The Consultative Committee for Space Data Systems

Recommendation for Space Data System Practices

REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

RECOMMENDED PRACTICE

CCSDS 311.0-M-1

MAGENTA BOOK

September 2008

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Practices

REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

RECOMMENDED PRACTICE

CCSDS 311.0-M-1

MAGENTA BOOK

September 2008

AUTHORITY

Issue:	Recommended Practice, Issue 1
Date:	September 2008
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 the *Procedures Manual for the Consultative Committee for Space Data Systems*, and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Space Communications and Navigation Office, 7L70
Space Operations Mission Directorate
NASA Headquarters
Washington, DC 20546-0001, USA

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 *Procedures Manual for the Consultative Committee for Space Data Systems*. 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 addressed to the CCSDS Secretariat at the address indicated on page i.

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

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- 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 e.V. (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.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Sciences (CAS)/China.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status/Remarks
CCSDS 311.0-M-1	Reference Architecture for Space Data Systems, Recommended Practice, Issue 1	September 2008	Original issue

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 SCOPE

This document presents a Reference Architecture for Space Data Systems (RASDS). The RASDS is intended to provide a standardized approach for description of data system architectures and high-level designs, which individual CCSDS working groups may use within CCSDS. This approach is aligned with current practices in the fields of system and software architecture and is specifically adapted for the space domain. While it is intended for use within CCSDS, it is also suitable for use by mission and project design teams to describe system architectures and designs within the space domain. It does not propose any specific formal modeling method or tool, but can be adapted to use methods like UML or SysML.

1.2 PURPOSE

Within CCSDS the RASDS will be used for the following purposes:

a) to establish an overall CCSDS recommended methodology for defining and developing domain-specific architectures;

b) to define a common language, taxonomy, and representation so that challenges, requirements, and solutions in the area of space data 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 Recommended Standards in a consistent way so that any standard can be used with other appropriate standards in a space data system;

e) to provide a framework and guidelines for presenting the Recommended Standards developed by CCSDS 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 individual working groups for the description of data system architectures in any relevant CCSDS documents. The use of RASDS is completely voluntary, and CCSDS architectures are not required to be expressed in RASDS-compliant notation. However, individual working groups who are looking for guidance or good engineering practice are likely to find it useful.

Note that not all Viewpoints are needed for every task. The Viewpoint Specifications and representational methodologies may be used where applicable, but only those Viewpoints that are needed for the specific purpose should be used. In many instances, only the Functional and Connectivity Viewpoints may be needed. The methodology provided in this

document may be used in describing mission space data systems—and system of systems—architectures, but its use is not required.

As a Recommended Practice, this document is not binding upon CCSDS. However, its use is encouraged in all CCSDS documents where system or reference architecture descriptions must be provided, and where the analytical and descriptive methodologies provided in this document are useful. New views using these concepts may be assembled as needed, and alternative representations may be adopted where they improve alignment with current practices in specialized subdomains.

1.4 RATIONALE

A number of different standard methods are currently in use or are being developed for description of software-intensive systems architectures. These include the ISO Reference Model of Open Distributed Systems (RM-ODP, reference [1]), the Recommended Practice for Architectural Descriptions of Software-Intensive Systems (IEEE 1471-2000, reference [2]) and 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 ISO Basic Reference Model (ISO-BRM, reference [14]), and others. All of these share the concepts of developing a consistent set of elements, terminology, viewpoints, views, and specifications with which to describe systems and their architectures.

All of these standard methods typically assume that the elements of these systems are fixed in place and that they are in continuous communication over what are nominally error-free communications channels that suffer only occasional disruptions.

Space data systems violate all of these assumptions.

RASDS, on the other hand, provides guidelines for the description of space data systems that take into account the realities of operating in the space environment. This is a domain-specific architectural approach adapted to the requirements of space data systems. RASDS directly addresses the fact that some elements of these distributed systems will be operated at great distances from one another with one-way light times measured in tens of minutes or hours, not milliseconds. These elements may only occasionally be in contact with one another, typically require use of very expensive and over-subscribed ground communications assets, and are strongly affected by the physical environment in which they have to operate. These environmental issues affect what must be done to provide reliable communications between elements, how control interactions may be designed, and how these systems may be operated.

RASDS introduces a set of conventions for representing space data systems from several viewpoints. These drawing conventions have been chosen to ensure that the diagrams for any View can be unambiguously interpreted. Other representations are possible, and some of the methods mentioned above use different representations. However, this document provides a consistent approach that provides the structure for a full formal representation of

space data systems such as might be developed in a UML or SysML tool. Development of a formal representation using a method such as SysML is the subject of proposed future work. An introduction to this topic is provided in annex B.

1.5 MANDATORY SECURITY SECTION

The CCSDS Management Council (CMC) has mandated that security topics be addressed in all CCSDS normative documents and 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).

This document is a reference architecture framework and does not address these technical security topics directly. However, it provides guidance to CCSDS document (and mission) developers on how to describe all of these security topics in the context of system architecture, and each Viewpoint provides the means to address the security issues that are relevant in that Viewpoint. Security topics pertinent for each Viewpoint are identified and briefly addressed in each section. Detailed explanations of security topics and approaches are treated separately in the Security Architecture and related documents (references [4], [17], and [18]).

NOTES

1 RASDS is consistent with IEEE 1471-2000 (reference [2]), the Recommended Practice for Architectural Description of Software Intensive Systems. Where IEEE 1471-2000 describes best practices for defining software architectures and defines the meaning of Viewpoint and View, it does not provide instances of these. These are to be defined to meet stakeholder concerns in the domain to which this approach is to be applied. The Reference Model of Open Distributed Processing (RM-ODP) takes this a step further and defines a specific set of Viewpoints for the domain of open distributed systems. The RASDS methodology goes further still and extends RM-ODP to provide a concrete description of how to develop architectural descriptions for the space data system domain.

2 Other systems architecture methodologies have also been considered in constructing this Recommended Practice. These include the DoD Architecture Framework (DoDAF), The Open Group Architecture Framework (TOGAF), the Unified Modeling Language (UML) and the System Modeling Language (SysML). UML, and SysML which has been derived from it, provide formalisms that are appropriate for use in the RASDS problem domain. However, even these need to be adapted to define the needed Viewpoints, Views, Objects, relationships, etc., that are relevant to the space data system domain, since these methods do not provide them in their native form.

1.6 DOCUMENT STRUCTURE

Section 2 provides an overview of RASDS in a narrative style and introduces Viewpoints and Views.

Section 3 introduces basic concepts and defines specific terms that are used throughout this document.

From section 4 through 8, RASDS is presented in detail for each of the Viewpoints introduced in section 2, and other special terms are defined as needed.

Section 9 presents some examples of how to extend RASDS, using the basic concepts, to describe new views on a system.

Annexes A-D provide additional information on use of RASDS, its relationship to other methods, including UML, SysML, and DoDAF, and a summary of the terminology and acronyms used in the document.

1.7 REFERENCES

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

[1] *Information Technology—Open Distributed Processing—Reference Model: Architecture*. International Standard, ISO/IEC 10746-3:1996. Geneva: ISO, 1996.

[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*. Draft Report Concerning Space Data System Standards, CCSDS 350.5-G-0. Draft Green Book. Issue 0. Washington, D.C.: CCSDS, n.d.

[5] *Reference Architecture for Space Information Management*. Draft Report Concerning Space Data System Standards, CCSDS 312.0-G-0. Draft Green Book. Issue 0. Washington, D.C.: CCSDS, n.d.

[6] *Unified Modeling Language (UML)*. Version 1.5, formal/2003-03-01. Needham, Massachusetts: Object Management Group, March 2003. <http://www.omg.org/technology/documents/modeling_spec_catalog.htm>

[7] *Unified Modeling Language: Infrastructure*. Version 2.0, formal/05-07-05. Needham, Massachusetts: Object Management Group, March 2006. <http://www.omg.org/technology/documents/formal/uml.htm>

[8] *Meta Object Facility (MOF) Core Specification*. Version 2.0, formal/06-01-01. Needham, Massachusetts: Object Management Group, January 2006. <http://www.omg.org/technology/documents/formal/uml.htm>

[9] *OMG Systems Modeling Language (OMG SysML™), V1.0*. OMG Document Number: formal/2007-09-01. Needham, Massachusetts: Object Management Group, September 2007. <http://www.omg.org/spec/SysML/1.0/PDF>

[10] *UML for Systems Engineering*. Request For Proposal. OMG Document: ad/03-03-41. Needham, Massachusetts: Object Management Group, March 2003. <http://www.omg.org/public_schedule/>

[11] "DoD Architecture Framework (DoDAF)." Version 1.0. Applied Information Technology Center (AITC). <http://www.aitcnet.org/dodfw/>

[12] *The Open Group Architecture Framework (TOGAF)*. Version 8.1 Enterprise Edition. San Francisco: The Open Group, n.d. <http://www.opengroup.org/architecture/togaf/>

[13] "xADL 2.0." *Institute for Software Research (ISR)*. University of California, Irvine. <http://www.isr.uci.edu/projects/xarchuci/>

[14] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1:1994. 2nd ed. Geneva: ISO, 1994.

[15] *Information Technology—Open Distributed Processing—Use of UML for ODP System Specifications*. Committee Draft, ISO/IEC CD 19793. Geneva: ISO, 2005.

[16] *Overview of Space Link Protocols*. Report Concerning Space Data System Standards, CCSDS 130.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, June 2001.

[17] *The Application of CCSDS Protocols to Secure Systems*. Report Concerning Space Data System Standards, CCSDS 350.0-G-2. Green Book. Issue 2. Washington, D.C.: CCSDS, January 2006.

[18] *Security Threats against Space Missions*. Report Concerning Space Data System Standards, CCSDS 350.1-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, October 2006.

[19] *Prioritized Requirements for RASDS*. Draft Record Concerning Space Data System Standards, CCSDS 311.1-Y-0. Draft Yellow Book. Issue 0. Washington, D.C.: CCSDS, May 2006.

[20] P. Shames and J. Skipper. "Toward a Framework for Modeling Space Systems Architectures." In *Proceedings of the Ninth International Conference on Space Operations (SpaceOps 2006)* (Rome, Italy). SpaceOps, 2006. <http://www.aiaa.org/spaceops2006/presentations/57418.ppt>

[21] M. Maier. "Architecting Principles for Systems-of-Systems." *Systems Engineering* 1, no. 4 (February 1999): 267-284.

[22] M. Maier, D. Emery, and R. Hilliard. "ANSI/IEEE 1471 and Systems Engineering." *Systems Engineering* 7, no. 3 (June 2004): 257-270.

[23] P. Kruchten. "The 4+1 View Model of Architecture." *IEEE Software* 12, no. 6 (November 1995): 42-50.

[24] "Rational Unified Process® for Systems Engineering, RUP® SE1.1: A Rational Software White Paper." TP 165A, 5/02. Cupertino, CA: Rational, 2002. <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/TP165.pdf>

[25] *UML Profile and Metamodel for Services (UPMS)*. Revised Submission. OMG document: ad/2007-11-02. Needham, Massachusetts: Object Management Group, December 2007. <http://www.omg.org/cgi-bin/doc?ad/07-11-02>

2 OVERVIEW

2.1 GENERAL

This overview section of the document introduces the basic modeling terminology and concepts in a narrative form. Most of these terms will be familiar, but some have special meanings in RASDS. Please refer to the Glossary, annex D, as needed for specific definitions of terms. This section is not normative.

2.2 VIEWPOINTS AND VIEWPOINT SPECIFICATIONS

Space data systems are complex entities that have many different aspects, and it is not easy to depict these aspects with a single view or in a single framework. Therefore the architecture of a space data system must often be described from multiple Viewpoints, each focusing on different concerns associated with the system. The RASDS reference architecture defines a specific set of Viewpoint Specifications to present architectures of space data systems.

A Viewpoint Specification is an abstraction that uses a selected set of architectural concepts and structuring rules in order to focus on particular concerns within a space data system. Each Viewpoint Specification is intended to be orthogonal to the others, but some specific areas of overlap exist to allow the elements in different Viewpoints to be related. Each exposes a different set of design concerns and issues, and each provides the means for reasoning about that aspect of the system.

Each Viewpoint Specification describes the space data system in question as a set of Objects and interactions among them. An Object is an abstract model of an entity in the system. Objects have behavior and state and are distinct from any other objects. Objects defined in their primary Viewpoint often have corresponding objects in other Viewpoints. A Viewpoint Specification defines the rules for constructing Views of the system.

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 and may offer services. A system is composed of interacting objects. An object is characterized by whatever distinguishes it from other objects and by 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 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 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.

To describe the architecture of space data systems, RASDS defines five Viewpoint Specifications, which are explained in the following subsections. The user may decide which of these Viewpoints to use to describe a particular space data system if that system can

be characterized with fewer than five Viewpoints. Often only two or three Viewpoints are needed to define simple system architectures.

A View is a representation of a specific system from the perspective of a set of concerns.

Views are themselves modular and well-formed; each View is intended to correspond to exactly one Viewpoint and is constructed using the rules defined by that Viewpoint Specification. Sometimes a Viewpoint Specification will contain objects that are representations of related objects in another Viewpoint. The user may also define a new Viewpoint Specification using the basic concepts defined in section 3 if it is impossible to capture all the important aspects of the system with objects described in the five Viewpoint Specifications defined here. Some aspects of a system design may benefit from being examined from two or more Views simultaneously, and some examples of this are provided in 2.9. In many of the examples used in this document, color is used to distinguish different classes of objects. RASDS does not assign any particular meaning to the use of color, and the user is free to employ it wherever it assists in clarifying the representation of the architecture or bringing consistency to a set of diagrams.

NOTE – The following Viewpoint Specifications are derived from RM-ODP Viewpoints (with some modifications), but the Connectivity Viewpoint was newly created in order to address issues and constraints related to the space data system physical environments, which are distinct from those encountered in typical terrestrial distributed systems. Challenges from the physical environment in which these systems operate, particularly the motion, discontinuous connectivity, and extremely distant and broad distribution of physical elements, require specialized protocols and systems design. The RASDS approach also has its focus on describing architecture rather than implementation. As a result, only certain subsets of the Engineering and Technology Viewpoints in RM-ODP are treated by RASDS in the Connectivity and Communications Viewpoints. For those aspects of the RM-ODP Viewpoints that are not treated directly in RASDS, the RM-ODP approaches may be used directly.

2.3 OVERVIEW OF VIEWPOINTS

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 sufficiently independent to simplify reasoning about the complete specification. Mutual consistency among the Viewpoint Specifications is ensured by the architecture descriptions defined by RASDS and the use of a common object model that binds them all together.

The RASDS framework extends the RM-ODP framework and provides five specific and complementary viewpoints on the system and its environment:

– The **Enterprise Viewpoint** focuses on the purpose, scope, and policies for the space data system. It describes 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 the space data system into abstract objects that interact at interfaces. It describes the functionality provided by the space data system, the behavior of the functional elements, and their functional decomposition.

- The **Connectivity Viewpoint**, which is new in RASDS, describes the engineered decomposition of the space data system into components (nodes) that interact across connectors (links). The Connectivity Viewpoint describes the physical aspects of the space data system and the external environment within which it operates, the physical behavior (and motion) of the nodes, and their physical decomposition. The links may be manifestly physical (network or data cables), or they may be more ethereal (RF and optical signals). The Connectivity Viewpoint also addresses the allocation of implemented functions (as engineered software or hardware objects) to these Nodes. Note that RASDS addresses only data system components, but a full space system design would include other classes of components and connectors not addressed here.

- The **Communications Viewpoint** focuses on the mechanisms and functions required to engineer and implement the protocols and communications standards for the space data system, including implementation choices and specifications, and allocation of communications functionality to engineered components of the system. This Viewpoint is a subset of the RM-ODP Engineering and Technical Viewpoints, but it is treated separately in RASDS because it is central to describing how to handle many communication issues in space data systems.

- The **Information Viewpoint** focuses on the kinds of information handled by the system, the semantics of the information, and the interpretation of that information. It describes the information managed by the space data system along with the structure, content, semantics, type, relationships, and constraints on the data used within the system.

Although separately specified, the Viewpoint Specifications are not completely independent; key items in each are related to items in the other Viewpoints.

2.4 ENTERPRISE VIEWPOINT

The Enterprise Viewpoint describes the organizations involved in a space data system and the relationships and interactions among them. It allows description of the complex organizational relationships among the stakeholders: scientists, staff, and contractors that are distributed among multiple organizations (space agencies, science institutes, companies, etc.) in typical space data systems. The stakeholders own, operate, or use the facilities that make up the system: spacecraft, instruments, and various ground systems. The Enterprise Viewpoint is used to address these aspects of space data systems.

The relationships among the organizations are described in terms of their roles; responsibilities and policies of the organizations, and the interactions among the organizations are described in terms of agreements and contracts. Stakeholders may include: funding organizations, developers, designers, operators, maintainers, subcontractors, or users, as well as the architects and system engineers themselves. The Enterprise Viewpoint

may also include stakeholder concerns, such as requirements, operations concepts, scenarios, lifecycle, and mission phases. However, specific formalisms for representing these last aspects are not provided at this time.

The Enterprise Viewpoint also includes facilities that are owned, operated, or used by these organizations. These facilities may be spacecraft of various types, e.g., orbiters, landers, or rovers, used for various activities, e.g., relay communications, mappers, observatories, or sample return missions. Facilities may also include ground station networks, mission and ground system control centers, and a variety of mission and science facilities. Instruments may also be modeled as separate facilities in an Enterprise View, especially when they belong to a different organization from the one that owns the spacecraft that carries them.

The primary elements in the Enterprise Viewpoint are Enterprise Objects and the interactions among them. One type of basic Enterprise Object is an abstract model of an organization involved in a space data system. This Enterprise Object represents an independent Enterprise (such as a space agency, a government institute, a university, or a private company) or a department or an organization of an Enterprise (such as a tracking network team, a control center team, a science team, or a research group). An Enterprise Object may be composed of other Enterprise Objects. A formal or informal group of Enterprise Objects that plays some role in a space data system (such as a community, a committee, or a joint project) can also be an Enterprise Object. Certain information objects, such as contracts, policies, scenarios, may also appear in the Enterprise Viewpoint.

