shown using a SysML State Chart. This example shows the internal connection establishment behavior of the client/server pair of TCP protocol entities that make up ISO Layer 4 of the component interface shown in figure C-7.

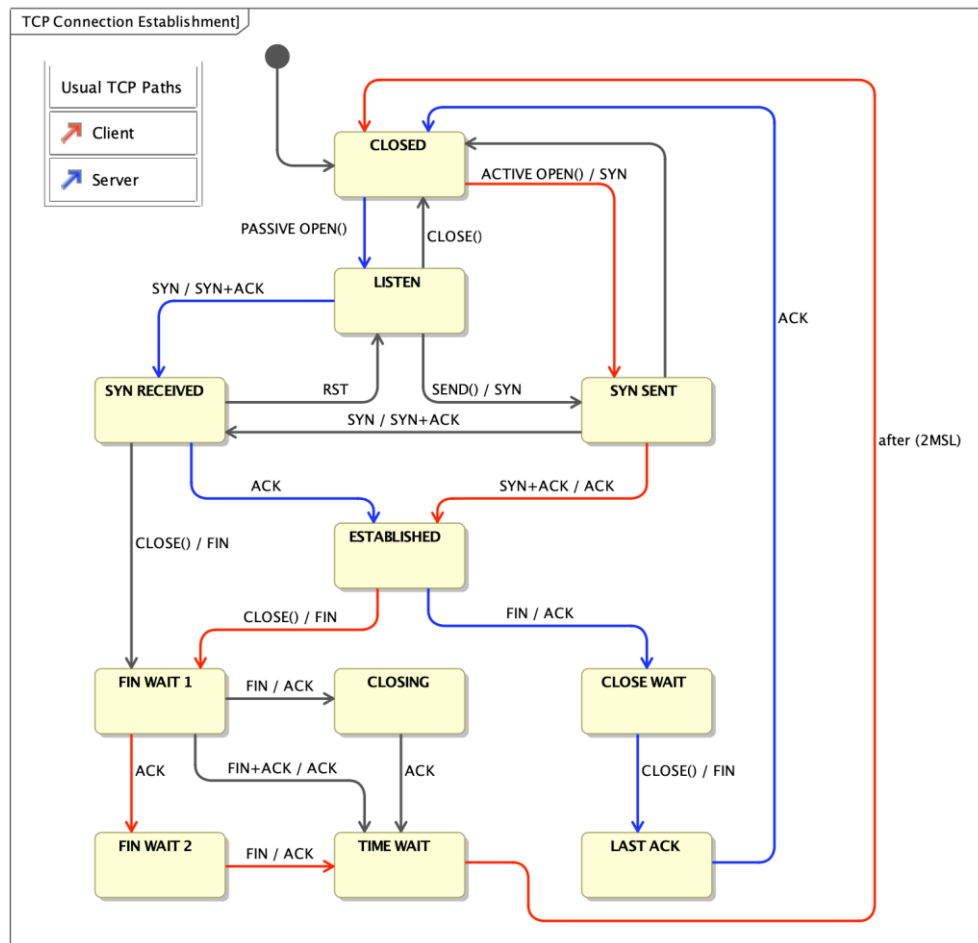


Figure C-9: RASDSv2 State Machine Model of TCP Protocol Connection Establishment Behavior

The behavior of the protocol entities in the central ‘Connection Established’ state may also be modeled in detail, describing the transfer of data PDUs between the two peer entities. SysML sequence diagrams may also be employed to show the sequence and timing of PDU flows between two cooperating peer protocol entities.

C9 DESCRIBE THE DATA OBJECTS IN AN INFORMATION VIEW

All the PDUs that are exchanged between Protocol Entities may be modelled as RASDSv2 Information Objects. These must be defined in a model as well, along with any other Data Objects that are exchanged at the Application Layer. Figure C-10 shows a method for modeling these Information Objects.

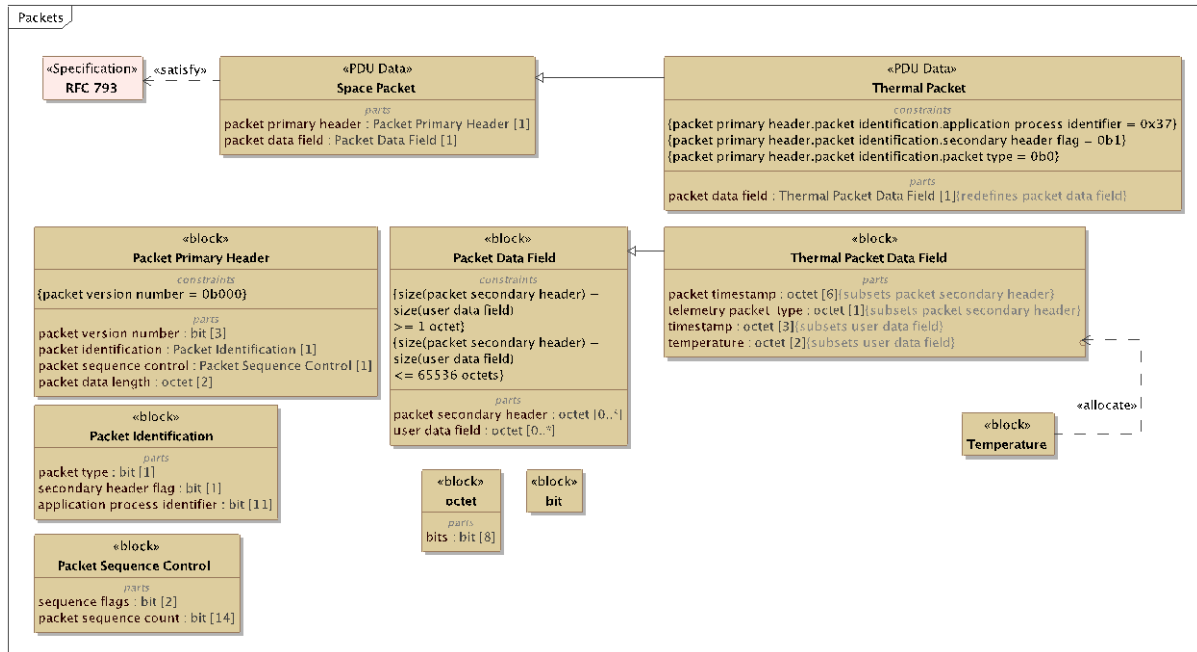


Figure C-10: RASDSv2 Model of Abstract and Realized Information Objects

Figure C-10 provides examples for modeling Information Objects from two different points of view, both abstract (bit, octet, and SPP Primary Header and Packet Data Field structures, shown on the left) and realized (Thermal Packet as a realized instance of a Space Packet, shown on the right). Of course, once the abstract Packet structures are defined all sorts of models of other realized data packets may easily be defined.

C10 PROVIDING RASDSv2 VIEWPOINTS IN SYSML

The main focus in this annex has been on providing examples of how SysML has been used to do formalized modeling of RASDSv2 compliant systems architectures. There are significant advantages to using formal modeling tools in that they provide automated means for creating and managing models, model elements, and model prototypes. The tools will also automatically maintain consistency where they can. But, as should be clear by now, these modeling tools do not provide all of the needed model elements, nor viewpoints and views, that are needed to do systems architecture modeling. SysML defines the notions of viewpoint and view, but does not provide a useful ‘starter set’ of viewpoints.