Another primary type of Enterprise Object that may appear in an Enterprise View is a facility that is owned, operated, or used by an organizational Enterprise Object. These objects may also be composed of other Enterprise Objects, and this composition may be shown in an Enterprise View. Typically, however, all of the detailed compositional aspects of the system elements are shown in an engineering view such as a Connectivity View, and only those needed to clarify Enterprise issues, such as roles and responsibilities, are shown in an Enterprise View.

Each Viewpoint Specification has a specific set of concerns that are addressed. For the Enterprise Viewpoint these include things like Objectives, Roles, Policies, Activities, Lifecycles/Phases, Configuration, Contracts, and Requirements.

A simple example of an Enterprise View is shown in figure 2-1, in which two Enterprise Objects (Enterprises P and Q) are shown as dotted boxes.

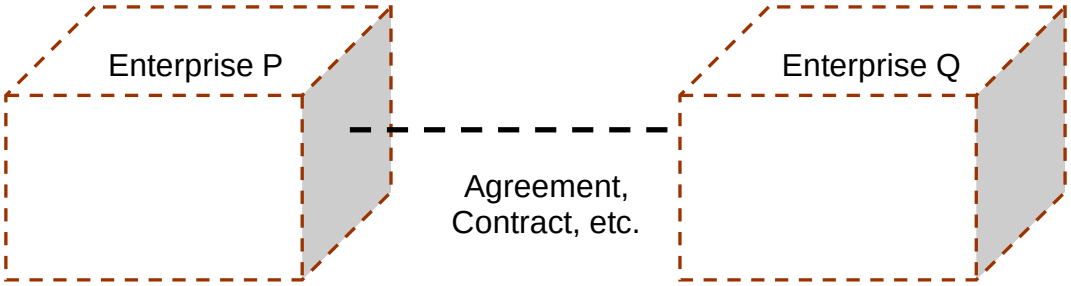


Figure 2-1: Simple Example of an Enterprise View

Systems in a RASDS model are represented as a set of elements, e.g., 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 RASDS by a set of Enterprise, Functional, or Connectivity Views. The organizations and facilities that are part of a system are addressed in an Enterprise View.

2.5 FUNCTIONAL VIEWPOINT

The primary elements in the Functional Viewpoint Specification are Functional Objects and the logical interactions among them. A Functional Object is an abstract model of a functional entity that receives requests, performs actions, and generates or processes data in a space data system. Functional objects that only transport data are called Protocol Entities in RASDS; these are treated explicitly in the Communications Viewpoint (see 2.7). A Functional Object may also be composed of other Functional Objects. A Functional Object may be realized (implemented) by people, but most of them are implemented as basic Engineering Objects, either software and/or hardware, and these basic Engineering Objects are described in the Connectivity Viewpoint.

The Functional Viewpoint shows functional elements and their logical interactions separately from the engineering concerns of how the functions are implemented, where they are allocated, how they are connected, which protocols are used, or what language or hardware is used to implement them. The Functional Viewpoint Specification describes the functional composition of a space data system, its interfaces, actions, and behaviors, and how functions interact with each other. The Functional Viewpoint is used to address these abstract functional aspects of space data systems.

A Functional Object may provide a service to one or more other Functional Objects, use a service provided by another Functional Object, or perform actions jointly with other Functional Objects. These interactions are described in the Functional Viewpoint.

In the Functional Viewpoint Specification the Concerns include: Functional Behavior (actions), Interactions, Interfaces, and Constraints.

A simple example of a Functional View is shown in figure 2-2, in which three Functional Objects (Functions A, B, and C) are represented as ovals, and the functional interactions are

represented by dashed lines. These interactions take place at the interfaces of these objects, and representations of information objects, described more fully in 2.7, are exchanged across these logical links.

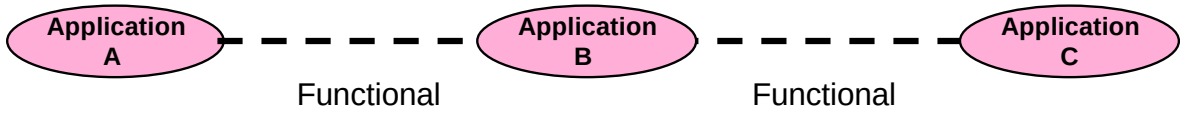


Figure 2-2: Simple Example of a Functional View

A Functional Object may be composed of other Functional Objects. A formal or informal group of Functional Objects that provides some services in a space data 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 may also appear in a Functional View.

The Functional Viewpoint may also be used to describe a class of Functional Objects called Information Management Functional Objects that support the location, access, delivery, and management of Information Objects. These Information Management Functional Objects support the operations of an Information Infrastructure, which is described in detail in the Reference Architecture for Space Information Management (RASIM, reference [5]). Other sets of high-level functionality may also be defined in the future.

2.6 CONNECTIVITY VIEWPOINT

The Connectivity Viewpoint shows how space data systems are made up of physical elements that must operate in space, and the connections between elements, the physics of motion, and external environmental forces that must be considered. Some space data system elements are in motion through space. Consequently, connectivity issues associated with pointing, scheduling, long Round-Trip Light Times (RTLTs), and low signal-to-noise ratios require special protocols and functionality. The Connectivity Viewpoint Specification is used to address these aspects of space data systems, along with the interactions with the 'fixed' external world.

For analysis of complete space systems, other physical aspects, including the propulsion, power, thermal, and structural aspects associated with them, must be considered and represented in what might be called a Physical Viewpoint. However, for the description of space *data* systems, focus is on the Connectivity Viewpoint, where consideration is given to nodes, links, external forces, implementation of functionality as basic Engineering Objects (software or hardware), and other considerations related to the engineering of data system functionality and performance.

In addition to describing the physical structure, behavior and environment of a space data system and the physical connections among elements, the Connectivity Viewpoint also

describes the allocation and engineering of elements that are defined in the Functional Viewpoint and are implemented as hardware and software basic Engineering Objects.

The primary elements in the Connectivity Viewpoint are Nodes, Links, and basic Engineering Objects. A Node is an abstract model of a physical entity that is used in a space data system and is operating in a physical environment. A Node is a configuration of hardware 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 represents a system entity (such as a spacecraft, a tracking system, or a control system) or an individual physical entity of a system (such as an instrument, a computer, or a piece of communications equipment). A Node may be composed of other Nodes. Each Node has one or more Ports where connections to other Nodes are made. For purposes of system analysis, people may also be modeled as Nodes that have functional responsibilities assigned to them.

A Link is a physical connection between or among Nodes. A Link represents an RF or optical link, a hardwired link, or a network of some kind (such as the Internet, a LAN, or a bus). Links connect to Nodes at one of the Ports of the Node. Ports are not always modeled explicitly, but ports are the physical interfaces of Nodes where Links attach. Links are themselves physical Engineering Objects, and they may be hardware (an Ethernet cable) or an RF or optical signal.

For the Connectivity Viewpoint Specification, the Concerns include Distribution, Communication, Physical Environment and interactions, Behavior, Allocation, Performance, Constraints, and Configuration.

A simple example of the Connectivity View is shown in figure 2-3, where three 3-D boxes represent Nodes and two Links 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 are software and hardware Engineering Objects that have correspondences to Functional Objects defined in a related Functional View.

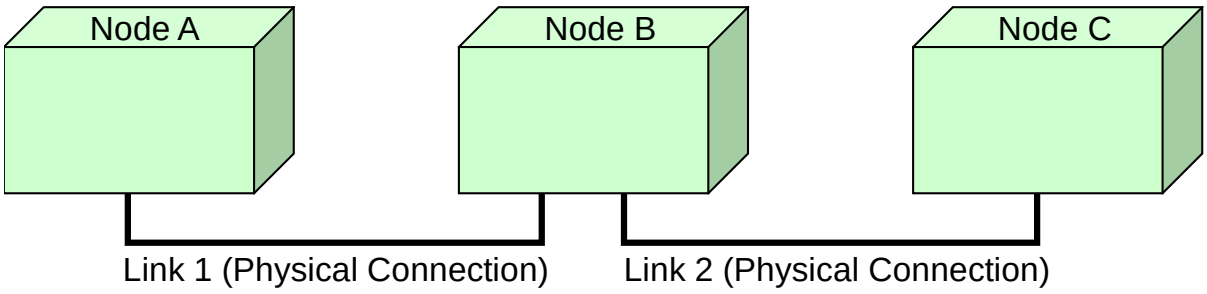


Figure 2-3: Simple Connectivity View Example

As part of the engineering process of designing the system, functions are allocated to physical nodes and implementation choices are made about use of software or hardware. It is also possible, for purposes of system analysis, that Functional Object responsibility may also

be assigned to people. 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.

When this level of detail is required to be described, the allocation of Software Engineering Objects (software or applications) is shown by overlaying a representation of the implemented Functions on the Nodes shown in a Connectivity View. Such an example is shown in figure 2-4, in which engineered versions of the Functional Objects from figure 2-2 are shown overlaid on the Connectivity View objects shown in figure 2-3.

The allocation of Hardware Engineering Objects (hardware) may be shown using the Node 3-D representation, if they are major components in the system, or they may be shown as simply an oval representing the function performed by minor hardware elements.

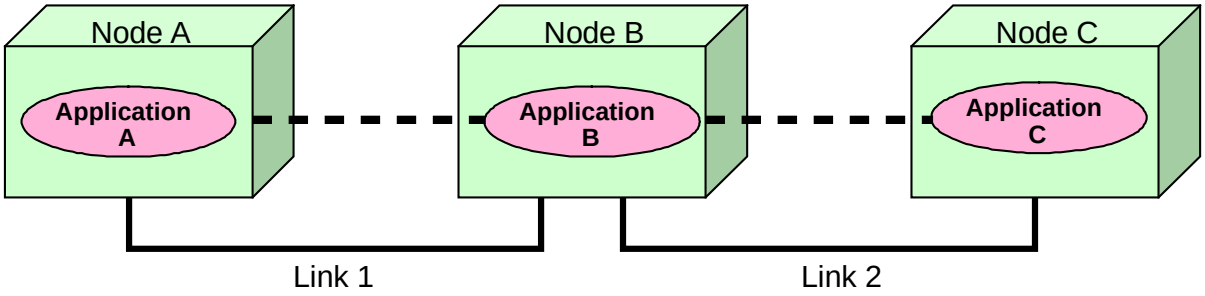


Figure 2-4: Example Connectivity View Showing Implemented Functions

A Node in the Connectivity View may represent a computer and its software (including the operating system and applications). A node may have internal structure such as the operating system, processing, storage, and communications functions provided for use by other Engineering Objects within the node to which it belongs. This internal structure may not be of concern in many Connectivity Views, but these elements may also be modeled and described explicitly in a more detailed view where it is useful to do so. A layered view of operating system elements is often used where this detail is required, but RASDS does not prescribe any particular representation for this.

2.7 COMMUNICATIONS VIEWPOINT

The Communications Viewpoint is used to describe the layered sets of communications protocols that support communications among the Nodes in the system. These protocols must meet the requirements imposed by the actual physical connectivity in the deployed system and the operational challenges of communicating in space. The communications protocol data units are sometimes called the ‘wire protocol’ and are often associated with the Link that is used to carry them, but they are generated and consumed by protocol peer entities that are operating in the connected Nodes. The Communications Viewpoint is used to address these protocol aspects of space data systems.

The Communications Viewpoint Specification describes the mechanisms that support information transfer among system elements (i.e., hardware or software Engineering Objects) in a space data system. These protocol 'stacks' are directly associated with the physical Links that exist between the Nodes of a system, and they have responsibility for transporting data across these links.

NOTE – The Communications Viewpoint is actually a subset of the RM-ODP Engineering and Technology Viewpoints. The objects modeled in the Communications Viewpoint might properly be called Communications Objects for consistency with the rest of the RASDS. However, to provide familiar terminology for protocol designers, and for consistency with the widely used ISO/IEC Basic Reference Model (ISO-BRM, reference [14]), they are here called Protocol Entities.

Protocol Entities (Communications Objects) support information transfer between or among Engineering Objects. A Protocol Entity is an abstract entity that implements a communications protocol. A Protocol Entity may be realized as software and/or hardware. A stack of one or more Protocol Entities is used to support information transfer from one Engineering Object to another Engineering Object to allow them to interact. In the stack, the topmost Protocol Entity directly supports some application Engineering Object, and the bottommost Protocol Entity interfaces to the Physical Link (i.e., the physical connection between or among Nodes). Protocol Entities offer services that are exposed at a Service Access Point (SAP).

Each Protocol Entity implements one or more Protocols, which are most often defined by the Protocol Data Units (PDUs) that are exchanged between peer entities and the actions performed by those Protocol Entities when they receive any of several defined PDUs. Activities within Protocol Entities may also be triggered by reception of service requests from users, by external management requests, and by internal events such as time-outs or in-line management requests.

For the Communications Viewpoint Specification the Concerns include Standards, Technology, Functionality, and Suitability.

A simple example of a Communications View is shown in figure 2-5, in which two stacks of Protocol Entities (implementing Protocols 1, 2, and 3) are represented as two groups of rectangles. Each rectangle represents a Protocol Entity that implements services for a 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 protocol data units with its peer N-layer Entity.

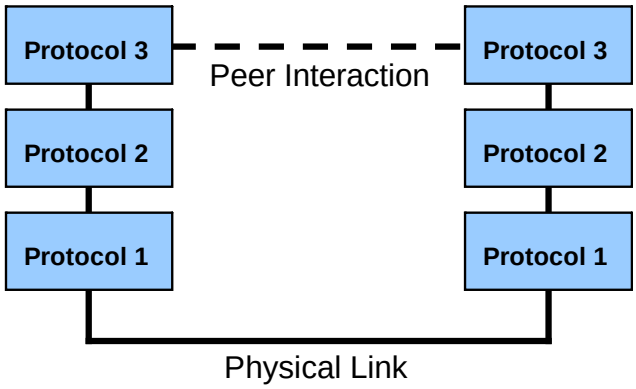


Figure 2-5: Simple Example of a Communications View

The selection of specific protocols to support information transfer between application Engineering Objects depends heavily on the characteristics of the functions being performed, the Nodes, the Link, and the characteristics of the communications environment within which they are operating. To define all these relationships, the Protocol Entities are typically shown altogether along with representations of the application Engineering Objects, the Nodes, and the Link. Such an example is shown in figure 2-6, in which the Protocol Entities are shown in relationship with representations of Nodes, Links, and other Engineering Objects, identified as Applications A and B.

In descriptions of Protocol Entities in a Communications View, the terms in the ISO-BRM are used by preference. This document has adopted the ISO-BRM seven-layer model that starts with the Physical layer and ends with the Application layer. Not all layers are required or used in space data systems, many of which have traditionally been constructed with applications interfacing directly to a Data Link layer SAP. Furthermore, in space data systems, it is typical to show the ISO Physical layer with separate modulation and coding sublayers, which are specialized in CCSDS for space communication. Where a messaging service is provided, which was not a defined service when the ISO-BRM was specified (1979), it is included with other Application layer services in layer 7 of the stack.

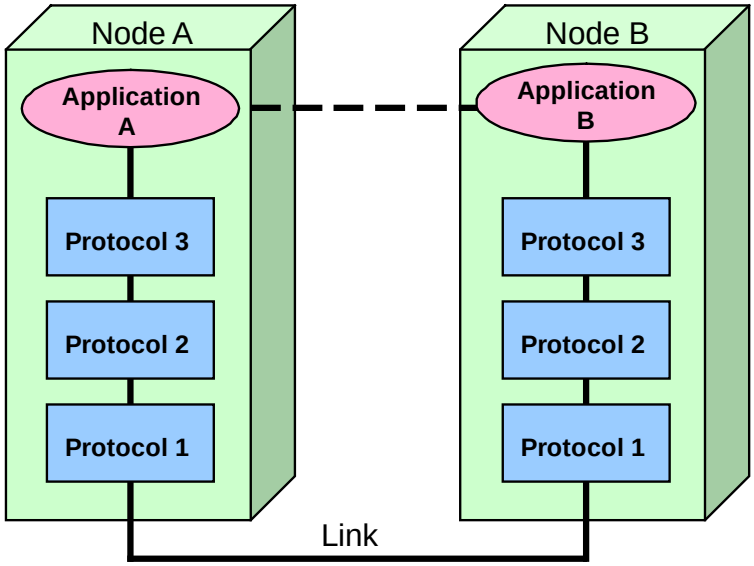


Figure 2-6: Communications View with Protocol Entities and Engineering Objects

NOTE – RM-ODP addresses communications protocols as part of its Engineering and Technical Viewpoint specifications. Use of the ISO/IEC Basic Reference Model (ISO-BRM) to describe this Viewpoint is entirely consistent with RM-ODP and is adopted here. In RM-ODP communications protocols are treated as just one of several Engineering and Technology Viewpoint choices that an architect must make in designing a system. The separate Communications Viewpoint is introduced in RASDS because communications protocols are a core element in the CCSDS body of standards and are an essential part of end-to-end design trades when designing systems to operate in space.

2.8 INFORMATION VIEWPOINT

The Information Viewpoint of space data systems describes how data objects and their structures, relationships, metadata, and constraints are defined and configured within the system.

The Information Viewpoint looks at space data systems from the perspective of the Information Objects that are defined and managed. It 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 relationships among Information Objects are shown in figure 2-7.

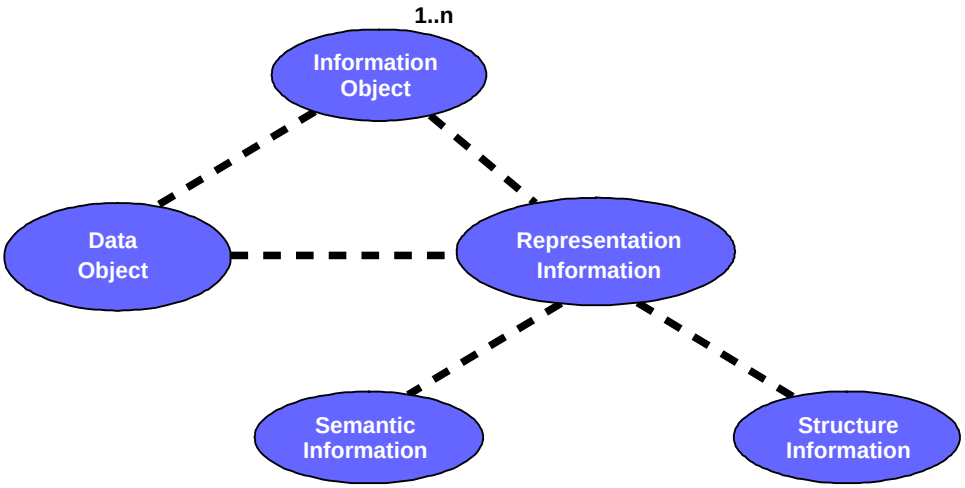


Figure 2-7: Example of Information View Showing Basic Object Model

For the Information Viewpoint Specification, the Concerns include Structure, Semantics, Relationships, Constraints, Permanence, and Rules.

The primary elements in the Information Viewpoint Specification are Data Objects and their logical links or relationships. Data Objects may be composed of other Data Objects. They may also have associated metadata, or data about data, that describes the syntax, semantics, and structure of the data and rules on its use. The Information Objects that are used in the system may have an explicit formal representation in a Data Architecture, or they may appear simply as Data Models or schema that have explicit representation in a Connectivity View as part of a database definition or data structure specification.

In this same context, an Information Object corresponds to a set of Connectivity Objects, a static schema of an Information Object corresponds to possible states of the Connectivity Objects. Every change in state of an Information Object corresponds either to some set of interactions between Connectivity Objects or to an internal action of a Connectivity Object. The schema of the Information Object corresponds to the behavior, state, and environmental interactions of the Connectivity Objects. Details of the information Viewpoint are addressed much more completely in the companion document, Reference Architecture for Space Information Management (reference [5]).

2.9 CORRESPONDENCES BETWEEN VIEWPOINTS

Each of the RASDS Viewpoint Specifications is intended to be orthogonal to the others, and a description of a given system from any one Viewpoint will be self-consistent. However, most of the objects defined in their home RASDS Viewpoint Specification also have identifiable relationships or correspondences with objects defined in other Viewpoint Specifications. This is an essential element of the methodology. In figure 2-8 the relationships among the core set of objects defined in RASDS are shown graphically.

In figure 2-8 only the top-level (or core) objects and attributes in RASDS are shown, those most central to understanding how core objects from different Viewpoints are related. Each of these core objects represents a class of objects, as described in prior sections. These classes may have subclasses and an associated set of attributes. A much longer list of attributes for each of these objects and their classes is not shown here. Finally, figure 2-8 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 called 'Perspective' in the figure 2-8 represents the RASDS 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 that 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 RASDS, 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 may 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.

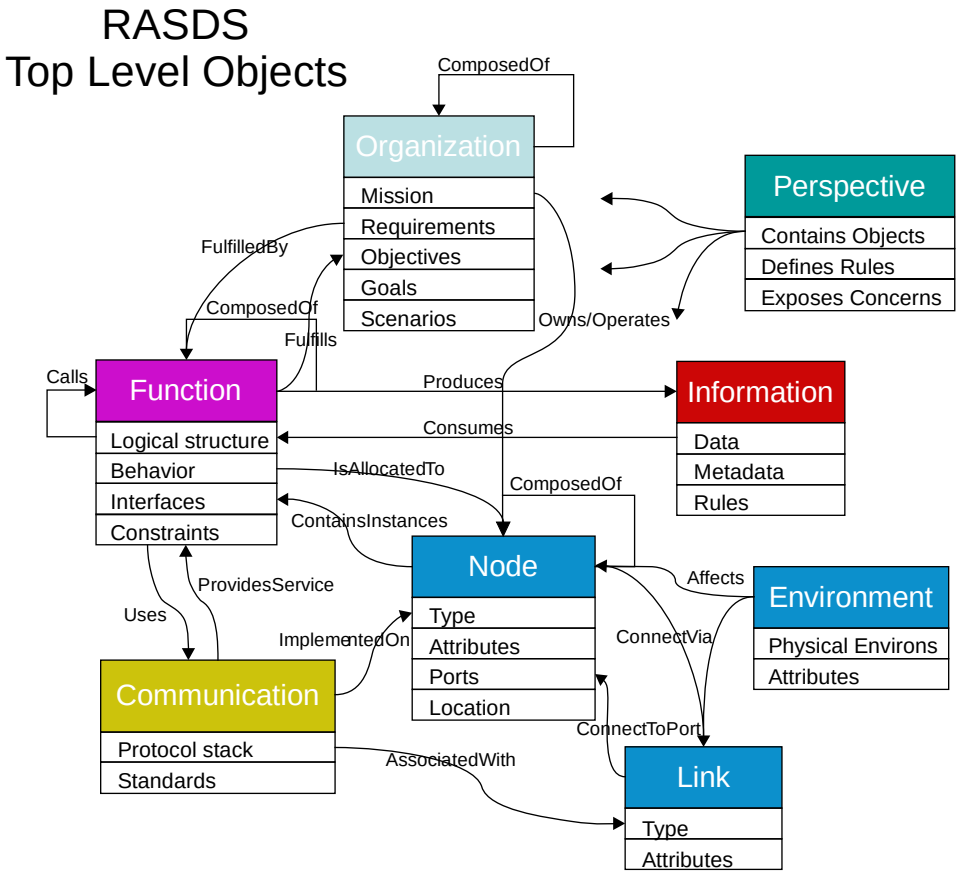


Figure 2-8: RASDS Top Level Object Relationships

Basic inter-Viewpoint Relationships:

- Organizations have missions, goals, objectives, and requirements that are fulfilled by the Functions defined within the system. They also own, operate, and develop the Facilities that are engineered as physical Nodes and Links in the system.
- Functions describe the behaviors in the system, and they are implemented as Engineering Objects (either hardware or software) and allocated to Nodes that may contain instances of several different implemented Functions.
- Applications (software Engineering Objects) use Communications protocols to transfer Information among themselves. These protocols are defined by Standards.
- Information is produced, transformed, and consumed by Functions.
- Nodes are connected via Links that connect to Ports on the Nodes. Communications protocols are associated with these Ports.
- The physical Environment affects Nodes and Links alike.
- Most of these classes of objects may be composed of other subclasses of objects.

Each Viewpoint Specification defines a particular set of basic objects, which can be considered as their home definition, but many of these objects have representations or correspondences in other Viewpoints. An example is an exchange of contracts between organizations in an Enterprise View. This View is the home for defining Enterprise Objects, i.e., organizations and facilities, and their relationships, which may involve establishment of a contractual association. However, the definition of the structure and information content of the contract, and its relationships with other information objects, will be found in an Information View. This concept of correspondences or representations is described at more length in the following subsections.

A set of RASDS-compliant system specifications that use different Viewpoint Specifications should not contain contradictory statements; i.e., the Views should be mutually consistent. Thus, a complete specification of a system will include statements of correspondences relating elements in one View to elements in another View, showing that the model is consistent. The minimum requirement for consistency in a set of RASDS specifications for a system is that they adhere to the rules for the selected views and exhibit within the set of specifications the correspondences defined in this Reference Architecture.

3 BASIC CONCEPTS

3.1 GENERAL

This section defines basic concepts and terms of object modeling that are used throughout this document. The RASDS, like the RM-ODP from which it is derived, is essentially a model-based engineering approach to specifying system architectures. Object modeling for system architecture provides a formalization of well-established abstraction and encapsulation design practices that are familiar from structured programming.

Abstraction allows the description of system functionality to be separated from 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 modeling concepts, providing rigorous definitions of a minimum set of concepts (object, interface, action, and interaction) that form the basis for RASDS system descriptions and are applicable in all Viewpoints;
- specification concepts that address notions such as object type and class that are necessary for reasoning about specifications and the relationships among specifications, provide general tools for design, and establish viewpoint specification languages;
- structuring concepts that build on the basic modeling concepts and the specification concepts to provide useful viewpoints on space data system architectures, address recurrent structures 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 8.

NOTES

1 In each of the sections that present Viewpoint Specifications, the definitions that are used in that Viewpoint will be repeated, and definitions that are used only in that Viewpoint will be given.

2 The following definitions were derived from RM-ODP but are somewhat simpler than those of RM-ODP so that they can be understood more easily and intuitively.

3 In common with RM-ODP all of the Viewpoints in RASDS are defined in terms of objects, but this terminology is not that same as 'object oriented' design terminology. In this key instance, and in many other the reader is cautioned to attend to the specific definitions provided for terms used in RASDS.

3.2.2 BASIC ELEMENTS

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. While in general the word entity can be used to refer to anything, in the context of modeling it is reserved to refer to things in the universe of discourse being modeled.

An **Element** is (1) a constituent part of something; (2) any thing that is one of the individual parts of which a composite entity is made up; (3) 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 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, i.e., 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.

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 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.

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 object to support actions of another object.

Relationship describes the way that two or more entities can be associated with each other.

Structure refers to the relationships that exist among a set of elements and 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.

3.2.4 SPECIALIZED 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 object to achieve a goal. The transformation of inputs to outputs that 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.

A logical object interacts with other objects over a logical link. A physical object interacts with other objects over a physical link.

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. 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.

A **resource** is (1) anything that is available to a system and can support the achievement of objectives; (2) any physical or virtual element that may be of limited availability within a system.

3.2.5 GENERAL TERMS

Interoperability refers to 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.

A **Requirement** is 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.

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

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

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 describes in detail the capabilities and establishes the requirements to be met by interfacing subsystems to achieve compatibility.

Architecture is the concepts and rules that define the structure, semantic behavior, and relationships among the parts of a system. 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 defining, documenting, maintaining, improving, and certifying proper implementation of an architecture. It is both a science and an art.

A **system** is 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. Every system has an architecture and includes a selected set of entities, even if the architecture is not clearly described.

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.

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

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. 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 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 data 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.

Correspondence is (1) a function such that for elements in one Viewpoint there is a related element in another Viewpoint; (2) the condition of being in conformity from elements in one Viewpoint to elements in another.

A **representation** is some (1) way of organizing, manipulating, presenting, and storing information; (2) a visual or tangible rendering of something.

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.

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.

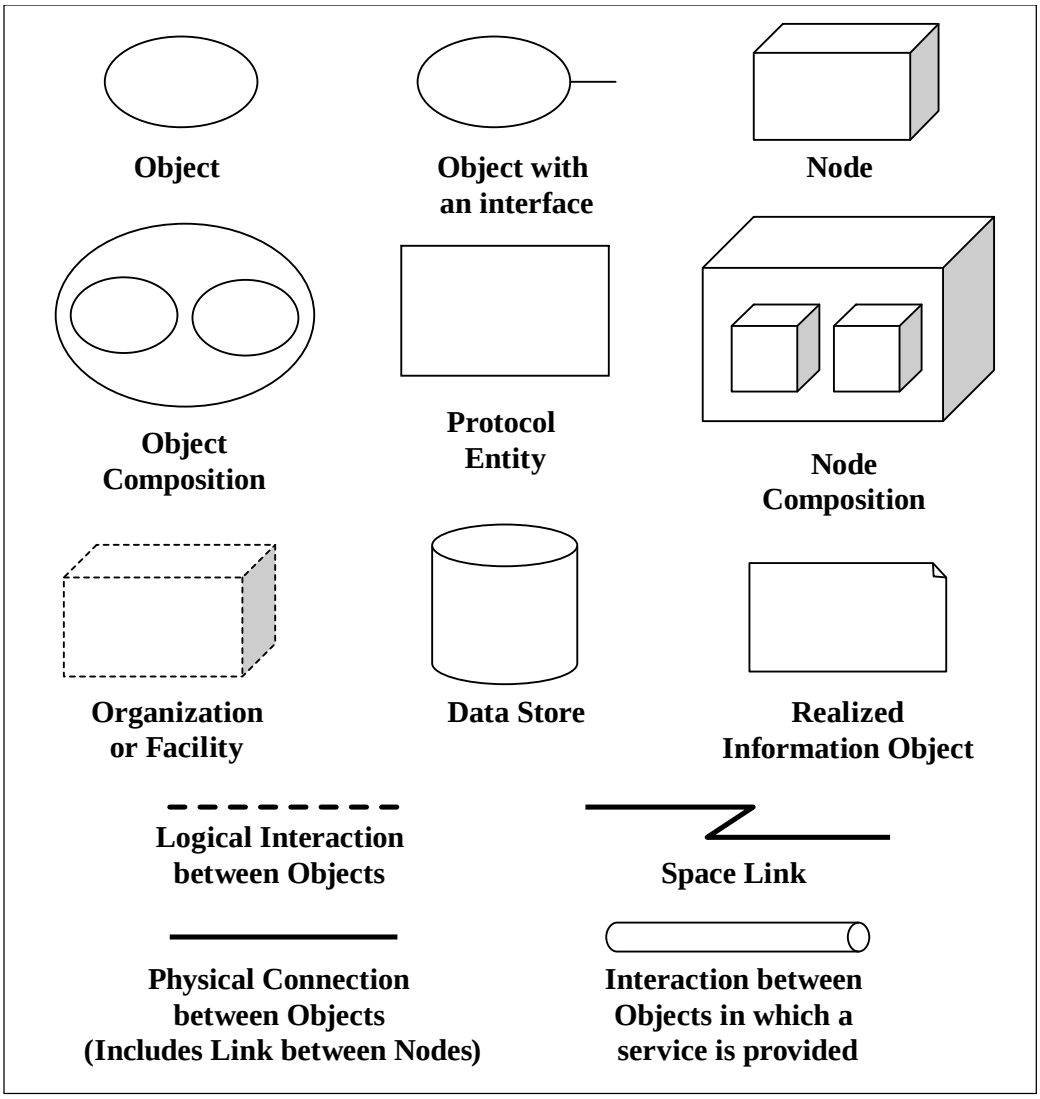


Figure 3-1: Icons Used in This Document

3.4 CHARACTERISTICS OF OBJECTS

As explained in section 2, the Viewpoints of RASDS are described with Objects and their interactions. The typical interfaces and attributes of Objects are shown in figure 3-2. A related specific representation of the objects relevant to each Viewpoint are used throughout this document.

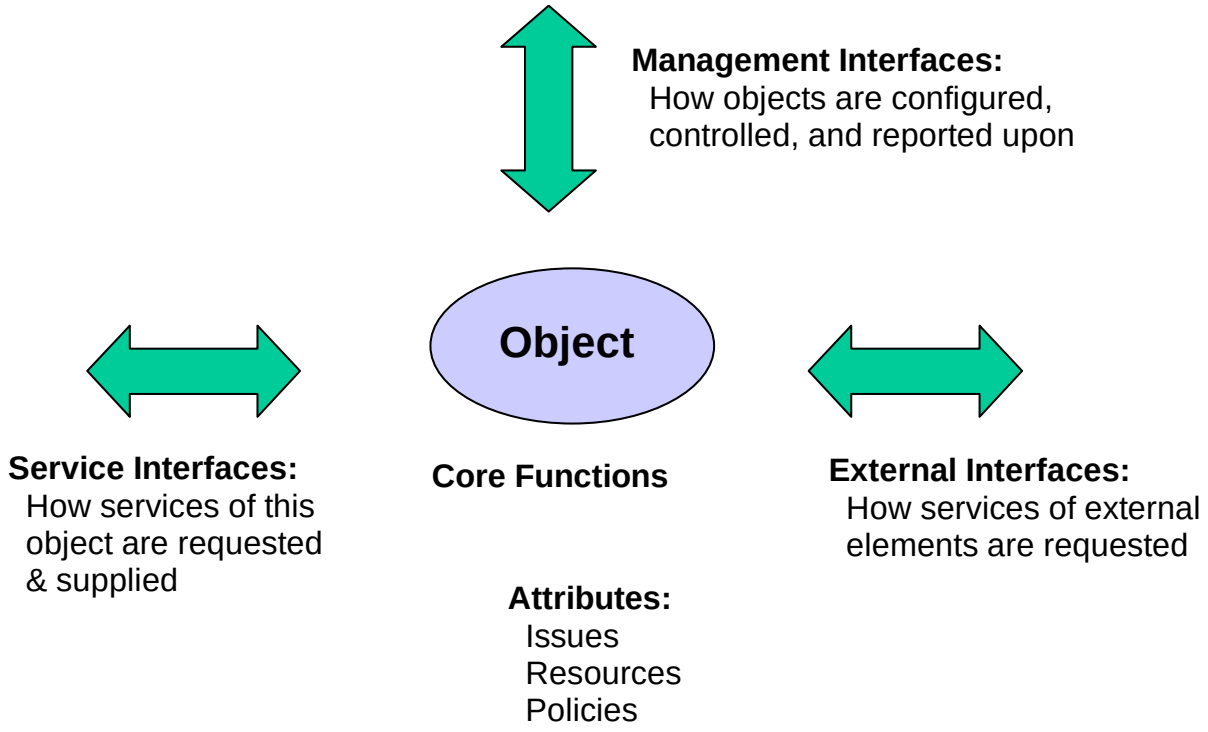


Figure 3-2: Representation of Objects

Any given Object may expose one or more Service Interfaces and provide one or more Core Functions. Through its External Interface, 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 RASDS that do not exhibit all of these interfaces are Information Objects.

NOTES

1 The types of arrows used to indicate the kinds of interfaces in this, and other object diagrams, has no special significance in RASDS, but double headed arrows are typically used to represent input/output interfaces and single headed arrows are used to indicate directionality of the interfaces.

2 RASDS does not assign any particular meaning to the use of color and the user is free to employ it wherever it assists in clarifying the representation of the architecture or bringing consistency to a set of diagrams.