RASDSv2 provides a useful set of viewpoints and views, but work must be done to create the necessary conceptual framework within a SysML tool to support these kinds of models. This section has provided some examples of how that has been done, but no pre-built ‘package’ is available to provide a set of building blocks. It is strongly recommended that any project that wishes to use SysML take some prior steps before starting to model a system; it should:

- a) identify the sets of viewpoints and views that will be needed for the model and establish the specific SysML diagram types to be used, along with stereotyped objects and formalized terms for each viewpoint;
- b) make the initial determinations of just what the scope is of the modeling effort and determine if viewpoints will include physical as well as logical decomposition, if deployment locations are important as well as structures, and document how these distinctions are to be represented’;
- c) create for the project a Profile (as in figure C-1) that defines the formal names for the decomposition hierarchy and, where useful, the set of color codes that will help make the diagrams more meaningful;
- d) if working at a Systems-of-Systems level, or doing Trade Space studies, consider how to partition these different aspects in the model and build a model decomposition framework that will support exploration of these different aspects.

As noted in annex B there are a number of RASDSv2 viewpoints where existing SysML drawing styles can be adapted and configured to provide suitable templates. Activity diagrams are equally applicable to Operations Views and Functional Views. State Charts may be applied to describe functional behavior or protocol behavior. Various kinds of composition diagrams can be used to describe systems elements, whether they are software, or communications hardware, or structural parts. A modeling Profile is needed to clarify the required mapping of different Viewpoint representations and object types to SysML drawing types.

All of the modeling and examples included here were done in SysML V1 because that is version of SysML that is, at the time of publication, the most widely available language with tool support. It is worth noting that OMG Systems Modeling Language™ (SysML®) SysML V2 (reference [48]) is now in development and that tools that support it are becoming available. SysML V2 will support all of the features of V1, and much more. As the current draft states:

It provides the capability to create and visualize models that represent many different aspects of a system. This includes representing the requirements, structure, and behavior of the system, and the specification of analysis cases and verification cases used to analyze and verify the system. The language is intended to support multiple systems engineering methods and practices. The specific methods and practices may impose additional constraints on how the language is used.

Some studies are now underway to re-create the interface modeling patterns (reference [28]) introduced in this section using SysML V2. What seems clear at this point is that this newer

modeling framework has all of the expressive power of SysML V1, and the advantage that it brings completely idempotent approaches for communicating the semantic, syntactic, and visual aspects of models, something that SysML V1 interchange formats were not capable of.

SysML V2 does support viewpoint and view constructs, but, similar to SysML V1, does not provide anything like a standard set of these. As a result, the same sorts of adaptations that were just described will also be required of a SysML V2 modeling tool and framework in order to support RASDSv2 systems architecture modeling. SysML V2 does include a concept documented as a ‘Domain Library’, which appears as though it will provide direct support for the Physical Viewpoint in the form of a ‘*Geometry Domain Library*’. Other domain libraries can be defined, and this appears to be the ‘hook’ upon which a set of RASDSv2 Domain Libraries might be hung.

C11 AVAILABLE TOOLS TO SUPPORT RASDSV2 MODELING

C11.1 INTRODUCTION

In an ideal world, CCSDS would have the resources to produce an open-source modeling tool that is already adapted to allow formal models complying with RASDSv2 to be produced. Unfortunately, this is not the case, but there are a couple of modeling tools and frameworks that may be close enough for some purposes. The two that appear, at the time of writing, to have the greatest potential are the Unified Architecture Framework (UAF), (<https://www.omg.org/uaf/>) and the ARChitecture Analysis and Design Integrated Approach (Arcadia) MBSE ‘tooled method’ offered as open source in <https://eclipse.org/capella>.

C11.2 UNIFIED ARCHITECTURE FRAMEWORK

The UAF (reference [32]) is an architecture framework that is an evolution of the earlier DoDAF (reference [11]) and other modeling frameworks. Quoting from the OMG Web site:

UAF defines ways of representing an enterprise architecture that enables stakeholders to focus on specific areas of interest in the enterprise while retaining sight of the big picture. UAF meets the specific business, operational and systems-of-systems integration needs of commercial and industrial enterprises as well as defense organizations.

The focus in UAF is on Enterprise level architectures and on supporting acquisition, rather than a focus on the architecture-to-engineering transition. The framework does include a quite complete Domain Model that starts at the Enterprise viewpoint and then maps to a series of Capabilities tied to Services, Operations, Resources (real elements), Personnel, and Security.

From the RASDSv2 point of view the elements that are provided, and the likely tooling support, would appear to provide at least partial support for the Enterprise, Functional, Connectivity, and Operational Viewpoints. What is missing from UAF is the

communications aspects of the Connectivity viewpoint, the Protocol Viewpoint, the Physical/Structural Viewpoint, and a focus on the interactions of all these elements with the environment. There is an Information Model that is the owner of Resource and Operations Information, but it does not appear to have a central role of its own nor a distinction between abstract and realized types of data. Interfaces for Resources and Operations are identified, but there do not appear to be strong modeling constructs offered to model them down to implementation details.

Security plays an important role in UAF and it has its own viewpoint, tied to processes and mitigations. Risks are modeled as affecting Operational and Resource assets, and compliance with Controls are modeled. This appears to place security considerations somewhat ‘on the side’ as opposed to embedding these considerations within each viewpoint. This aligns well with published NIST practices, but may leave important aspects, which can be more directly explored within each viewpoint, out of consideration.

UAF is standardized in the OMG and, at time of writing, it is supported by a number of the major SysML tool vendors. Where the focus is more on Enterprise considerations (what needs to be done) and less on how to build it (how it works and how to engineer it), UAF may provide a really useful modeling environment.

C11.3 ARCADIA FRAMEWORK AND CAPELLA TOOL

Arcadia (reference [49]) is described as a ‘tooled method devoted to systems & architecture engineering’. The Arcadia method is supported by the Capella modelling tool (reference [50]), which runs inside the Eclipse open-source environment. The core set of tools is available in an open-source form under the Eclipse Public License (EPL), and there are other open-source and commercial add-ons available. As such, it appears to provide a rather rich ecosystem for certain kinds of architecture modeling.

There are many parallels between Arcadia and RASDSv2, both in the overall metamodels and in the intended use. Both use a very similar set of viewpoints: Operational Analysis (Operational VP and some aspects of Enterprise VP); Functional & Non-Functional Need (Functional VP); Logical Architecture (functions mapped to abstract systems elements in Connectivity View); and Physical Architecture (functions mapped to actual hardware and software elements in Connectivity VP).

Many of the terms that are used in Arcadia can be traced back to ISO 42010 (reference [14]), but there is no stated attribution. And, like RASDSv2, the specific set of viewpoints that are used can be traced back to RM-ODP (reference [1]), but there is also no attribution provided. One consequence of this is that an extremely useful feature documented in ISO 42010, RM-ODP, and RASDSv2, that of correspondence between related items in different viewpoints, appears to be missing. There are hints that these relationships are understood, but it is nowhere clearly documented. Data appears only as an ‘Exchange Item’ that is carried by operational and functional exchanges. The relationships that exist between abstract functions and implemented ones show up in sketches, but not in any metamodel formalisms.