4 ENTERPRISE VIEWPOINT

4.1 OVERVIEW

The Enterprise Viewpoint 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.).

NOTE – The Enterprise Viewpoint is based on the Enterprise Viewpoint of RM-ODP, but some modifications have been made to better describe the space data systems. In particular, special enterprise objects called Space Enterprises are introduced.

4.2 CONCEPTS

The **Enterprise Viewpoint** of a space data 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 data system is depicted as a set of Enterprise Objects and their relationships, interactions, and the roles that they perform. Enterprise Objects representing system elements 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.

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 in general terms 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. 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.[1]

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.

[1] System management, lifecycle views on systems, scenario specifications, and other aspects that are relevant to the Enterprise Viewpoint will be addressed in a later issue of this document.

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.

4.3 ENTERPRISE OBJECTS

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

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.

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., 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.

Table 4-1 also shows other typical Enterprise Objects involved in many space data systems and some of their containment relationships. How any specific Space Enterprise is decomposed into component Enterprise Objects 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

4.4 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.[1]

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, as well as data acquisition, data relay, orbiter, lander, or other descriptive names.

[1] Detailed descriptions of the attributes of Enterprise Objects will be provided in a later issue of this document.

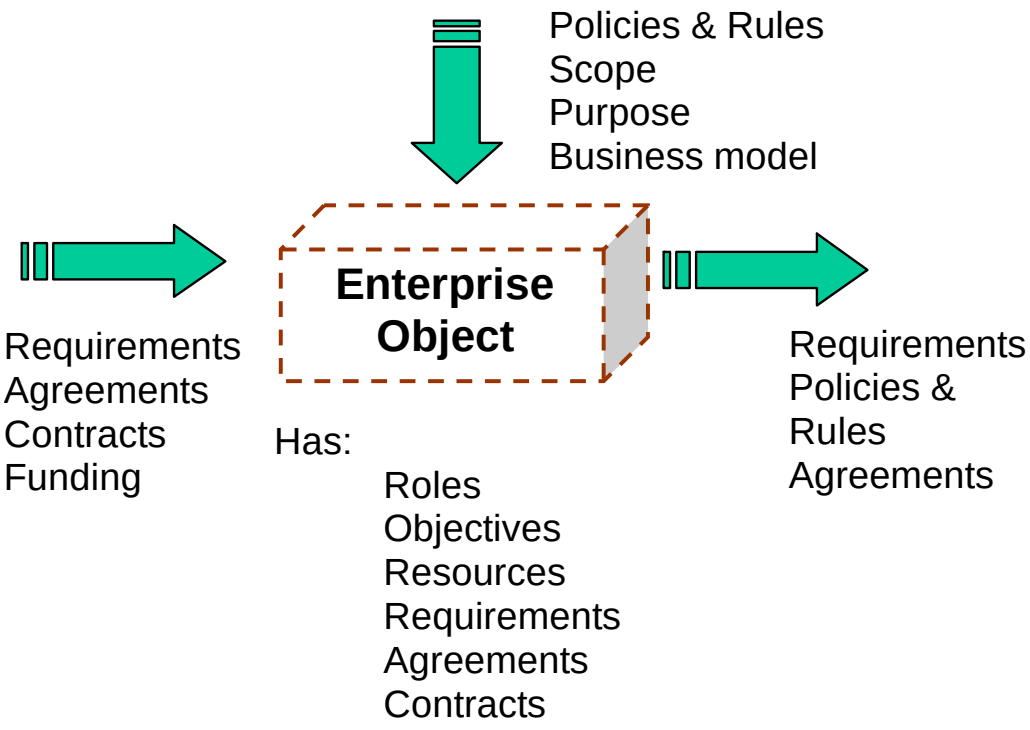


Figure 4-1: Attributes of Enterprise Objects

4.4.2 OBJECTS AND RELATIONSHIPS

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 Resources (facilities or other elements having enterprise operational or service roles);

  • Attributes: name, role, objectives, point of contact, location, members, 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).

4.4.3 CONCERNS

Concerns are

- the purpose, scope, and policies for the system;
- the objectives, operations concepts, and scenarios for the system;
- the requirements and constraints on the system;
- roles played by the system elements;
- activities undertaken by the system.

4.4.4 OTHER TERMS

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.

An **operations concept** is a verbal or graphic statement, in broad outline, of assumptions or intent in regard to 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 EXAMPLES OF SPACE DATA SYSTEMS DESCRIBED WITH ENTERPRISE VIEWPOINT

Some examples of the Enterprise Viewpoint are shown in figures 4-2 and 4-3.

Enterprise Objects involved in the operation of Mission A are shown in figure 4-2 together with the interfaces between them. Mission A is a mission of Agency ABC, not involving cooperation with other Agencies, and therefore all the Enterprise Objects belong to Agency ABC. The domain boundary of Agency ABC operations is shown as an oval. Organizational elements are shown as dashed 3-D boxes and the logical links between them are shown as dashed lines.

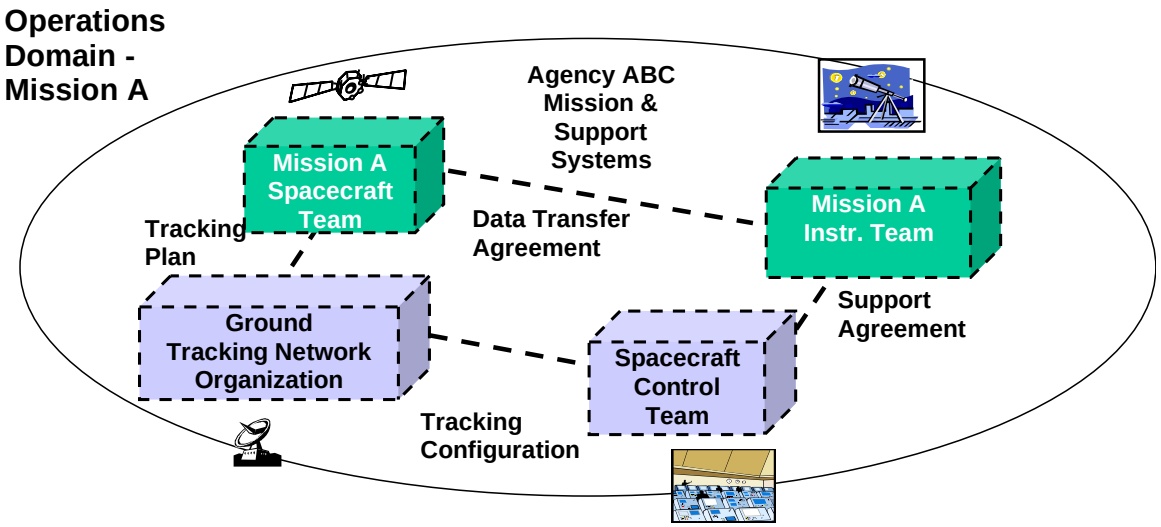


Figure 4-2: Example of an Enterprise View (Mission A)

NOTES

1 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.

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

Enterprise Objects involved in the operations of Mission Q are shown in figure 4-3 together with some of the interfaces between them. Mission Q is a joint mission between Agencies ABC and QRS, and therefore some Enterprise Objects belong to Agency ABC and some to Agency QRS. Shared missions often are based upon quid-pro-quo arrangements, involving some sort of agreement or contract, and require a relationship of trust and interdependence between organizations. Two primary kinds of elements are shown in this Enterprise Viewpoint, organizations and their resources (facilities).

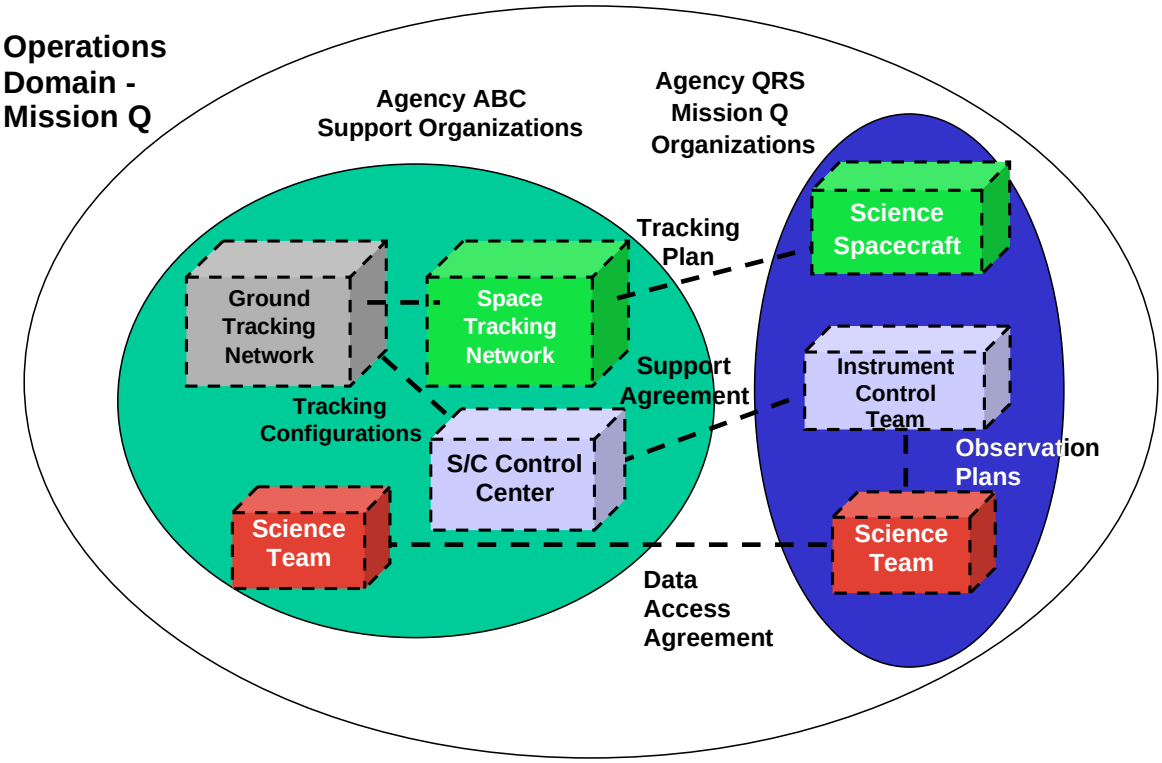


Figure 4-3: Example of Multi-Agency Enterprise View (Mission Q)

4.6 SECURITY ISSUES IN THE ENTERPRISE VIEWPOINT

In the Enterprise Viewpoint the security issues that should be addressed include organizational roles, policies, rules, trust relationships, domain boundaries (e.g., operational vs. science) and cross-support security agreements. The implementation mechanisms to enforce these rules and agreements are detailed in other views. Security responsibilities, threats, counter-measures, and issues are being addressed in detail in a separate Security Architecture document (reference [4]) and in other related security documents within CCSDS.

5 FUNCTIONAL VIEWPOINT

5.1 OVERVIEW

The Functional Viewpoint separates the analysis of abstract functional elements and their logical interactions 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 do design trades separate from functional design.

NOTE – 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. In RM-ODP, application processes are always expected to be implemented as pieces of software residing in computers, hence the name computational viewpoint. In space data systems, however, application processes do not always reside in computers. They may reside in simple devices or be implemented in hardware for efficiency. For this reason the word 'functional' was chosen instead of 'computational'. The detailed engineering of these application elements, whether in software (e.g., choice of languages) or hardware (e.g., selection of discrete logic or FPGA), is not directly treated in RASDS except that the implementation of functions as Engineering Objects and their allocation to Nodes is addressed in the Connectivity View.

5.2 CONCEPTS

The **Functional Viewpoint** of a space data 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 data 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.

For describing 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 separate views of the same Functional Objects, all of which obey the same rules, may be

required in order 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 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 8).

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 Viewpoint. The physical means for providing connections among implemented functions are treated in the Connectivity Viewpoint (section 6), as are the physical attributes of the connections and their behavior. These allocation processes are part of the Engineering Viewpoint in RM-ODP.

5.3 FUNCTIONAL OBJECTS

Table 5-1 shows typical Functional Objects used in space data systems. These are provided only as examples, and any given system may decompose these differently or use other names for the same functions. For instance, Orbit Determination and Trajectory Design may 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.[1]

[1] Table 5-1 will be revised in a future issue of this document so that the Functional Objects shown therein will also represent specific services provided by CCSDS and other space data systems standards.

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.

Table 5-2 shows a number of typical infrastructure 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 are shown as a layer, as described in 9.3.

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 considered to be a Communications Viewpoint set of objects.

5.4 CHARACTERISTICS OF FUNCTIONAL OBJECTS

5.4.1 GENERAL

The interfaces of Functional Objects are shown in figure 5-1.

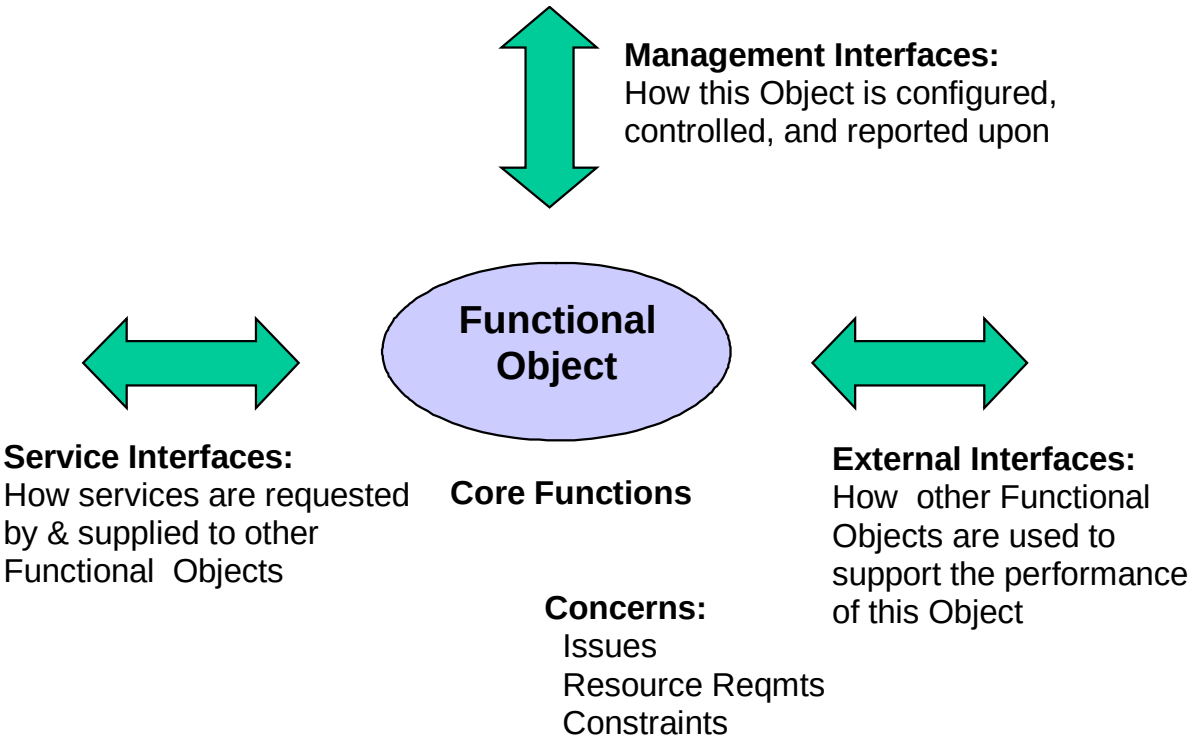


Figure 5-1: Functional Object Interfaces

The interfaces of Functional Objects are classified into three categories: Service Interfaces, External Interfaces, and Management Interfaces.[1] 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. These interfaces may have different signatures, such as client–server, producer–consumer, or publisher–subscriber. Good practice identifies generalized sets of Functional 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.

5.4.2 OBJECTS AND RELATIONSHIPS

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).

5.4.3 CONCERNS

Concerns are

- a functional decomposition of the system into objects that interact at interfaces;
- the abstract behavior of the system, its interactions and constraints.

[1] Detailed descriptions of the attributes, interfaces, and behavior of Functional Objects will be provided in a later issue of this document.

5.4.4 OTHER TERMS

Abstraction is a mechanism and practice to reduce and factor out details so that one can focus on 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.

5.5 EXAMPLE OF A SPACE DATA SYSTEM DESCRIBED FROM THE FUNCTIONAL VIEWPOINT

Figure 5-2 shows a representative set of Functional Objects used in typical space data systems together with the logical interactions that occur among them (shown with dotted lines). The points of connection are at the interfaces, and on these logical links flow various forms of information, which are not shown on this particular view.

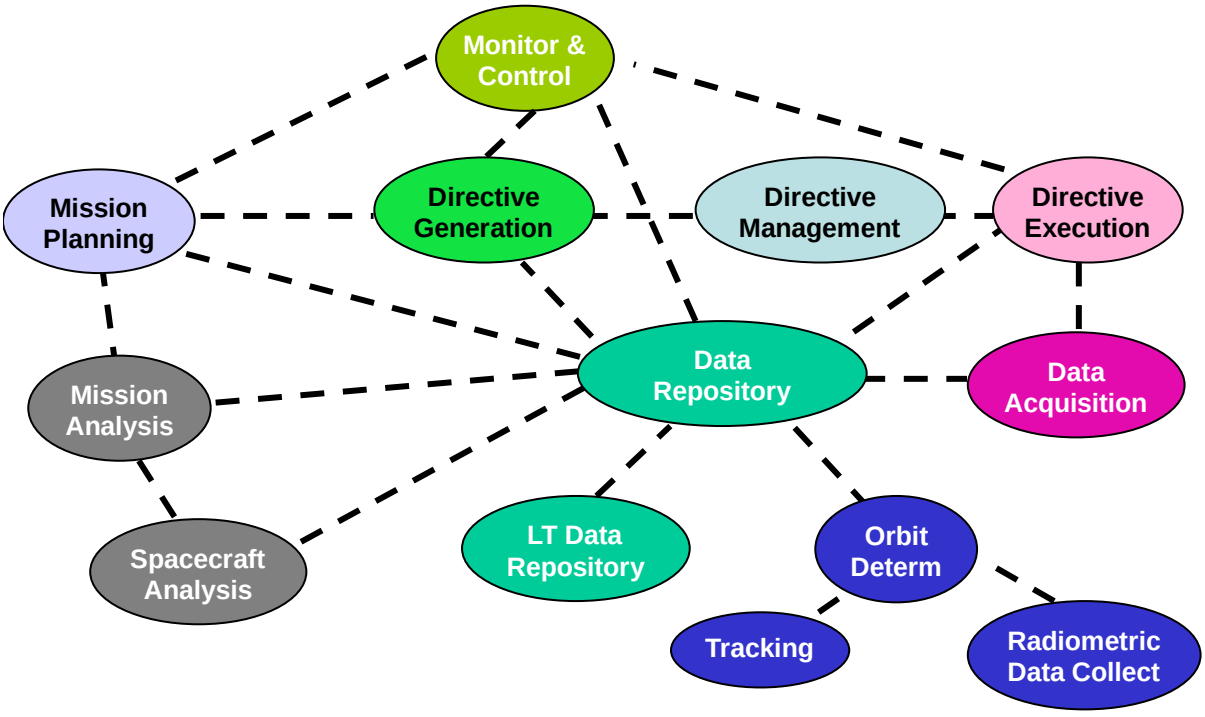


Figure 5-2: Example of Functional View (Functional Objects and Interactions)

Other views of this same set of Functional Objects might be shown, including a view 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; or another view might show the specific information objects that flow between the elements or the signatures and other details of the interfaces. All of these views are

developed using the same Functional Viewpoint specification. RASDS has a simple representation for showing information flows between objects. For an example, see figure 5-3, which uses arrows to indicate directionality and a simple object to indicate flow of real data objects. See section 8 for more discussion of the Information Viewpoint where these objects are defined.

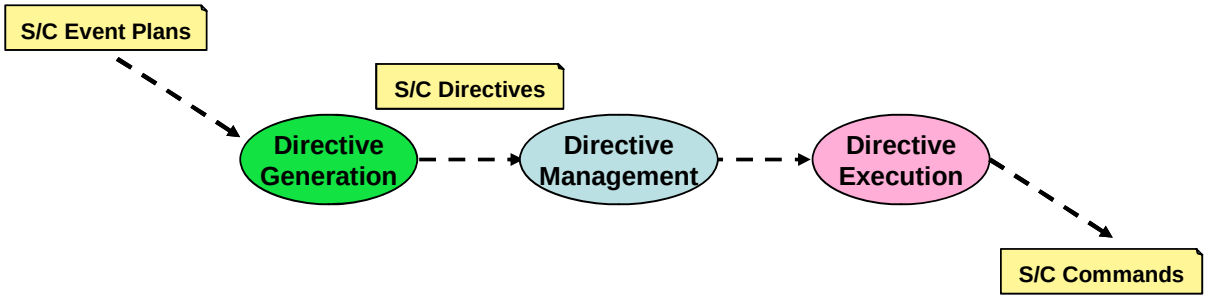


Figure 5-3: Example of Information Flows between Objects

RASDS does not provide any specific recommendations for the representation of the behavior of Functional Objects. Decomposition to lower levels of detail is one way of showing the this. Other formal means, such as use of state charts, activity diagrams, or algorithmic specifications may also be employed as needed. For describing interface signatures any suitable annotation may be adopted to represent and label information objects or to label interfaces. RM-ODP uses a simple name/attribute/operation construct, and this may be adopted.

5.6 EXAMPLE OF SPACE DATA SYSTEM WITH INFORMATION MANAGEMENT INFRASTRUCTURE

Included in the Functional Viewpoint are Information Management Functional Objects, the elements of an information infrastructure that supports 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.

Figure 5-4 shows a representative set of Functional Objects that might be used to carry out some activity. Supporting these is the set of Information Management Functional Objects that provide an infrastructure for managing, accessing, locating, and distributing the information exchanged by the Functional Objects.

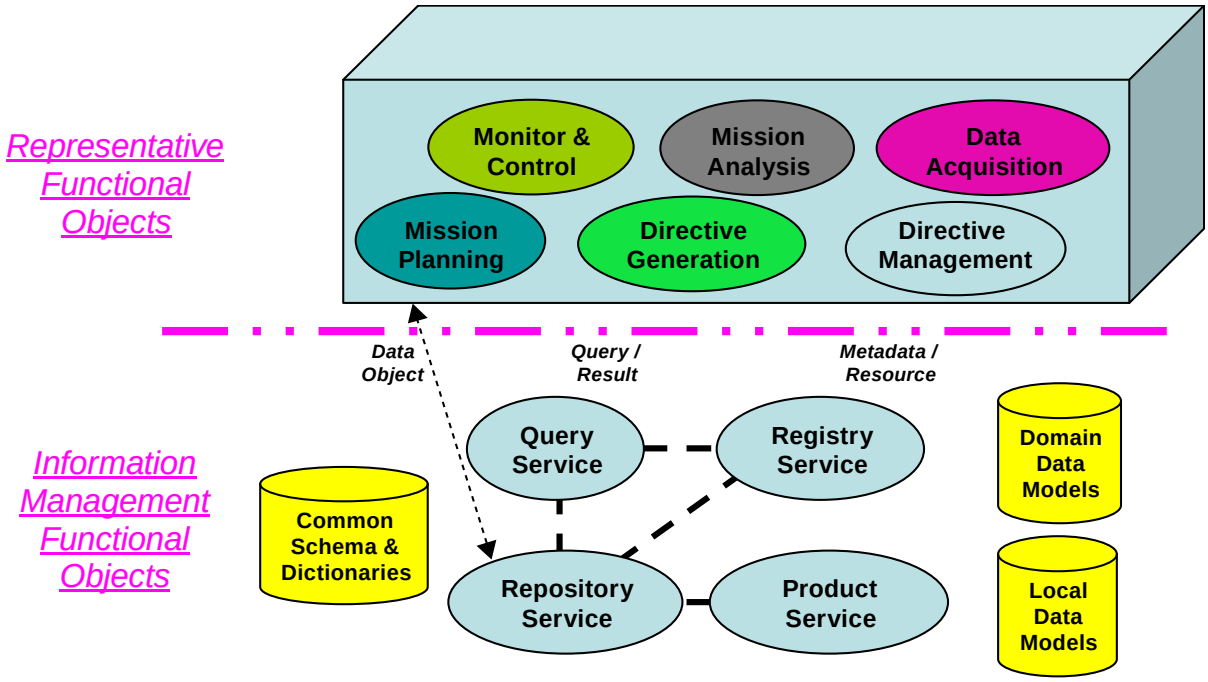


Figure 5-4: Representative Functional Objects and Information Management Infrastructure Elements

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 functional elements in their own right, implemented as subsystems and using various commercial elements like data base management systems (DBMS) and distributed system frameworks.

These basic Information Management Functional Objects may be composed into a broad set of other 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. The description of these Information Management Functional Objects, their functions, and interfaces is being separately addressed in the Reference Architecture for Space Information Management (reference [5]).

5.7 SECURITY ISSUES IN THE FUNCTIONAL VIEWPOINT

In the Functional Viewpoint, the Functional Objects and services 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).

6 CONNECTIVITY VIEWPOINT

6.1 OVERVIEW

The Connectivity Viewpoint represents physical elements that operate in space, where connections between elements, the physics of motion, and interactions with forces in the external environment are considered. The Connectivity Viewpoint deals with the composition of these physical elements and their connections and interactions. For analysis of space systems in general, all of the physical aspects of the system, including the propulsion, power, thermal, structural, etc., aspects associated with them, must be considered and represented in what might be called a Physical Viewpoint. For the description of space *data* systems, focus is only on the aspects addressed in the Connectivity Viewpoint, where consideration is given to nodes, links, computational and data transport functions, external forces, and other considerations related to the engineering of data system functionality and performance.

Some elements of space data systems are in motion through space and consequently connectivity issues exist associated with pointing, scheduling, long round-trip light times, intermittent visibility, and low signal-to-noise ratios. All of these require special protocols and functionality to deal with. The Connectivity Viewpoint is used to address these aspects of space data systems.

The Connectivity Viewpoint also includes all of the other aspects of space data 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, and other components as well as hardwired links, buses, and RF and optical links. The Connectivity Viewpoint is where these engineering issues are handled, along with the issues associated with choosing the strategy of how to implement the selected logical functionality in hardware and software components.

NOTE – The Connectivity Viewpoint is one aspect of the Engineering Viewpoint of RM-ODP. It is called out separately in RASDS because it exposes, in the design of space data systems, physical issues and constraints that are distinct from those encountered in typical terrestrial distributed systems.

6.2 CONCEPTS

The **Connectivity Viewpoint** is an engineering view on a space data system that shows Engineering Objects, which may be hardware or software. The Connectivity Viewpoint is focused on a **Node** and **Link** view of a system, the composition of the Nodes, the physical connections among Nodes, their physical and environmental constraints, and their physical dynamics. The Connectivity Viewpoint also describes how the abstract functional design described in the Functional Viewpoint is to be implemented as software Engineering Objects, i.e., applications or software components, or hardware Engineering Objects, and how these are allocated on the major hardware Engineering Objects (Nodes) of the system. These Engineering Objects are implemented representations of the Functional Objects that were described in detail in section 5.

In the Connectivity View, a space data system is depicted with Nodes and the physical connections among them (Links). This view is also used to describe how these Nodes move and the effects that the environment has upon their behaviors. This view includes description 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 to understanding the challenges from the physical environment in which the systems operate, particularly the motion, discontinuous connectivity, and extremely distant and broad distribution of physical devices. Specialized protocols and systems design are needed to deal with the many aspects of the physical environment.

6.3 CHARACTERISTICS OF CONNECTIVITY OBJECTS

6.3.1 GENERAL

The primary objects shown in the Connectivity Viewpoint are physical Nodes and the Links that connect them. The interfaces of Nodes are shown in figure 6-1.

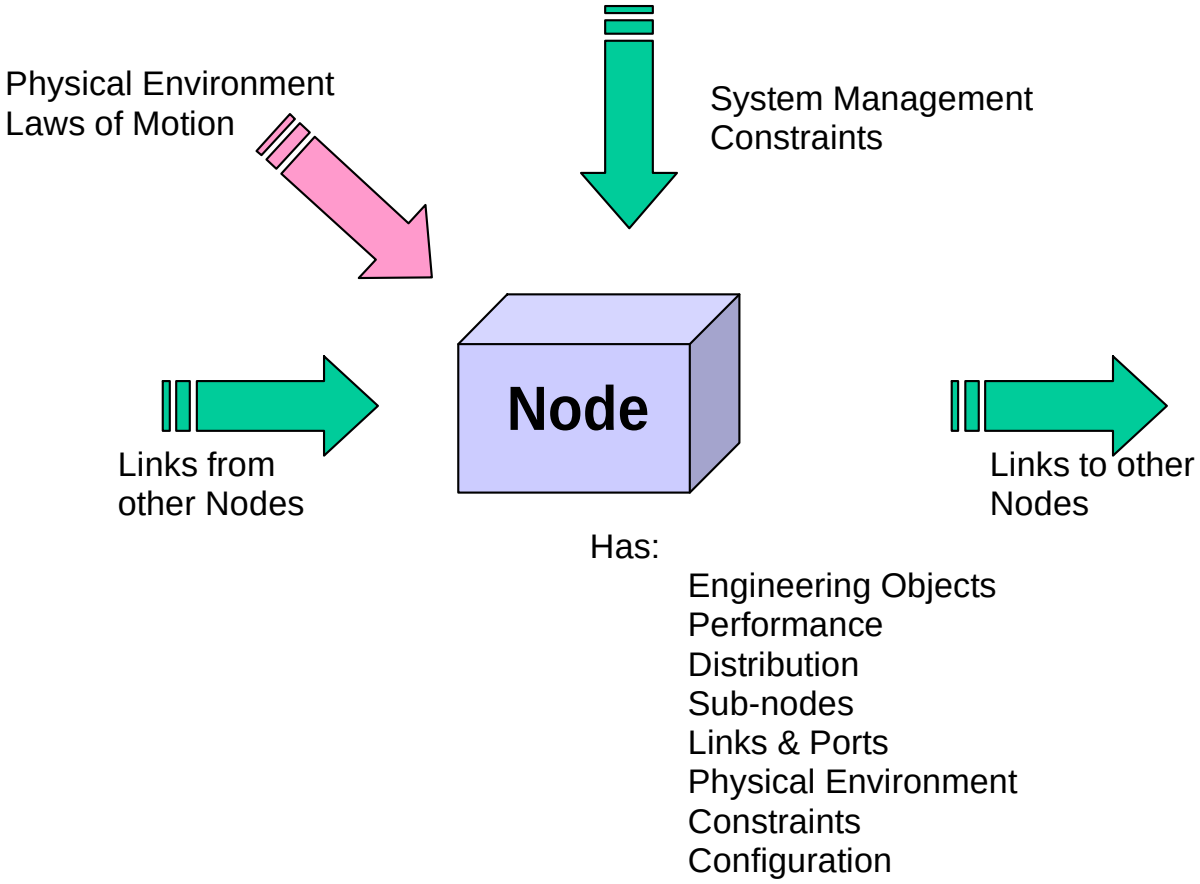


Figure 6-1: Attributes of Nodes

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.

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 are 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.

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 environmental elements 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 are described in the Communications Viewpoint are selected to deal with these behavioral and environmental factors.

6.3.2 OBJECTS AND RELATIONSHIPS

The following elements may appear in the Connectivity Viewpoint:

- Engineering Objects (Nodes, Links, Applications):
  - Nodes (hardware Engineering Objects, provide processing and other computing and data resources, ports, performance and other associated physical behavior):
    - Types: hardware objects, composite hardware objects, ports;
    - 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, Round Trip Light Time (RTLT), 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, 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);

– Information (defined representations of data that are exchanged among Engineering Objects, where formal specifications of data architecture are found in the Information Viewpoint).

NOTE – Physical Nodes and Links in any given space system may include many more types (power, propulsion, thermal) and many more attributes than those shown here. Because focus is upon space data systems these are not directly addressed in this document. See *Toward a Framework for Modeling Space Systems Architectures* (reference [20]) for more information on one such extended approach.

6.3.3 CONCERNS

Concerns are

– the mechanisms and functions required to support distributed interaction 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.

6.3.4 OTHER TERMS

Hardware is 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 are the components of information systems that provide operating instructions for specific task-based applications that run on computing hardware. **Firmware** is 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.

An **Engineering Object** is an implementation or realization of some abstract function. It may be implemented as hardware (Node) or as software (application or software component).

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. The larger discrete items of hardware in a space data system are referred to as nodes. From an architecture description perspective below some level of granularity it may not be useful to describe minor hardware elements as a nodes.

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. They 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 to hardware.

6.4 NODES

Nodes are physical hardware Engineering Objects that provide a set of resources to support the activities of other elements; they are the physical elements where implemented Functional, Information, and Communications Objects are instantiated. Depending upon the design of the specific system, Nodes may contain other Nodes. Thus, a Ground Tracking Network (a Node) may consist of one or more Tracking Stations, which are also Nodes. Each Tracking Station contains a common set of implemented functions for Tracking, Data Handling, Radiometric Data Collection, and Directive Execution (slews, pointing). Similarly, a single spacecraft is a Node that might be composed of Nodes such as instruments, a Command and Data Handling (C&DH) computer, and an RF system. Each of these Nodes implements one or more Functions. Each Node is owned by some Enterprise (see section 4), but it may contain Nodes owned by other Enterprises (e.g., spacecraft owned by Agency A flying instruments owned by Agency B).

Table 6-1 shows typical Nodes that are used in space data systems. Which Nodes are used in any given space data system may differ from system to system, and the following list shows only typical Nodes used in many space data systems.

Table 6-1: 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 data 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 data systems.
Computer	Component of a spacecraft used to process data.
Onboard data system	Subsystem of a spacecraft used to store and manage data. It may not exist as a separate physical object in all spacecraft.
Ground tracking network	Typically a multi-mission subsystem that may be composed of one or more Nodes and one or more tracking stations, used for communicating with spacecraft and performing radiometric measurements against spacecraft.
Tracking station	A ground tracking network subsystem 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 mission operations system facility that is used to plan, control, and monitor spacecraft operations.

Nodes	Description
Instrument control facility	A mission operations system facility 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 mission operations system facility 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 mission operations system facility 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 mission operations system facility 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 data systems.
Data analysis facility	A mission operations system facility 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 data systems.

6.5 LINKS

Links provide interconnections between Nodes for communication and coordination. Nodes are connected together using Links that have specific behavioral, functional, and physical attributes. These attributes include performance and physical constraints upon the Links (e.g., maximum bandwidth or data rate, spacecraft and planetary motion, physical distance and RTLT, environmental noise, interference, occultation, pointing, bus length limitations, etc.). Links are connected to Nodes at a Physical port. A single Link may support multiple logical connections between different functions that are hosted on the connected Nodes. For example, a space link may carry separate command, telemetry, software upload, file transfer, and monitor data channels.

Table 6-2 shows typical Links that are used in space data systems. Which Links are used in any specific space data system differs from system to system, and the following lists only typical Links and their attributes that are considered in many space data systems.

Table 6-2: 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).	– Continuous vs. episodic connectivity – Pointing and view periods – Frequency band – 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.	– 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.	– Single vs. multiple access – Redundancy – Bit rate

6.6 EXAMPLES OF SPACE DATA SYSTEMS DESCRIBED WITH CONNECTIVITY VIEWS

Figure 6-2 shows the Nodes used to support Mission A, as shown in figure 4-2.