In a similar way, the handling of communications is weak. There are interfaces between components, but the definitions of physical link, communications, and interfaces are all very generalized, and there is no concept of protocols, protocol stacks, or service interfaces. So none of the features that RASDSv2 supports in the Connectivity Viewpoint or the Protocol Viewpoint are supported by Arcadia. Interactions with the environment are similarly limited and there is no support for the Physical/Structural Viewpoint. Security concerns are likewise left out of consideration entirely.

All of those concerns aside, because it exists in an accessible, open-source form, the Arcadia/Capella tool set may provide a useful and pragmatic environment within which to do systems architecture and engineering, at least up to the point at which the tool, as implemented, provides features that meet the needs. But in the space data systems domain, as we approach it in RASDSv2, it seems clear that the limitations of the tool will show up. That said, it might provide a great platform on which to build the needed extensions to accommodate the other concepts that RASDSv2 deals with. The document set is really quite extensive and it does include a mapping of Arcadia to SysML (reference [50]), which, along with the Arcadia Metamodel (reference [49]) will provide useful leverage to use to build any extensions.

ANNEX D

ARCHITECTURE, MODELS, METAMODELS, AND FRAMEWORKS

(INFORMATIVE)

D1 INTRODUCTION

Most of the published systems engineering documents that have been developed by different agencies use the term ‘systems architecture’ without ever really defining what one is nor what a good one might look like. An example of this can be seen in the latest NASA Systems Engineering Handbook (reference [34]):

The system architecture can be seen as the strategic organization of the functional elements of the system, laid out to enable the roles, relationships, dependencies, and interfaces between elements to be clearly defined and understood. It is strategic in its focus on the overarching structure of the system and how its elements fit together to contribute to the whole, instead of on the particular workings of the elements themselves. It enables the elements to be developed separately from each other while ensuring that they work together effectively to achieve the top-level (or parent) requirements.

Much like the other elements of functional decomposition, the development of a good system-level architecture is a creative, recursive, collaborative, and iterative process that combines an excellent understanding of the project’s end objectives and constraints with an equally good knowledge of various potential technical means of delivering the end products.

But at that point the user is left hanging, because the document contains no well defined ‘system architecture’ methodology nor modeling process.

Logical and data architecture models are an intended outcome of these SE processes, but there is no stated process or conformance criteria for such a set of products. By contrast, The Open Group Architecture Framework (TOGAF) (reference [12]), shows several stages of architecture development (Vision, Business, Information Systems, and Technology architectures) all feeding into a central Requirements Management process and then the design and development processes.

This document provides both a systems architecture methodology and an abstract model that is grounded in best current practices in the field. The fundamental motivation for such an approach is well stated in this quote:

Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer, or cheaper than reality instead of reality for some purpose.

A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.

—Rothenberg, Jeff. ‘The Nature of Modeling’, RAND,
<https://www.rand.org/content/dam/rand/pubs/notes/2007/N3027.pdf>

D2 RASDSv2 AND ISO 42010

ISO has published an abstract systems architecture method, ISO 42010-2022 (reference [14]), *Software, Systems, and Enterprise—Architecture Description*, that is based on the earlier IEEE-P1471-2000, the Recommended Practice for Architectural Description of Software Intensive Systems (reference [2]). ISO 42010 is a ‘meta-architecture’ document that describes ‘core terms, definitions, and relationships for the Architecture Description’, describes best practices for defining software architecture descriptions, and defines the meaning of viewpoint and view.

RASDSv2 utilizes concepts and terms defined in ISO 42010. The fundamental concepts defined in ISO 42010 are captured in figure D-1 (reference [31]).

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

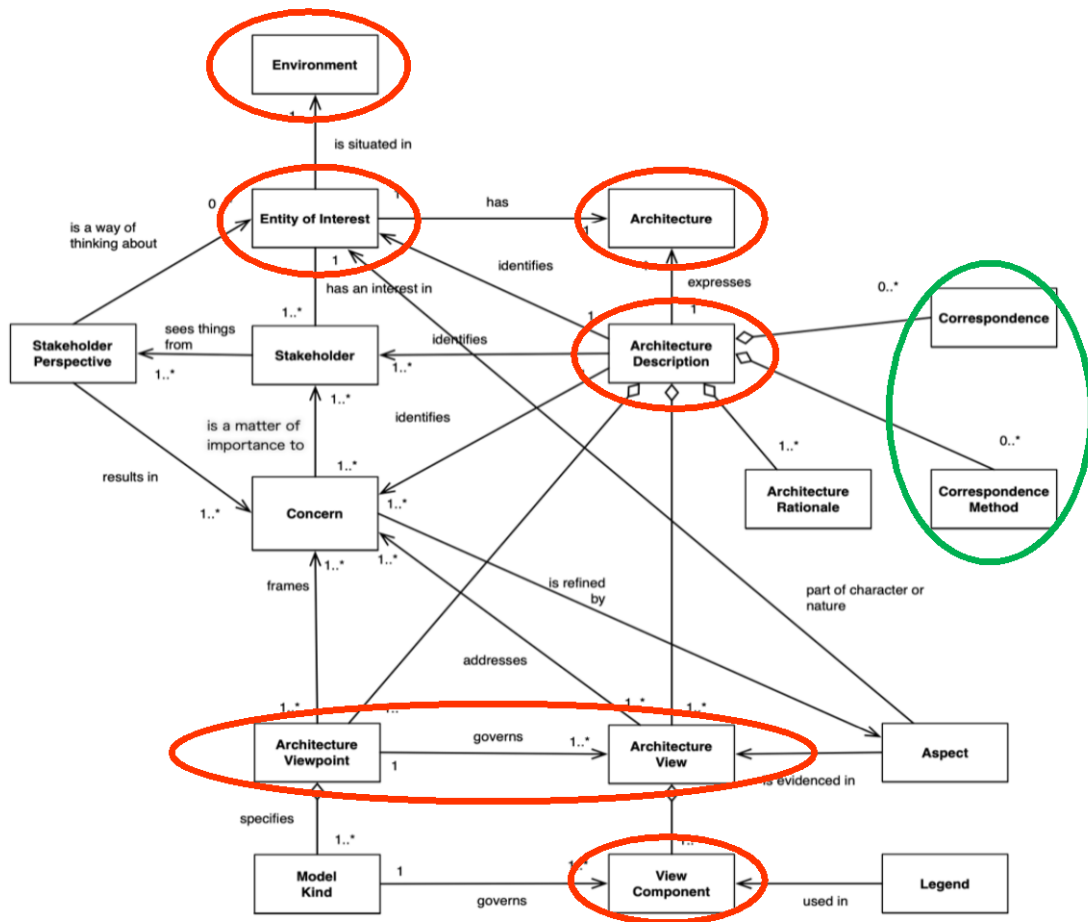


Figure D-1: ISO 42010 Fundamental Concepts²³

The objects highlighted in red and green are the ones that are focused on the most in RASDSv2, which uses these metamodel concepts to create a Reference architecture with a specific set of viewpoints and views suitable for describing the architectures of space systems. Figure D-2 provides a view of how these different ‘meta model’ levels relate, and how they relate to a specific project model that uses this methodology.

²³ Copyright 1998-2016 by Rich Hilliard, <http://www.iso-architecture.org/42010/cm>; used with permission.

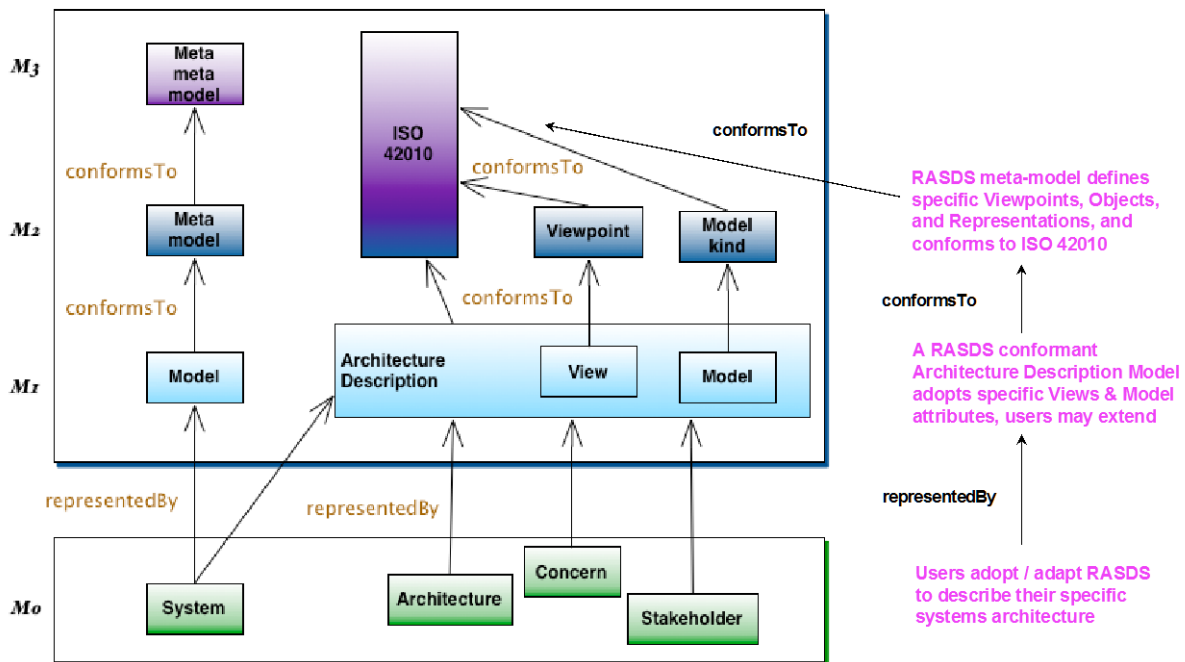


Figure D-2: RASDSv2 Related to Metamodel Levels²⁴

As can now be understood, the RASDSv2 metamodel (M2) defines a set of viewpoint specifications and a set of objects and representations. The RASDSv2 metamodel leverages the ISO 42010 (M3 and M2) definitions for all these fundamental viewpoint/view terms and provides specific guidance on viewpoint specifications suitable for the space system domain.

A user of RASDSv2 defines those parts of the Architecture Description Framework (M1) that they are going to use to create their model. The users of this adapted M1 model framework can then use the defined views to model their specific system (M0). A recent paper, Toward Systems Engineering Meta-Methodology (reference [33]), explores in some depth the current work on modeling meta-methodology, of which this work is an example.

D3 VIEWPOINTS AND VIEWPOINT SPECIFICATIONS

Space systems are complex entities that have many different aspects and it is usually impossible to depict these all of aspects with a single view or in a single framework. Therefore the architecture of a space system must often be described from multiple viewpoints, each focusing on different concerns associated with the system. The RASDSv2 reference architecture defines a carefully selected set of viewpoint specifications to describe and analyze architectures of space systems.

²⁴ Copyright 1998-2016 by Rich Hilliard, <http://www.iso-architecture.org/ieee-1471/meta/>; used with permission.

The notion of describing complex space systems from a variety of viewpoints is not a new one, as figure D-3 from the Voyager Project (reference [30]) makes clear. What has changed over time is the formalisms and methods, like ISO 42010 and RASDSv2, that are available to describe these complex architectures and help inform and structure the architecting process.

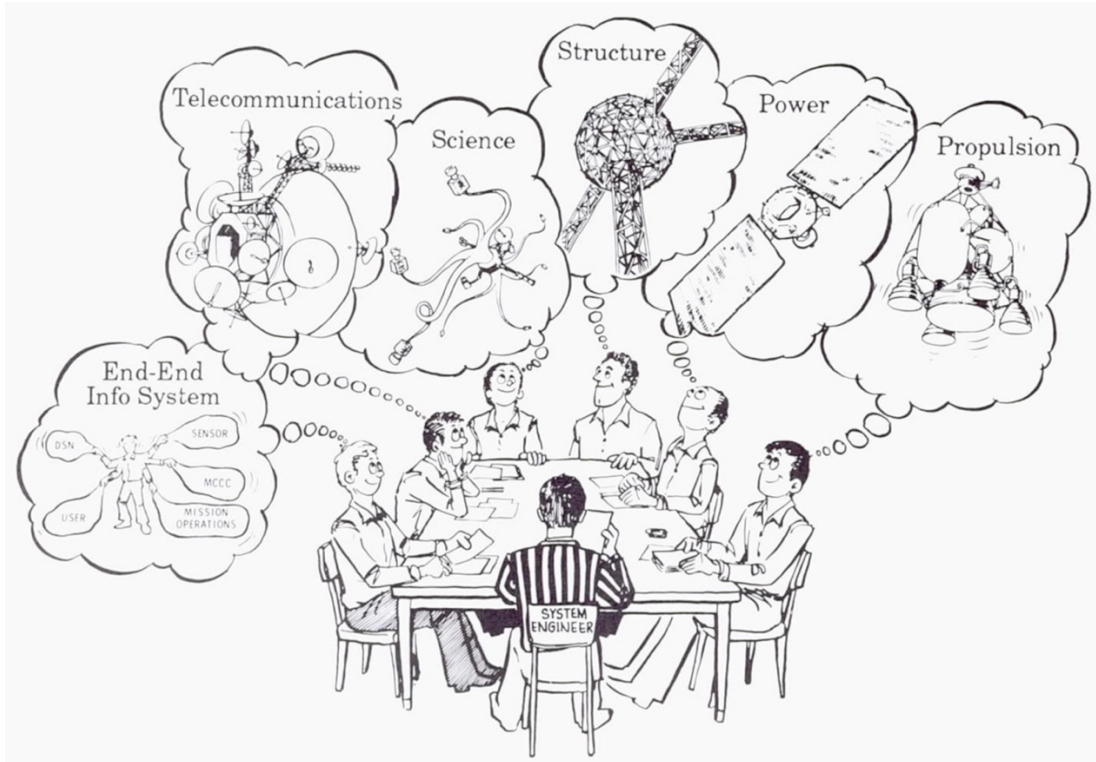


Figure D-3: Different Viewpoints of a Space System²⁵

A viewpoint specification is an abstraction that uses a selected set of architectural concepts and structuring rules to focus on particular concerns within a space system. Each viewpoint documents a different set of design concerns and issues, and each provides the means for reasoning about those aspects of the system. Each viewpoint is intended to be orthogonal to the others, but a specific modelling method has been defined to allow related elements appearing in different viewpoints to be connected. This is called correspondence, and it is the topic highlighted in green in figure D-1.