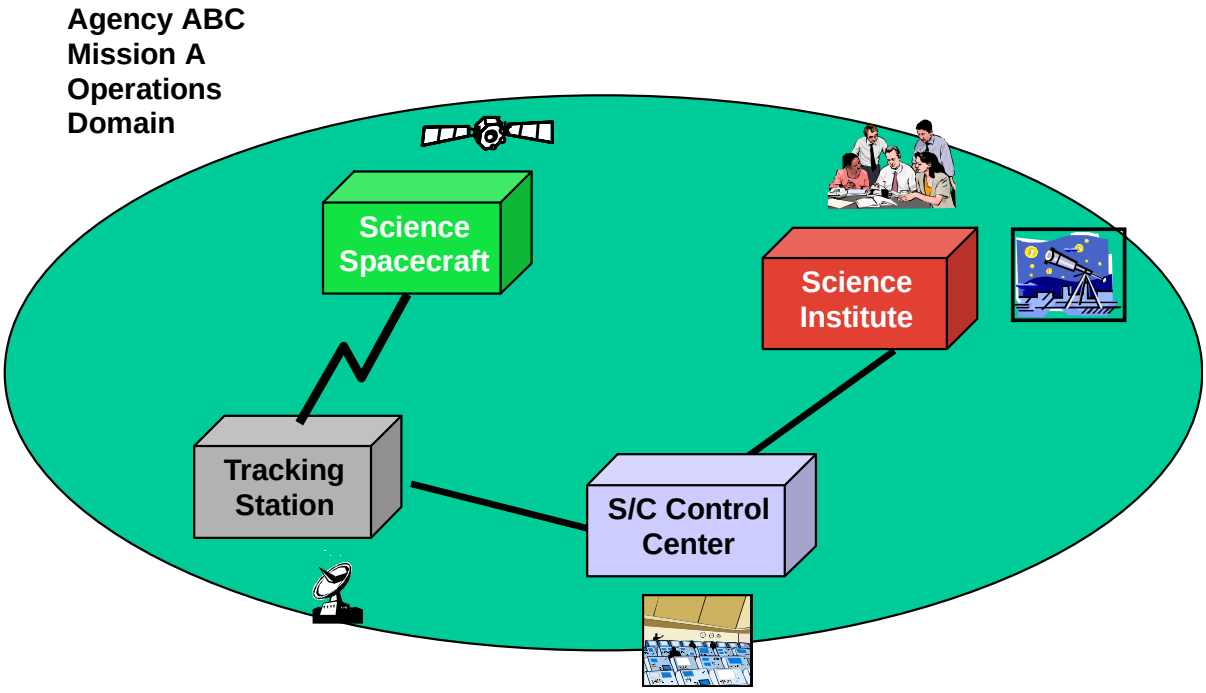


Figure 6-2: Example of Connectivity View (Nodes for Mission A)

Figure 6-3 shows Nodes supporting Mission Q.

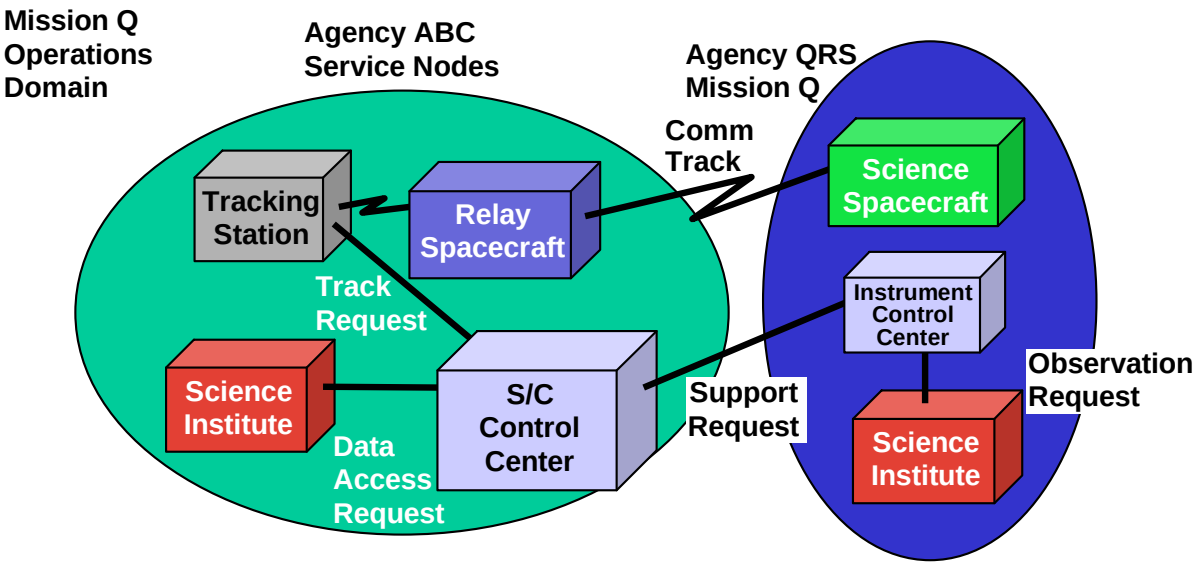


Figure 6-3: Example of Connectivity View (Nodes for Mission Q)

Figure 6-4 shows one possible decomposition of the Nodes used for Mission A (shown in figure 6-2) into example component Nodes. For clarity the ordering of nodes is the same as shown in figure 6-2 and the nodes are colored like the corresponding Functional Objects in figure 5-2, but there is no requirement to do this nor is there anything special about these colors. In reality the as-implemented functions from figure 5-2 might be allocated to different hardware nodes than those shown, and no one-to-one mapping can be assumed.

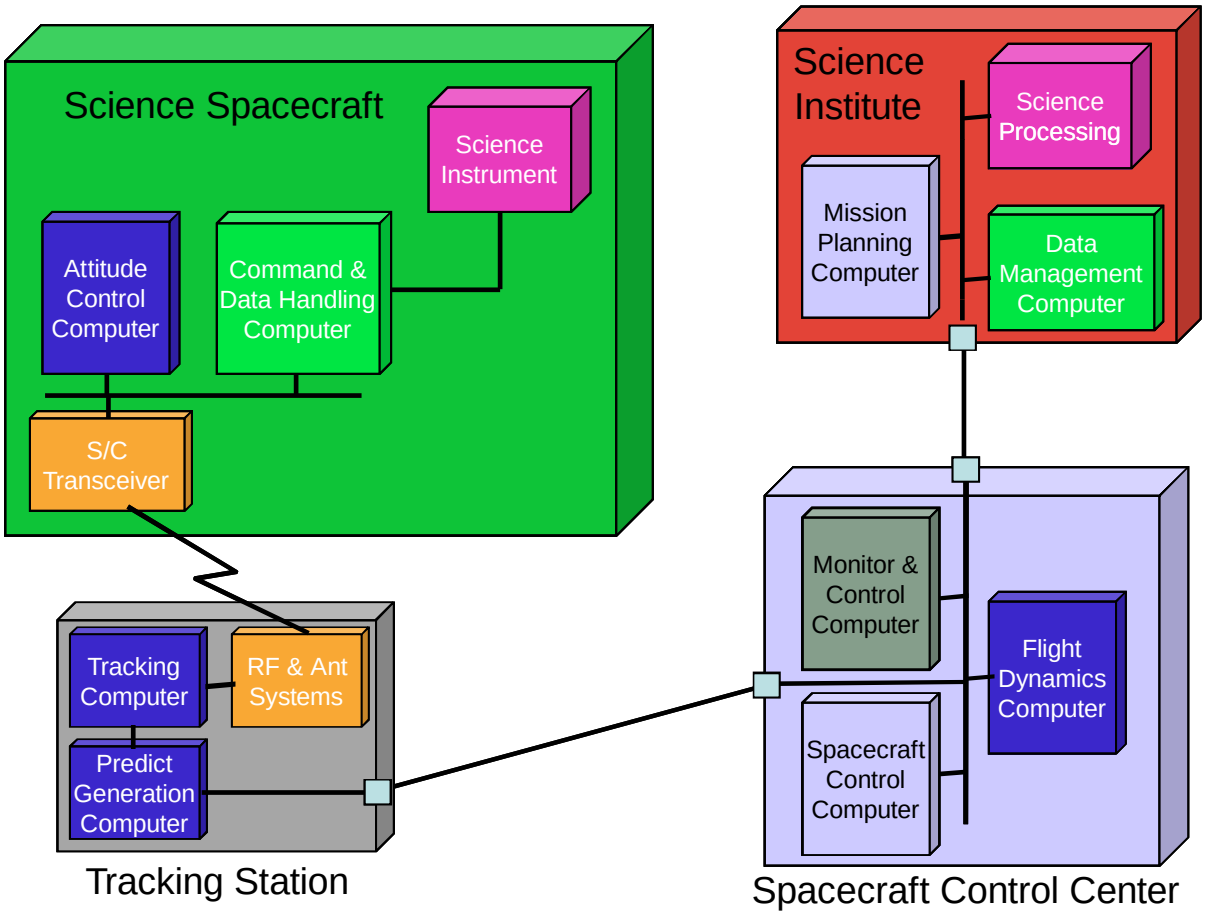


Figure 6-4: Example of Connectivity View (Node Decomposition)

Of course, the nodes shown in figure 6-4 may be further decomposed into lower-level nodes, with their own internal links where this level of detail is required. Many system-engineering disciplines provide a hierarchy of names to describe different levels of decomposition of Nodes within a system. IEEE 1220 (reference [3]) defines the terms system, product, subsystem, assembly, component, and subcomponent, but other terms for system decomposition may be appropriate; none is prescribed here. Similarly, this approach supports definition of Systems-of-Systems (SoS) views, but no specific recommendation is made other than to suggest starting at the highest level overview and then using successive decomposition and clear specification of interfaces at each level. In Architecting Principles for Systems-of-Systems (reference [21]) 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.

As an example of how allocation works, figure 6-5 shows how the functionality defined by the set of example Functional Objects shown in figure 5-2 might be allocated to the Nodes that were shown in figure 6-2. This is a Connectivity View diagram showing how the Functional Objects, implemented as Engineering Objects, might be allocated to Nodes. On a more detailed view the explicit choice of implementation options as hardware or software

Engineering Objects may be shown. Once the combination of the performance support requirements of the implemented Engineering Objects and the performance capabilities of the Nodes and Links has been defined, analysis of the end-to-end performance of the system may be determined.

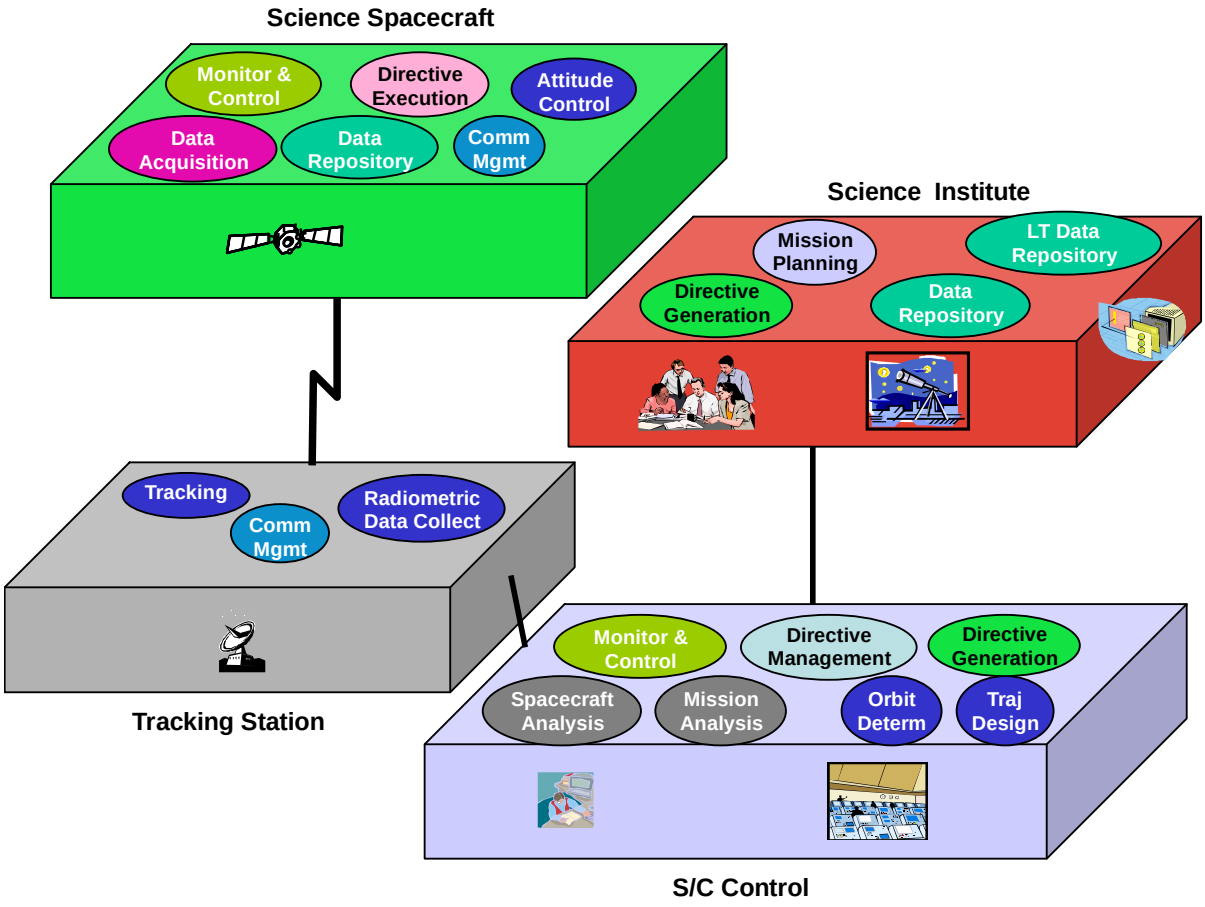


Figure 6-5: Example of Connectivity View with Allocated Engineering Objects

If figure 6-5 were redrawn to show an autonomous spacecraft, some of the Planning, Directive Generation, and Monitor and Control functions might be moved on-board. Exploring the implications of these allocation options on system functionality, performance, and support requirements is possible once all of the elements of the Functional View have been identified and the allocations and engineering 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 C&DH processor, or to implement a hardware compressor that might be a board integrated into the flight data recorder. Both of these would implement the identical functional flow as shown in figure 6-6(a), but the connectivity views would look significantly different, as shown in

figures 6-6(b) and 6-6(c), and the performance characteristics would also be significantly different.

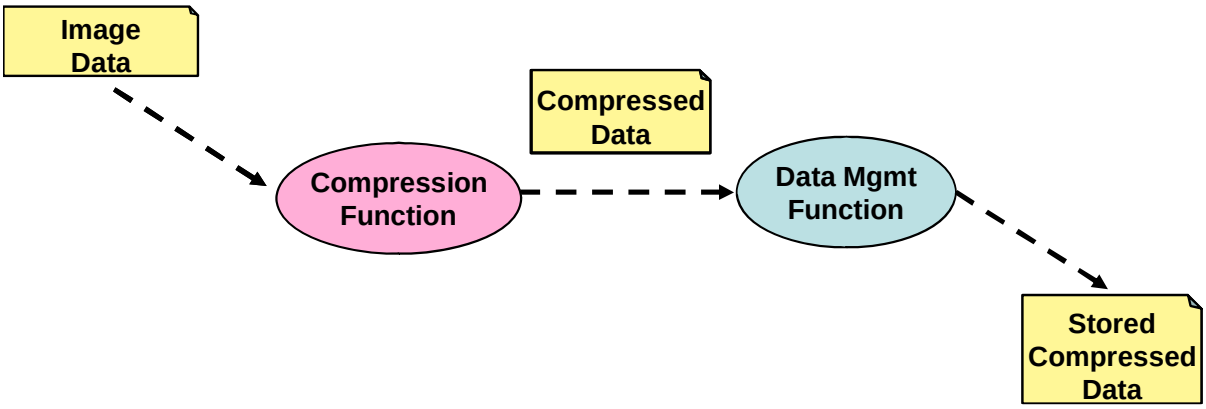


Figure 6-6(a): Functional View of Image Compression

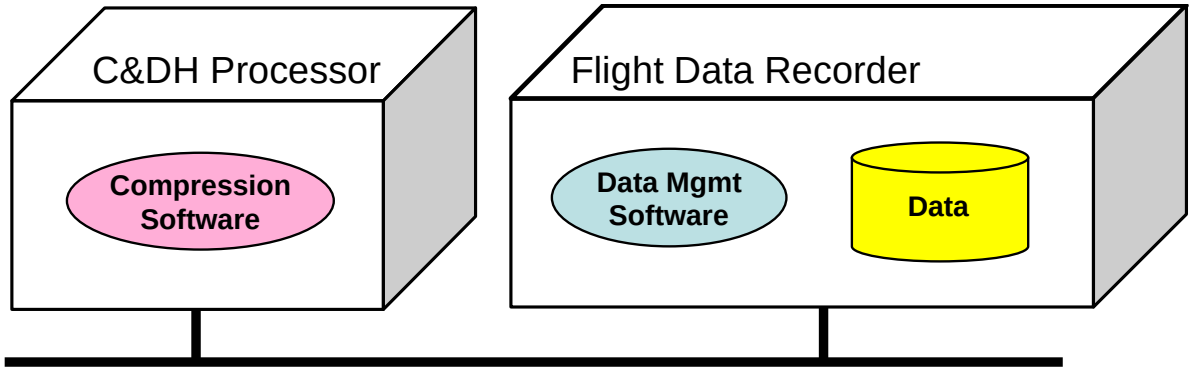


Figure 6-6(b): Connectivity View of Software Compression Approach

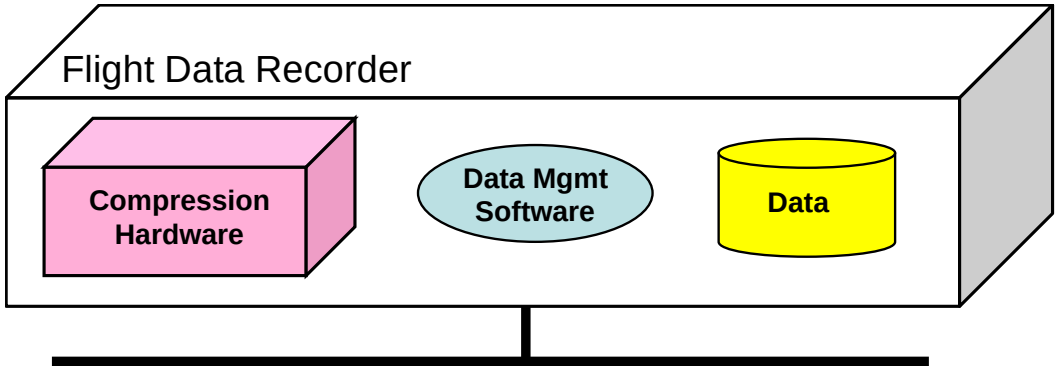


Figure 6-6(c): Connectivity View of Hardware-Only Compression Approach

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

6.7 SECURITY ISSUES IN THE CONNECTIVITY VIEWPOINT

In the Connectivity Viewpoint security issues are dealt with by the physical elements that are used to implement security policies and barriers. These might include: routers and firewalls, hardware encryption devices, and possibly physical boundaries such as shielded rooms or air gaps. The protocol entities that may implement elements of security functionality, such as security protocols or routing filters, will be addressed in the Communications Viewpoint.