Each viewpoint specification defines the methods for describing a space system as a set of objects and the interactions among them. An object is an abstract model of an entity in the system. Objects have type, behavior, and state and are distinct from other types of objects. Objects are defined in their primary viewpoint, but may have corresponding objects that appear in other viewpoints. There are explicit Correspondence relationships that describe this. A viewpoint specification defines the rules for constructing views of the system.

²⁵ From JPL Pub 89-24, “The Voyager Neptune Travel Guide”, Charles Kohlase editor, June 1, 1989 (artist Phil Gwinn).

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

viewpoint specifications are described in terms of the objects that may appear in them, their attributes, and the relationships among them. An object is a representation of an entity in the real world. It contains information, may exhibit behavior, and may offer services. A system is composed of interacting objects. An object is characterized by whatever distinguishes it from other objects and by specialization, encapsulation, abstraction, and behavior. Encapsulation is the property that the information contained in an object is accessible only through interactions at the interfaces supported by the object. Because internal objects are encapsulated, there are no hidden side effects of interactions. That is, an interaction with one object cannot affect the state of another object without some defined secondary interaction taking place with that object. Thus any change in the state of an object can only occur as a result of an external request of an object, an internal action of the object, or an interaction of the object with its environment.

ANNEX E

GLOSSARY AND ACRONYMS

(INFORMATIVE)

E1 ACRONYMS

2FA	two factor authentication
3GPP	3rd Generation Partnership Project
ADF	architecture description framework
AMS	Asynchronous Message Service
ANSI	American National Standards Institute
AOS	Advanced Orbiting Systems
API	application program interface
ASL	application and support layer (architecture)
ATO	authorization to operate
BP	Bundle Protocol (DTN)
BPsec	Bundle Protocol Security
BWEM	bandwidth efficient modulation
C&DH	command and data handling
CCITT	Consultative Committee for International Telephony and Telegraphy
CCSDS	Consultative Committee for Space Data Systems
CDM	continuous diagnostics and mitigation
CEOS	Committee on Earth Observation Satellites
CFDP	CCSDS File Delivery Protocol
CPU	central processing unit
CSTS	Cross Support Transfer Service

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

DDOS	distributed denial of service
DEDSL	Data Entity Dictionary Specification Language
DNS	domain name service
DoD	(U.S.) Department of Defense
DoDAF	DoD Architecture Framework
DoT	dictionary of terms
DSN	Deep Space Network
DTN	delay (and disruption) tolerant network
EMI	electromagnetic interference
EPL	Eclipse Public License
ESLT	Earth-space link terminal
F-CLTU	forward control link transmission unit
FEC	forward error correction
FPGA	field programmable gate array
FTP	File Transfer Protocol
GTN	ground tracking network
H/W	hardware
HTTP	HyperText Transport Protocol
HTTPS	HyperText Transport Protocol Secure
HVAC	heating, ventilation, and air conditioning
IaaS	infrastructure as a service
ICAM	identity, credential, and access management
ICD	interface control document
IDS	intrusion detection system
IEC	International Electrotechnical Committee

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IPSec	IP Security (protocol)
ISO	International Standards Organization
ISO-BRM	ISO/IEC Basic Reference Model
ISP	internet service provider
ITU	United Nations International Telecommunication Union
ITU-T	ITU Telecommunications standardization sector
JTC	Joint Technical Committee (of ISO)
KM	key management
LAN	local area network
MBSE	model based systems engineering
MDSD	model driven system design
MEX	Mars Express
MIB	management information base
MOF	Meta Object Facility
MOS	mission operations system
MRO	Mars Relay Orbiter
MSL	Mars Science Lander
MTS	message transfer service
NIST	National Institute of Standards and Technology
OAuth	open authorization
ODP	open distributed processing
OMG	Object Management Group
OS	operating system

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

OSCP	Overview of Space Communication Protocols
OV	Operational View (of DoDAF)
OWL	Ontology Web Language
PaaS	platform as a service
PDU	protocol data unit
PKI	public key infrastructure
RASDS	Reference Architecture for Space Data Systems
RASIM	Reference Architecture for Space Information Systems
RCF	Return (Virtual) Channel Frames
REST	Representational State Transfer
RF	radio frequency
RFC	Request for Comment (Internet standards)
RM-ODP	Reference Model-Open Distributed Processing
RMP	registry management policy
ROM	read only memory
RTL	round-trip light time
SaaS	software as a service
SAML	Security Assertion Markup Language
SANA	Space Assigned Numbers Authority
SAP	service access point
SAWG	System Architecture Working Group
SBU	sensitive but unclassified
SC	Subcommittee (of ISO)
SCCS	Space Communication Cross Support
SCCS-ARD	Space Communication Cross Support—Architecture Requirements Document

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

SDLS	Space Data Link Security
SDU	service data unit
SE	Systems Engineering
SIEM	security information and event management
SLA	service level agreement
SLE	Space Link Extension
SOA	service oriented architecture
SOAML	service oriented architecture modeling language
SOE	sequence of events
SOP	standard operating procedure
SOS	system of systems
SPP	Space Packet Protocol
SSI	Solar System Internetwork
SV	System View (of DoDAF)
S/W	software
SysML	System Modeling Language
TC	telecommand
TCP	Transmission Control Protocol
TOGAF	The Open Group Architecture Framework
UAF	Unified Architecture Framework
UI	user interface
UML	Unified Modeling Language
UPMS	UML Profile and Metamodel for Services
USLP	Unified Space Link Protocol
VP	viewpoint

VPN	virtual private network
W3C	World Wide Web Consortium
WAN	wide area network
WG	working group
WiFi	wireless fidelity, trademarked by IEEE 802.11x
WSDL	Web Services Description Language

E2 TERMS

abstraction: A mechanism and practice to reduce and factor out details so that one can focus on few concepts at a time. It is the process of extracting the underlying essence of a concept, removing any dependence on real world objects with which it might originally have been connected, and generalizing it so that it has wider applications. [3.2.2, 5.5.2]

abstract data architecture metamodels: Models for specification and standardization of data elements (e.g., ISO/IEC 11179, DEDSL). [10.5.2]

action: Something that happens within an object, either with or without participation of another object. An interaction is an action performed with participation of another object. [3.2.3, 12.5.2]

activity: A specification of behavior described as a sequence of actions. [3.2.3, 12.5.2]

aggregation: Several things grouped together or considered as a whole: the act of gathering things together. [3.2.4]

allocation: A mapping between one set of model elements and another. The mapping is often performed as part of the design process to refine the design. Typical examples of allocation include allocation of functions to nodes, logical to physical components, logical to physical links, and software to hardware. [7.5.2]

application: One or more pieces of software designed to perform some specific function; it is a configuration of interacting implemented software engineering objects. [7.5.2]

application programming interface, API: A set of definitions of the ways one piece of computer software communicates with another. It is a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software. [9.6.2]

architecting: The process of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture. It is both a science and an art. [3.2.5]

architecture: The concepts and rules that define the structure, semantic behavior, and relationships among the parts of a system; a plan of something to be constructed. It includes the elements (entities) that make up the thing, the relationships among the elements, the constraints that affect those relationships, a focus on the parts of the thing, and a focus on the thing as a whole. [3.2.5]

architecture description: A work product used to express an Architecture. It may contain stakeholders, concerns, viewpoint specifications, viewpoints, views, correspondences, representations, and other elements. [3.2.5]

artifact: Any tangible objects made, modified, or used by people, or produced during system design, development, testing or operations. [10.5.2]

aspects: A set of characteristics or features of the entity of interest in its environment to address concerns within an Architecture Description. [3.2.5]

asset: A person, data, or other resource that is valued by an organization. [4.5.2]

attribute: A characteristic of an object; a language construct that system designers use to add additional information to system elements (e.g., objects, modules, types) to define their functionality. [3.2.3]

behavior: A set of actions performed by an object for some purpose. [3.2.3, 5.5.1, 11.5.1]

capability: The ability to achieve a desired outcome under specified conditions using a combination of activities and resources to satisfy a stakeholder need. [4.5.1]

center of mass: The location of the balance point of a Structural Object in the coordinate system of the object. [8.5.2]

Communications Viewpoint: An engineering and technology view on a space system that focuses on the protocols and mechanisms of information transfer performed by that system. [9.3]

community: An entity (e.g., Earth Science) that may exist within one Space Enterprise or across multiple Space Enterprises. It is distinguished by being bound by common objectives and relationships and offers a set of resources that are sharable within the Community and with other Communities. [4.5.2]

component: A physical entity operating in a physical environment. A component is a configuration of engineering objects forming a single unit for the purpose of location in space, and embodying a set of functions. A component has some well-understood, possibly rapidly moving, location, and it may be composed of two or more (sub)components. [6.5.2]

composite object: An object composed of two or more objects via aggregation. The behaviors of the composite object are determined by those of the objects that it aggregates. [3.2.4]

composition: A form of aggregation. Composition may be recursive. [3.2.4]

concerns: Those interests which pertain to the system's development, its operation, or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability. [3.2.5]

configuration: A collection of objects able to interact at interfaces. A configuration determines the set of objects involved in each interaction along with constraints on their interactions. [3.2.3, 6.5.1, 7.5.1]

connections: The types of connections and the sub-components that they connect. Thermal conductivity, wiring harness, etc. [6.5.1]

Connectivity Viewpoint: An engineering viewpoint on a space system that focuses on the node and link view of a system, the physical connections among nodes, their physical and environmental constraints, physical dynamics, and (optionally) the allocation of implemented functions to nodes. [7.3]

connector: A thing which links two or more things together. A connector may be rigid, flexible, hinged, rotational, articulated or simply energetic. Connectors connect components at a port. [4.5.2]

constraint: A limitation or implied requirement that limits the design solution or implementation, is not changeable by the enterprise, and is generally non-allocable. [3.2.3, 7.5.1, 12.5.1, 6.5.1]

contract: An agreement that specifies certain legally enforceable rights and obligations pertaining to two or more parties. A contract typically involves the transfer of goods, services, money, or a promise to transfer any of those at a future date. [4.5.1]

correspondence: An identified relationship from an element in one viewpoint to a related element in another viewpoint. It may be used to express a wide range of relationships, such as equivalence, transformation, composition, refinement, consistency, or constraint. [3.2.5]

cross support: An agreement between two or more organizations to exploit the technical capability of interoperability for mutual advantage, such as one organization offering support services to another in order to enhance or enable some aspect of a space mission. [4.3]

data: A representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means. [10.5.1]

data architecture: Models of the structure and relationships among the data elements used within a system. [10.5.2]

data element: A basic unit of information that has a unique meaning and subcategories (data items) of distinct value. Examples of data elements include gender, race, and geographic location. [10.5.1]

data model: Schema and Structure Definitions. [10.5.2]

data objects: Information Objects, either physical or digital. [10.5.2]

data structure: A data organization, management, and storage format that is usually chosen for efficient access to data. [10.5.1]

data semantics: Information that defines the meaning rather than the physical representation of data. Semantics potentially cover a very large domain, from the simple domain, such as the units of one data entity, to a more complex one, such as the relationship between a data entity and another. [10.5.1]

domain: A Community that is under single organizational, administrative, or technical control (e.g., NASA Space Operations Mission Directorate). A domain may have resources, policies, access control, and possibly quality of service constraints. A Domain may be subdivided into Subdomains. Multiple Domains may be collected into a Federation. [4.5.2]

element: A constituent part of something; any thing that is one of the individual parts of which a composite entity is made up; an identifiable component, process, or entity of a system. [3.2.2]

engineering object: An implementation or realization of some abstract function. It may be implemented as hardware (node) or as software (application or software component). [6.5.1]

Enterprise Object: An organizational entity that is governed by a single authority that has its own objectives and policies to operate the object. An Enterprise Object may be a component of another larger Enterprise Object. Enterprise Objects may participate wholly or in part in other Enterprise Objects. [4.3]

Enterprise Viewpoint: A view of a space system that focuses on the community, purpose, scope, and policies for that system. This viewpoint includes organizations as well as the Enterprise Objects that have assigned roles, responsibilities, and interactions. [4.3]

entity: Any concrete or abstract thing of interest. For example, an entity may be a physical instrument, a computer, a piece of software, or a set of functions performed by a system. While in general the word entity can be used to refer to anything, in the context of modelling it is reserved to refer to things in the universe of discourse being modelled. [3.2.2]

environment: A complex of external factors that acts on a system and determines its course and form of existence. An environment may be thought of as a superset, of which the given system is a subset. An environment may have one or more parameters, physical or otherwise. The environment of some system or object consists of the substances, circumstances, objects, or conditions by which it is surrounded or in which it occurs. [3.2.4]

event: Any observable system or natural occurrence. The fundamental entity of observed physical reality represented by a point designated by three coordinates of place and one of time in the space-time continuum postulated by the theory of relativity. [12.3]

facility: A physical infrastructure element that supports the use of services and other resources. [4.3]

federation: A Community consisting of multiple Domains (e.g., CEOS or CCSDS) that come together to share resources while retaining their autonomy over those resources. Federations are bound by negotiated agreements. A Federation may include only some members of a Domain or Subdomain (e.g., a particular Earth Observing project). Members of a Federation agree to rules for sharing resources and for joining and/or leaving the Federation. [4.5.2]

firmware: Software that is contained in a read-only memory (ROM) device. It is typically treated as software unless there is a reason for showing the hardware component itself. [6.5.1]

flows: Representation of the movement of data, substance, or energy. Flows of data are shown using named Data Objects. The formal definitions will be found in Information views. Flows of matter or energy may appear in the Structural Viewpoint. [8.5.3, 12.5.1]

function: The set of actions or activities performed by some object to achieve a goal. The transformation of inputs to outputs that may include the creation, modification, monitoring, or destruction of elements. [3.2.4]

Functional Object: An object that performs functions to achieve a goal of a space system, or to support actions of another Functional Object and to transfer, generate, or process data in performing those actions. [5.3]