7 COMMUNICATIONS VIEWPOINT

7.1 OVERVIEW

The Communications Viewpoint defines the layered sets of communications protocols that are required to support communications among the software 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 protocols and their construction, and to address these aspects of space data systems.

NOTE – This Viewpoint is related to both the Engineering (implemented functionality) and Technical (standards) Viewpoints of RM-ODP. It is addressed separately in RASDS because of the need for specialized protocols to deal with the physical challenges of designing space data systems. It is also the first of the Viewpoints where the lower five layers of the ISO seven-layer communications stack, rather than the upper ones, are addressed. This Viewpoint is orthogonal to the other, upper-layer, Viewpoints, where multiple perspectives on the applications in a distributed system are provided. Here discussion concerns layered communications protocols and the technical approaches used to describe them.

7.2 CONCEPTS

The **Communications Viewpoint** is a space data system 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, which is orthogonal to the first three Viewpoints, provides details on layers one through five of the ISO seven-layer model. The first three RASDS Viewpoints are directly related to the top, or Application layer, of the ISO Basic Reference Model (ISO-BRM, reference [14]) and the Information Viewpoint is most closely related to the Presentation layer of the ISO-BRM model.

In the Communications Viewpoint, a space data system is depicted with Communications Objects that are called Protocol Entities for alignment with ISO-BRM terminology. For context these are often shown along with representations of the Nodes, Links, and software Engineering Objects that are defined more fully in the Connectivity Viewpoint. The Communications Viewpoint describes the protocols that are required for the software Engineering Objects actually to 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 data 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. The Engineering Objects that use protocol services may be implemented in hardware or software, and the Protocol Entities themselves may be implemented in hardware or software.

Note that some Nodes in a space data system may only have communications functions. There may be Nodes in space data systems that contain only Protocol Entities (without other Functional elements); a Router or Bridge is an example.

While a full communications stack (application, transport, network, data link, physical) is often used in the terrestrial subdomain of a system, in many space subdomains only the lower Data Link and Physical layers may be specified, with applications providing any upper-layer functions that are required.[1]

7.3 CHARACTERISTICS OF COMMUNICATIONS OBJECTS

7.3.1 GENERAL

The interfaces of communications objects (Protocol Entities) are shown in figure 7-1.

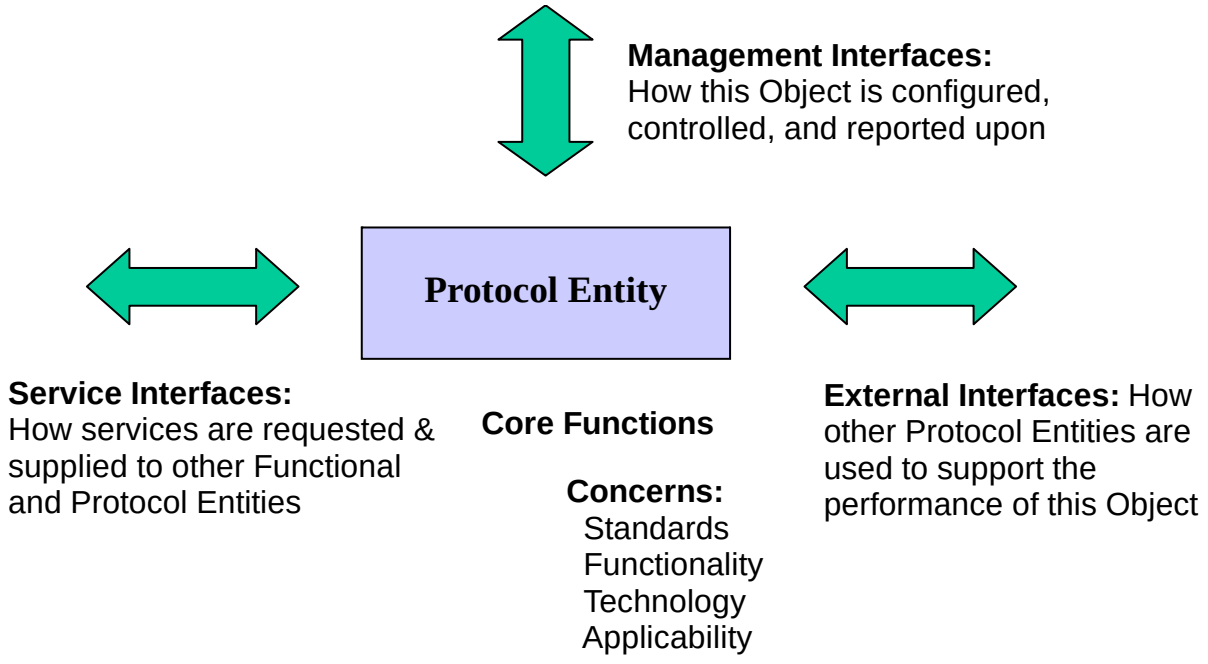


Figure 7-1: Attributes of Protocol Entities

[1] Management approaches that are relevant to the Communications Viewpoint will be addressed in a later issue of this document. Security issues will be identified here, but dealt with in detail in the Security Architecture document.

The interface of a Protocol Entity with an Engineering Object or other Protocol Entity is described by requests, indications, responses, and confirmations. The services provided by a Protocol Entity are made available at its Service Interface, which is called a Service Access Point or SAP.

Protocol Entities communicate with peer Protocol Entities, either directly or indirectly through other lower-layer Protocol Entities. The interactions between peer Protocol Entities are described by exchanges of Protocol Data Units (PDUs), and the activities 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 entity.

7.3.2 OBJECTS AND RELATIONSHIPS

The following elements may appear in the Communications Viewpoint:

- Protocol Entities (elements that implement specific protocols, with 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), API (where appropriate), standard reference identifier, standards organization;
- Protocol design specification elements (PDU description, state machine or table description), reliability (acknowledged, unacknowledged, selectively acknowledged), other design views of the communications protocol or protocol stack;
- Nodes and Links (representations of physical elements from the Connectivity Viewpoint, for context);
- Software Engineering Objects (representations of implemented functions from the Connectivity Viewpoint, for context).

7.3.3 CONCERNS

Concerns are

- the choice of communications and data transfer standards in the system;
- the end-to-end communications protocol functionality and reliability;
- implementation of the protocol specifications and the services they provide;
- the relevant interfaces and interactions;
- interaction of protocol design with environmental constraints;
- support for design, evaluation of suitability, and integration into the rest of the system.

7.3.4 OTHER TERMS

Most of the definitions in this section are drawn directly from the ISO-BRM.

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 **Protocol Data Unit** (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 **Service Access Point** (SAP) is the point at which (N)-services are provided by an (N)-protocol-entity to an (N+l)-protocol-entity.

An **Application Programming Interface** (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.

7.4 PROTOCOL ENTITIES

Table 7-1 shows examples of typical Protocol Entities used in space data systems. Not all combinations of these protocols are valid. See *Overview of Space Link Protocols* (OSLP, reference [16]) for more information about which combinations are acceptable. Protocols are normally associated with some layer in the ISO-BRM, but these layers are not identified here except by reference to protocol type. This table is representative, but does not include all of the available or applicable protocols for use in space data systems.

Table 7-1: Typical Protocol Entities

Protocol Entities	Type	Description
Space Message Transfer	Messaging	Provides message transfer services between functions
CCSDS File Delivery Protocol (CFDP)	File transfer protocol	Transfers files over one or multiple space links
SCPS File Protocol	File transfer protocol	Transfers files over space links
File Transfer Protocol (FTP)	File transfer protocol	Transfers files over Internet protocols
SCPS Transport Protocol	Transport protocol	Provides end-to-end communications over space links
Transmission Control Protocol (TCP)	Transport protocol	Provides end-to-end communications in Internet
Space Packet Protocol	Network protocol	Provides routing through a network involving space links
SCPS Network Protocol	Network protocol	Provides routing through a network involving space links
Internet Protocol	Network protocol	Provides routing through Internet
TM Space Data Link Protocol	Data link protocol	Provides communications over a point-to-point space link
TC Space Data Link Protocol	Data link protocol	Provides communications over a point-to-point space link
AOS Space Data Link Protocol	Data link protocol	Provides communications over a point-to-point space link
TM Synchronization and Channel Coding	Channel coding	Provides mechanisms for data synchronization and error control

Protocol Entities	Type	Description
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 over a point-to-point space link
Point-to-Point Protocol (PPP)	Data link protocol	Provides communications over a point-to-point link for Internet protocols
CCSDS RF and Modulation	Physical protocol	Define physical frequency bands and power or Bandwidth Efficient Modulation (BWEM) to transmit and receive signals over a space link

7.5 EXAMPLES OF SPACE DATA SYSTEMS DESCRIBED WITH COMMUNICATIONS VIEWPOINT

Figure 7-2 shows a set of Protocol Entities that support the communications between two software Engineering Objects (Data Acquisition and Data Monitor). These applications are contained in two Nodes (Payload and Ops Center), and the communications path includes two other typical Nodes (which are the on-board C&DH and Ground Station). The Communications View, as in this case, may include representations of Nodes and Application entities, or it may just show the communications stacks by themselves and the relationships between peer Protocol Entities.

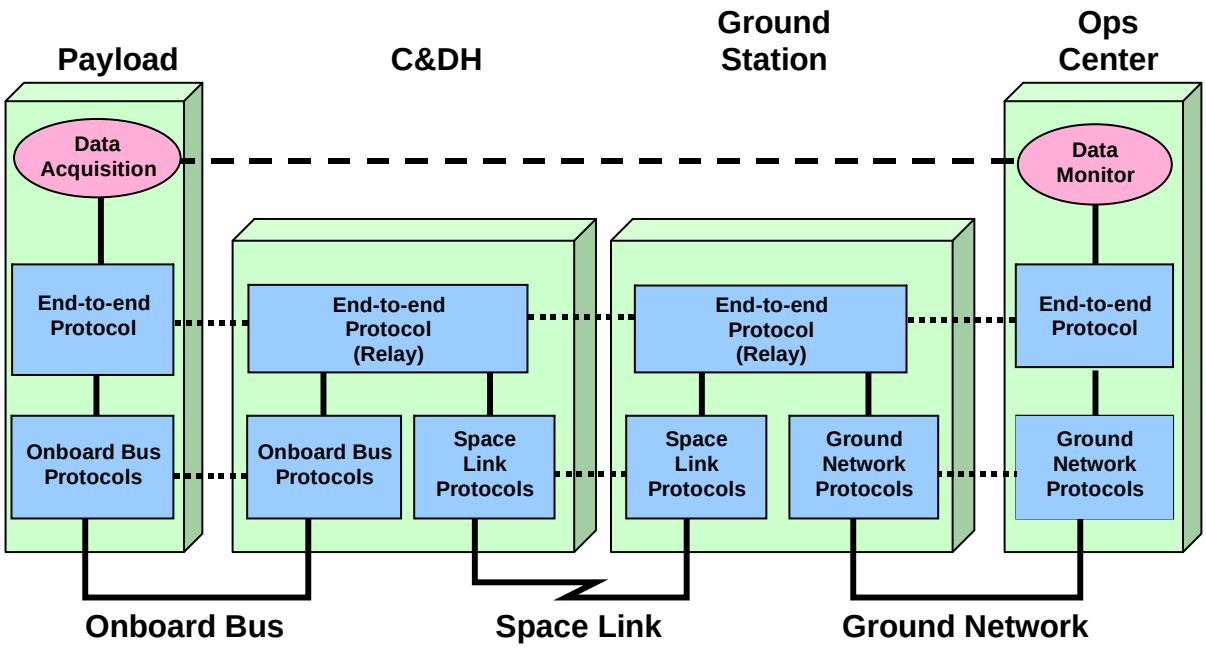


Figure 7-2: Example of Communications View Showing Protocol Stack and Nodes

7.6 PROTOCOL REPRESENTATIONS IN THE COMMUNICATIONS VIEWPOINT

A Communications View may just show the Protocol Entities in a 'black box' view, with only the SAPs and some representation of a peer protocol entity indicated. Where required, more engineering details of the protocol specification may also be represented by showing the flow, 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 styles are in use, and a specific representation for PDUs is not defined here. However, figure 7-3 provides a useful example from *Space Packet Protocol* (CCSDS 133.0-B-1). The Internet Request for Comment (RFC) that documents protocols mandates a similar representation of PDUs, but expressed in an ASCII text form. The important aspect of this is to show the data structures, sizes of fields, and to identify the control and data elements in the PDU.

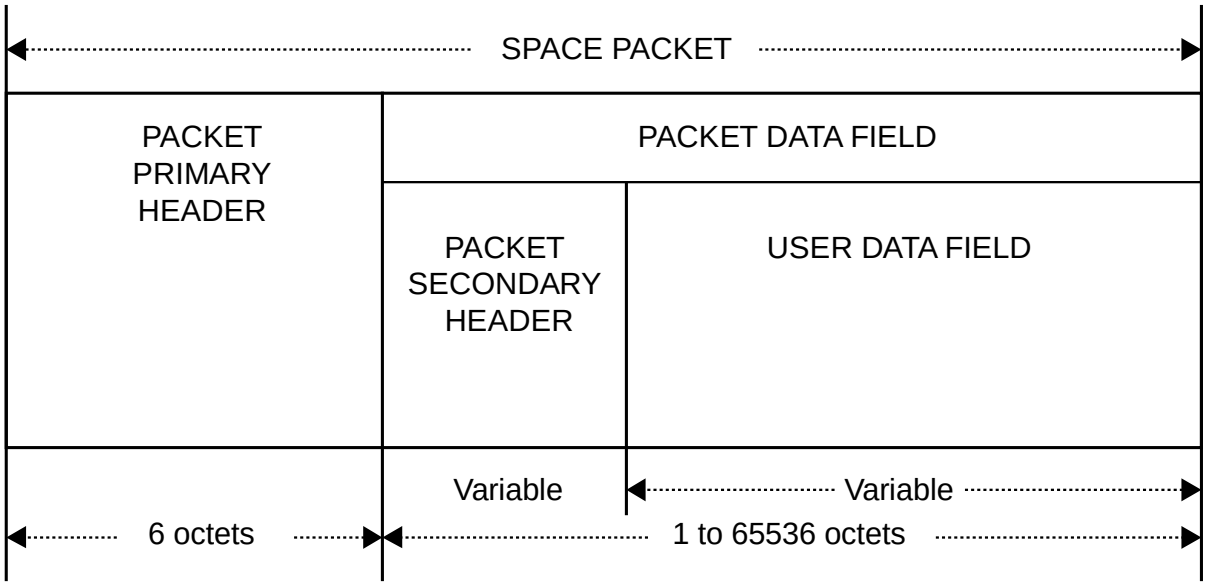


Figure 7-3: PDU Example, Space Packet Protocol

In addition to description of a set of PDU structures, there is usually need for specification of 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 7-4, which is a simple one taken from *Space Link Extension—Return Channel Frames Service Specification* (CCSDS 911.2-B-1), or it may be described using a state table or even narrative text, as is done in a number of CCSDS and Internet documents. RASDS does not require use of any specific form to describe protocol entity behavior, but a state machine or state table specification is preferred, as these are typically more precise and more readily converted to code.

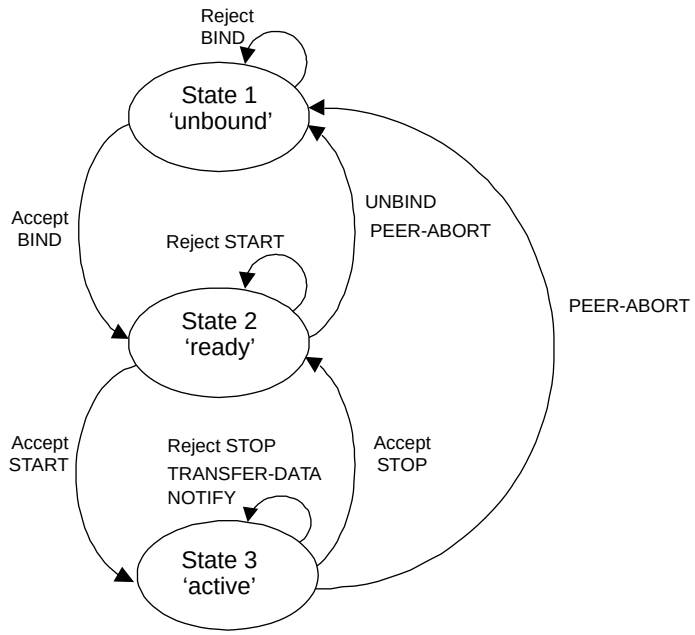


Figure 7-4: Example State Machine Diagram—SLE RCF

Regardless of the representation used, for interoperability the complete specification of a protocol must include both the PDU definitions (the data that are exchanged between peer Protocol Entities) and 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 with no effect on interoperability as long as the state machines and PDUs are correctly implemented.