Functional Viewpoint: A view on a space system that focuses on the structure of the functions performed by that system and their behavior and on the interactions among the functions. This includes Functional Objects, the logical connections between them, their interactions, and logical interfaces. [2.3.2, 5.3]

goal: An aim or purpose; the end toward which effort is directed. Goals tend to be broad or abstract and to state general intentions. [3.2.4]

hardware: The mechanical, magnetic, electrical, and electronic devices or components of a system used for producing, collecting, processing, storing or transporting data. [6.5.1]

inertia matrix: The moments of inertia of a Structural Object arranged in an array. [8.5.2]

information: Any communication or representation of knowledge such as facts, data, or opinions in any medium or form, including textual, numerical, graphic, cartographic, narrative, or audiovisual. [10.3]

Information Management Functional Objects: Active functional elements that support the location, access, delivery, and management of passive Information Objects. These Information Management Functional Objects are a class of Functional Objects. [5.5.2]

Information Object: Data, along with the necessary structure and syntax to allow interpretation and use of that data; may also have associated *metadata*, including the relationships among Data Objects, rules for their use and transformation, and policies on access. [10.3]

Information Package: A primary Information Object, optional ancillary information, and associated supporting information that is needed to use the Information Object. The Information Package has associated Packaging Information used to delimit and identify the primary Information Object and Supporting Information. [10.3]

Information Viewpoint: A view of a space system that focuses on the information used by that system. This includes structural (syntactic) and semantic views of the information, the relationships among information elements, and rules for their management and transformation. [10.3]

instantiation: Creation of an instance of some abstract element, achieved by an action of an object in the model. Elements can be anything that can be instantiated, in particular objects and interfaces. Data models must be instantiated as real Information Objects in order to participate in system activities. [10.5.2]

interaction: An action performed by an object with participation of another object or with its environment. [3.2.3, 12.5.2]

interaction modes: Request-acknowledgement-response forms among Operations elements. These may include request and immediate response, or request with immediate acknowledgement (and later response), or even a request followed some time after with a response or acknowledgement. [12.5.1]

interface: A set of interactions performed by an object for participation with another object for some purpose, along with constraints on how they can occur. An interface is therefore where the behavior of an object is exposed. Objects may have one or more interfaces. [3.2.3]

interoperability: The technical capability of two or more systems or components to exchange information and to use the information that has been exchanged. Multiple degrees of interoperability are possible, ranging from basic Physical Layer (e.g., frequency, modulation, and coding) compatibility up to full Application Layer information exchange. [3.2.3]

knowledge: Facts, information, and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject. [10.3]

knowledge model: A representation of knowledge in a form that can be interpreted by both humans and machines and is used in knowledge-based systems. [10.5.2]

link: The locus of relations among nodes. It provides interconnections between nodes for communication and coordination. It may be implemented by a wired connection or with some RF or optical communications media. It may periodically become inactive because of the motion of a node or the lack of availability of communications resources, for example, links connect nodes at a port. [7.5.2]

location: A point or extent in space. [3.2.4, 8.5.3]

logical link: The locus of relations among logical objects. It may be considered separately from any particular implementation or deployment and has no physical manifestation except as part of a model. [3.2.4]

logical object: An abstract entity that may be considered separately from any particular implementation or deployment. It has no physical manifestation except as part of a model, but it may have associated behaviors and interfaces. [3.2.4]

mass: The inertial property of a Structural Object. [8.5.2]

mass properties: Center of mass, inertia matrix [6.5.1]

metadata: ‘Data about data’; the information that describes content. It is information about the meaning of data, as well as the relationships among Data Objects, rules for their use and transformation, and policies on access. [10.3]

metamodel: An explicit model of the constructs and rules needed to build specific models within a domain of interest. [10.5.2]

mission operations service, MOS: A suite of end-to-end application-level services that constitute an SOA for space mission operations. [11.5.2]

model: A formal specification of the structure and/or function of a system. All models are abstractions; abstraction is the suppression of irrelevant detail. [3.2.5]

(N)-layer: Any specific layer in a multi-layer protocol stack. The layer above is called the **(N+1)-layer**, the layer below is called the **(N-1)-layer**. This notation is also used for other concepts in the model which are related to these layers, for example **(N)-protocol**, **(N+1)-service**. [9.6.2]

network address: An identifier for a node or host on a telecommunications network. Network addresses are designed to be unique identifiers across the network, although some networks allow for local, private addresses, or locally administered addresses that may not be unique. Special network addresses are allocated as broadcast or multicast addresses. These too are not unique. [7.5.1]

node: A model of a space system physical entity operating in a physical environment. A node is a configuration of engineering objects forming a single unit for the purpose of location in space, and embodying a set of processing, storage, and communication functions.

A node has some well-understood, possibly rapidly moving, location, and it may be composed of two or more (sub)nodes. [7.5.1]

object: An abstract model of an entity in the real world, containing information, having behavior, and offering services. A system is composed of interacting objects. An object is characterized by that which makes it distinct from other objects. [3.2.2]

objective: Something to be done or achieved. Objectives tend to be precise, tangible, and concrete. [3.2.4]

ontology: Representation, formal naming, and definitions of the categories, properties, and relations between the concepts, data, or entities that pertain to one, many, or all domains of discourse. More simply, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject. [10.3]

operations concept: A verbal or graphic statement, in broad outline, of assumptions or intent regarding the operation of the system. The concept of operations frequently is embodied in observing plans and operations plans. The concept is designed to give an overall picture of the operation of the system. [4.5.2]

organization: A formal group of people with one or more shared goals. [4.5.2]

orientation: The rotation of a Structural Object from alignment of its coordinate system with the coordinate system of the assembly that contains the object. [8.5.3]

ownership: Having administrative and fiscal responsibility for the owned element and the right to exclusively control and use that which is owned for one's own purposes. The state or fact of having exclusive possession or control of some object, facility, intellectual property, or some other kind of property. [4.3]

perspective: In systems architecture, the choice of a context or a reference (or the result of this choice) from which to describe, categorize, explain, or codify system design, typically for comparing with another. [3.2.5]

physical environment: Limits on environmental properties, such as pressure, ambient temperature, etc. [6.5.1]

plan: The (acceptable) balance of risk vs. result in generating an operational product (e.g.: nominal or off-nominal plans). [12.5.2]

policy: A set of guidelines and constraints on the behaviors exhibited by the objects in the system. [3.2.5]

port: The physical element of a node where a link is connected. Nodes may have one or more ports. [7.5.1]

process: What needs to happen in the plan to generate an operational product. [12.5.2]

protocol: A set of rules and formats (semantic and syntactic) used to determine the communication behavior of (*N*-layer)-protocol-entities in the performance of (*N*-layer)-functions; the description of the state machines within a Protocol Entity and the PDUs that are exchanged between these entities. [9.6.2]

protocol data unit, PDU: A unit of data specified in an (*N*-layer)-protocol, consisting of (*N*-layer)-protocol-control-information and possibly (*N*-layer)-user-data; the actual Data Objects that are exchanged between peer protocol entities. [9.6.2]

Protocol Entity: An active element within an (*N*-layer)-communications-subsystem embodying a set of capabilities defined for the (*N*-layer)-layer that corresponds to a specific (*N*-layer)-entity-type (without any extra capabilities being used). Protocol Entities implement protocols. [9.3]

provenance: Documentation of the place of origin, proof of authenticity or record of previous processing. These are valuable pieces of information in the history of an object. [10.5.2]

realization: The act or the condition of becoming real. Abstract data architecture elements must be realized as data models and stored in some sort of repository. [10.5.2]

relationship: The way in which two or more entities can be associated with one another. [3.2.3, 10.5.1]

representation: Some way of organizing, manipulating, presenting, and storing information; a visual or tangible rendering of something. [3.2.5]

requirement: A formal statement of: (1) An attribute to be possessed by the element or a function to be performed by the element. (2) the performance standard for the attribute or function. (3) the measuring process to be used in verifying that the standard has been met. [3.2.5, 4.5.1]

resource: Anything available to a system that can support the achievement of objectives; any physical or virtual element that may be of limited availability within a system. A resource may be shared by more than one activity. In the Enterprise Viewpoint a resource is an entity that has some role, offers services, and performs some action within a system. [3.2.4, 4.3]

risk management: The program and supporting processes to manage information security risk to organizational operations (including mission, functions, image, reputation), organizational assets, individuals, other organizations, and the Nation, and includes: (i) establishing the context for risk-related activities; (ii) assessing risk; (iii) responding to risk once determined; and (iv) monitoring risk over time. [4.5.1]

role: The way in which an entity participates in a relationship; an object's set of behaviors and actions associated with the relationship of that object with other objects. [3.2.3]

scenario: A specific sequence of activities illustrating behaviors. A scenario may be used to illustrate an interaction or an operations concept instance. [4.5.2]

schedule: Temporal context for Plan. (e.g., keeps track of SoE <anomaly vs. ‘as predicted/expected’>). [12.5.2]

schema: An information model defined in a document or a database. The universe of objects that can be described is defined in the schema. For each object class, the schema defines what attributes an instance of the class must have, what additional attributes it may have, and what object class can be a parent of the current object base. [10.5.2]

semantics: Rules by which syntactic expressions are assigned meaning. [3.2.3]

sequence of events, SOE: A number of events or activities that come one after another in a particular order. The predicted order of events during spacecraft operations, and also the observed order of events during spacecraft operations. [12.5.2]

service: A provision of an interface of an object to support actions of another object. [3.2.3, 11.5.2]

service access point, SAP: The point at which (N -layer)-services are provided by an (N -layer)-protocol-entity to an ($N+1$ -layer)-protocol-entity. [9.6.2]

service data: A generic term for the kinds of data provided by a specific service. There are many different kinds of service data, which may be a discrete Data Object, a stream of Data Objects (that could be turned into audio or video), or other forms such as a file or a message. [11.5.1]

service interface: A mechanism to enable access to a set of one or more functions of an element, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. [3.2.3]

service provider: The role played by a physical, functional, or organizational entity that provides a cross support (or other) service for a service user. [11.5.2]

service system: The set of hardware and software components used to implement a service in a real system. Service systems may be implemented using one or more hardware and software components. [11.5.2]

service type: Any one of: network service, Data Link Layer service, cross support service, mission operations service, web service, name service, or any of many other types of defined services. [11.5.1]

service user: The role played by a physical, functional, or organizational entity that uses a cross support (or other) service provided by a service provider. [11.5.2]

software or computer programs: The components of information systems that provide operating instructions for specific task based applications that run on computing hardware. [6.5.1]

space enterprise: A top-level autonomous entity (e.g., NASA) that is dedicated to the exploration and/or exploitation of space. It has its own objectives, resources, and policies, and it is not a component of any other Space Enterprise. [4.5.2]

space environment: Conditions in space that affect the design and operation of spacecraft. Effects on spacecraft can arise from temperature extremes, radiation, space debris, and meteoroid impact, upper atmospheric drag, spacecraft electrostatic charging, and gravity. [7.5.1]

Space Link Extension service, SLE service: The set of services that extend one of the CCSDS Space Link Subnetwork services, providing access to the ground termination of that service from a remote ground-based system. An SLE service supplies or consumes one or more channels of the same Space Data Channel type. [11.5.2]

spacecraft: Spacecraft that travel in space, rovers, habitats, and other element in space or on a remote planetary surface. [4.5.2]

specification: A set of requirements or other descriptive information for a system or classifier. [3.2.5]

stakeholder: An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system. [3.2.5]

standard: A formal specification that defines and governs functions and protocols at interfaces of a data system. It describes in detail the technical capabilities of, and establishes the requirements to be met by, interfacing subsystems to achieve compatibility and interoperability. [3.2.5]

state: A condition or situation during the life of some object; at a given instant in time, the condition of an object that determines the set of all sequences of actions in which the object can take part. [9.6.2, 12.5.1]

state machine: The description of the discrete sequence of states that an object or interaction goes through during its life in response to events, together with its responses and actions. [9.6.2]

state table: An alternative tabular representation of the same information. [9.6.2]

stress: A measure of forces (internal or external) acting over some cross sectional area of an object. [8.5.3]

structure: The relationship between a set of elements, contributing to the properties of the whole and enabling them to interact. [3.2.3]

syntax: The grammar defining the valid set of symbols and well-formed linguistic constructs of a language. [3.2.3]

system: A set of elements (people, products [hardware and software], facilities, equipment, material, and processes [automated as well as manual procedures]) that are related and whose behavior satisfies customer and/or operational needs. [3.2.5]

system performance: The amount of useful work accomplished by a system. Outside of specific contexts, performance is estimated in terms of accuracy, efficiency, and speed of executing computer program instructions or transferring data. [7.5.1]

taxonomy, taxonomical classification: A hierarchical classification or categorization system in which all the terms belong to a single hierarchical structure and have parent/child or broader/narrower relationships to other terms. Many taxonomies are hierarchies (and thus have an intrinsic tree structure), but not all are. [10.3]

tracking station: A node in a Connectivity Viewpoint that occupies a fixed location on a planet (including Earth) or asteroid. [7.5.2]

type: The set of values allowed and the primitive operations, which an object can provide. Types are grouped into classes, which share the same primitive operations. [3.2.3]

unique identifier: A value used in specified fields of CCSDS-defined (or other) Data Link Layer data structures. It provides a unique identifier for the node. [7.5.1]

use case: A list of actions or event steps typically defining the interactions between an actor and a system to achieve a goal. A Use Case may describe a situation where a system may be used or a potential scenario in which a system receives an external request and responds to it. [4.5.1]

view: A representation of a whole system from the perspective of a set of concerns. Views are themselves modular and well formed, and each view is intended to correspond to exactly one viewpoint. A view may include representations or correspondences to elements defined in other viewpoints. [3.2.5]

viewpoint: A set of conventions, achieved using a selected set of architectural concepts and structuring rules, for the creation interpretation, and use of an architecture viewpoint to frame one or more particular concerns within a space system. [3.2.5]

viewpoint specification: A form of abstraction achieved using a selected set of architectural concepts and structuring rules in order to focus on particular concerns within a space system. A viewpoint specification defines a pattern or template from which to construct individual views, and it establishes the rules, techniques, and methods employed in constructing a view. [3.2.5]

CCSDS RECOMMENDED PRACTICE—
REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Web service: A software component or system designed to support interoperable machine- or application- oriented interaction over a network. A Web service has an interface described in a machine-processable format, specifically WSDL. [11.5.2]