7.7 SECURITY ISSUES IN THE COMMUNICATIONS VIEWPOINT

Certain functions for implementing data system security may be allocated to the Communications Viewpoint. These will typically include Network layer security protocols, Data Link layer encryption, and spread-spectrum or related jamming avoidance approaches. The details of when to apply these approaches will be described in the Security Architecture (reference [4]) and Threat Assessment (reference [18]) documents, and the specific details of how to provide these capabilities are defined in other standards and specifications.

8 INFORMATION VIEWPOINT

8.1 OVERVIEW

The Information Viewpoint describes 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 definition aspects of space data systems. Representations of the Information Objects that are fully defined in this Viewpoint appear in other Viewpoints. They are managed (that is, stored, located, accessed, and distributed) by information infrastructure elements and also shown as being passed among enterprise, functional, and application entities.

NOTE – The Information Viewpoint corresponds directly to the information viewpoint of RM-ODP, but without reference to the static and dynamic views of data and its transformations. This abstract view on the system will be expressed during implementation (RM-ODP Engineering and Technology Views) using concrete specifications. In the case of Information Objects, there will be a set of abstract data specifications and a related set of data elements that are bound to some particular language or framework.

8.2 CONCEPTS

The **Information Viewpoint** specification of a space data 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 data 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 associated metadata, and information views may define the relationships among Data Objects, rules for their use and transformation, and policies on access.

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.

8.3 CHARACTERISTICS OF INFORMATION OBJECTS

8.3.1 GENERAL

The attributes of Information Objects are shown in figure 8-1. Unlike most other objects considered in this document, Information Objects do not have input or output interfaces. Information Objects have schema that describe their structure, rules for use and transformation, and policies on access and permanence.

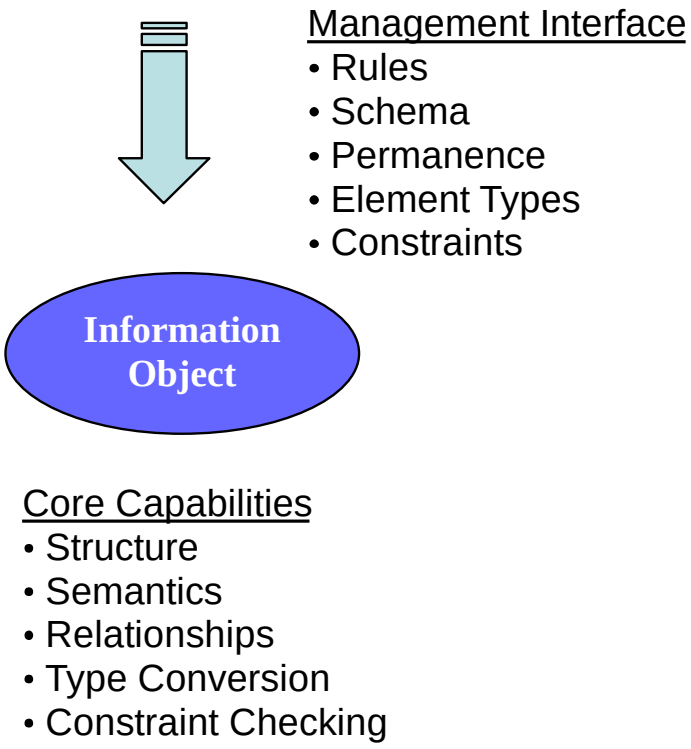


Figure 8-1: Attributes of Information Objects

NOTES

1 Detailed descriptions of this view and the means for expressing them may be found in a separate document, the *Reference Architecture for Space Information Management* (reference [5]).

2 RM-ODP describes static, dynamic, and snapshot aspects in their Information Viewpoint. In RASDS 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.

8.3.2 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, meta-model;
  - Attributes: name, type, length, structure, syntax, semantics, permanence, provenance, realized by, rules, policies;
- relationships (information object aggregates, transformations);
- constraints (type checking rules, permanence, policies).

8.3.3 CONCERNS

Concerns are

- the structure and semantics of information and information management in a RASDS system;
- the rules and constraints on information transformations and permanence in a RASDS system.

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

Abstract Data Architecture Meta-Models 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 **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 in order to participate in system activities.

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.

8.4 INFORMATION OBJECT VIEWS

Information Objects in a system are often represented from several different views, ranging from very abstract to quite concrete. 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 8-2.

The Information Viewpoint is focused primarily upon the more abstract representations of Information Objects, or what might be described as abstract data architecture and data models, as shown in figures 2-7 and 8-2. This includes 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 are also more concrete representations of Information Objects as they are implemented within the system. These are shown in RASDS as correspondences in other Views, such as the Enterprise or Functional Views.

The most concrete representations of an Information Object are the Actual Data Objects, or the set 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 both the semantic content and a description of the syntax.

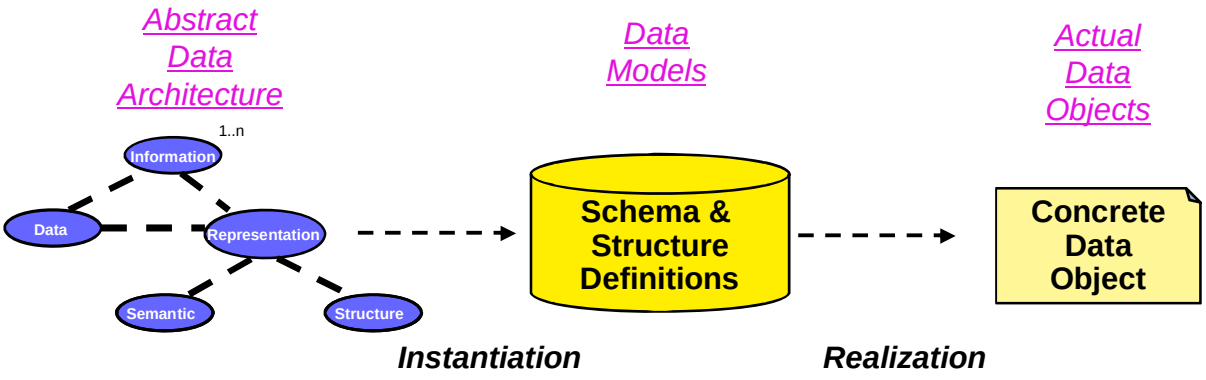


Figure 8-2: Information Object Representations

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 the data elements and their relationships. This may be stored in a machine-accessible format, or it may be defined in a document. 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, i.e., is related to, is part of, or is used by. Increasingly, formal description mechanisms, such as an ontology, are being used to permit machine access to all these levels of abstraction.

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 may appear in Functional or Connectivity Views, as the system is engineered. 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.

8.5 EXAMPLE OF SPACE DATA SYSTEM FUNCTIONS WITH INFORMATION VIEWPOINT

Figure 8-3 shows the relationships among some typical space data 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 ‘dog eared box’ 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 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.

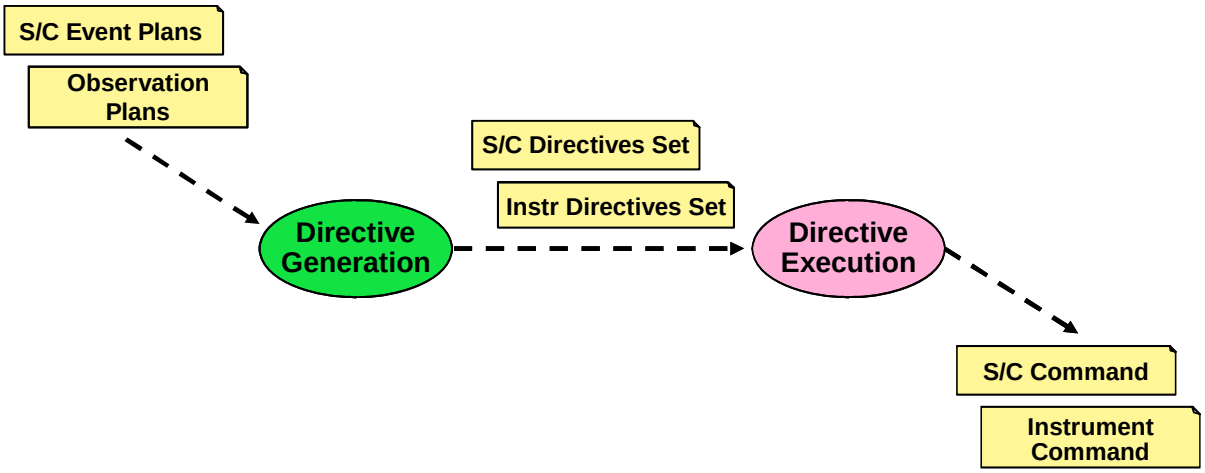


Figure 8-3: Example of Functional View with Representation of Information Objects

The Reference Architecture for Space Information Management (RASIM, reference [5]) has a much more complete treatment of the definition and use of information objects. It also provides a more complete description of the information management Functional Objects introduced in 5.6. The interested reader is directed to this document.

9 DERIVING OTHER VIEWS FROM THE BASIC VIEWPOINTS

9.1 GENERAL

RASDS provides a general set of viewpoints that may be used to describe most aspects of space data system architectures. However, depending on the set of concerns that must be addressed during design of a particular space data system, other Views may need to be constructed from the objects and relationships described in the five basic Viewpoints. This may be done either by defining correspondences to objects defined in two or three basic Viewpoints or by defining new classes of objects or relationships using the rules from one of the basic RASDS Viewpoints. Moreover, adequate visibility into particular design concerns can often be provided 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.

9.2 CROSS SUPPORT SERVICE VIEWS

Many of the data system standards defined by CCSDS are intended to be used for inter-Agency cross support, the need for which is typically first identified as an Enterprise-level agreement. Where detailed relationships between organizations or the facilities that they own and operate are of particular concern in a space data system, an Enterprise View will be used. In order to represent this a related set of Views may need to be constructed to describe the full cross support service interoperability from different perspectives. A Functional View may be used to show which functions are provided by the support organization and which are provided by the using organization. Additionally, a detailed view of the 'as built' interfaces, which is required for real interoperability, may be constructed by developing a Communications View showing protocols and representations of the required Engineering Objects.

Enterprise P provides a service to support a mission of Enterprise Q as shown in figure 9-1(a).

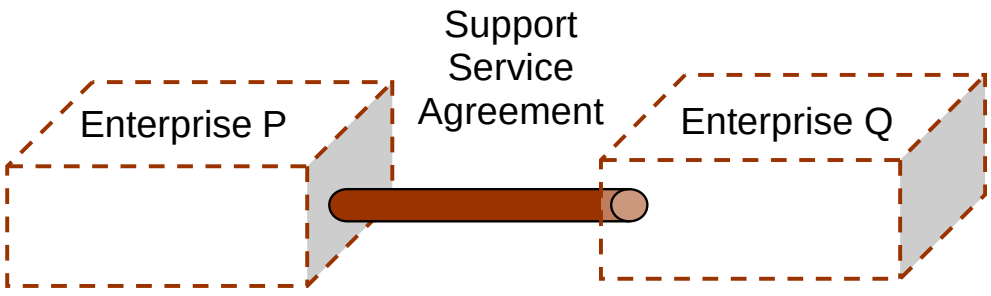


Figure 9-1(a): Enterprise View of a Cross Support Service

The service provided by Facility P (owned by Enterprise P) is implemented by exposing a service-providing interface of Application A (provider) to a service client of Facility Q (user) in a Connectivity View diagram, figure 9-1(b).

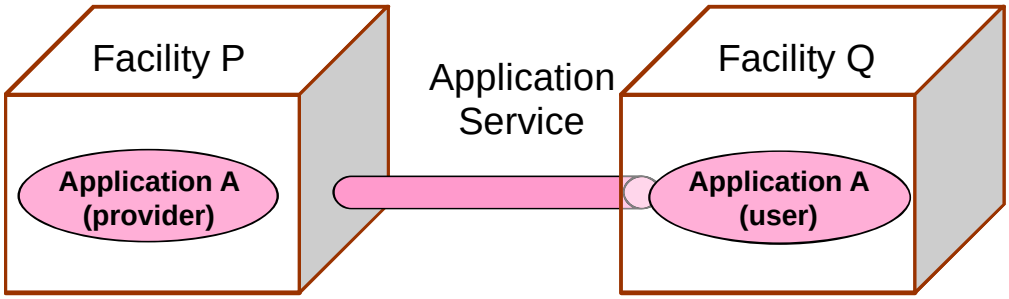


Figure 9-1(b): Connectivity View of a Cross Support Service

In this diagram, a pipe is drawn between a pair of Applications (Engineering Objects) representing the cross-support application service. One Application provides a service to the other Application. In other words, one Application is a service provider, and the other is a service user. A pipe represents a user-provider relationship in the same layer. In this way, the set of services provided by one Application to another Application can be described.

To support communications between these two applications, communications protocols are used as shown on a Communications View figure 9-1(c). The relationships between N-layer protocol entities are not shown here as a Service Interface (using a pipe); they are shown here as peer-level protocol associations when this level of detail is needed. The implementation of the cross-support service is between the two user and provider Application objects, but it requires the support of (and complete specification of) the underlying protocols in order to be implemented and fully interoperable.

The reason these Views are not new basic Views is that no new Object or relationship is introduced in any of them. These cross-support service interfaces are really just the exposed set of interfaces of applications layer Engineering Objects. All application Engineering Objects have interfaces; the ones that are exposed between facilities owned and operated by (the same or) different Enterprises are called cross-support service interfaces. These interfaces may exist and be documented between ground entities, between space and ground elements, or in a space-to-space context.

NOTE – The RM-ODP Engineering Viewpoint specifications introduces the concepts of binders and stubs to describe the detailed engineering of service interfaces and how they operate. These provide a basic CORBA-like model of service interactions, but other service interactions, such a client/server, peer-to-peer, and Service Oriented Architectures (SOA) are also possible. Providing detailed guidelines for these is outside the scope of this document, and readers are encouraged to explore the RM-ODP engineering concepts as well as alternative related approaches (see references [1], [6], [9] and [25]).

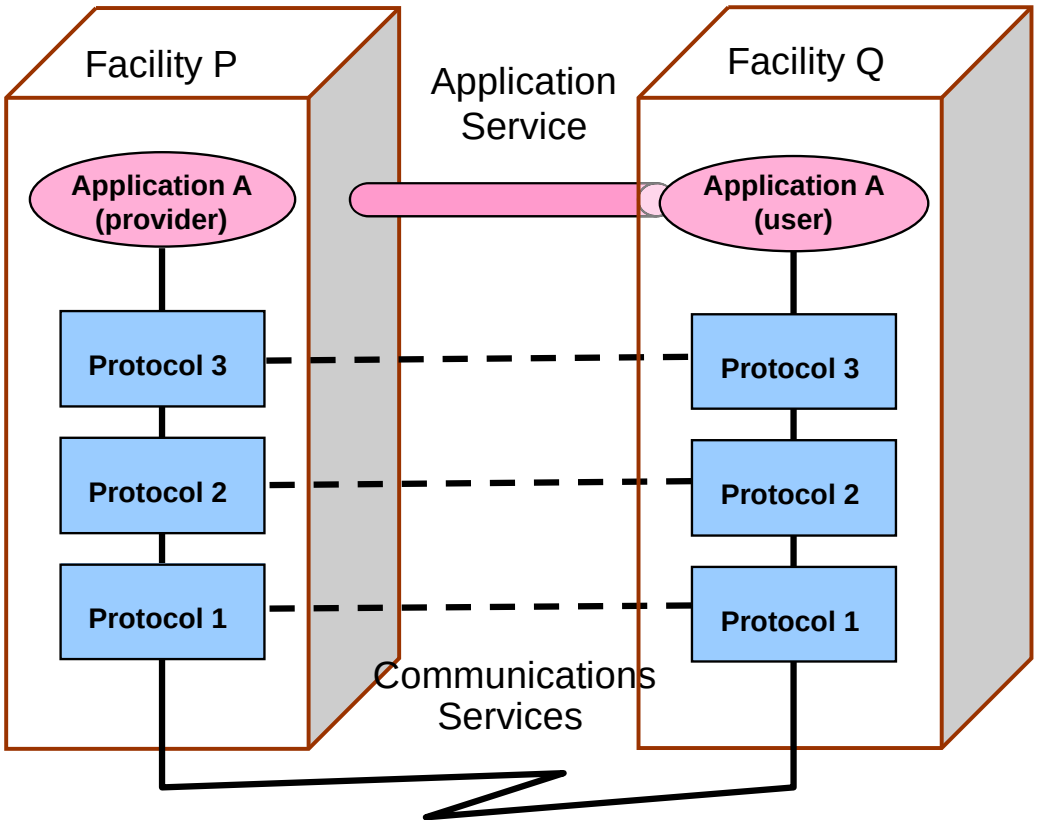


Figure 9-1(c): Communications View of a Cross Support Service

9.3 LAYERED VIEW EXAMPLE

Another example of an extended Connectivity or Communications View is a Layered View, in which layers of Engineering Objects and possibly Protocol Entities are shown. If a group of Objects provides services of a certain class of functionality to another group of Objects with different functionality, each group of Objects may be thought of as a layer. This particular kind of diagram is often used to show the hierarchical configuration of layers of Objects in a space data system.

A simple example is shown in figure 9-2, which shows three layers that include Engineering Objects (Applications), Distributed Computing Service Protocol Entities that provide services to isolate Applications from more basic communications protocols, and the communications protocols themselves. The Communications Layers can be further decomposed into sublayers, as is shown here. The Distributed Computing Service Layer is represented as a Protocol Entity because it provides ISO-BRM layer-7 communications services and does not perform any essential information transformations, just data transfer and marshaling services.

A lower layer (N-1)-Layer in the protocol stack is said to provide services to the (N)-Layer. The upper (N)-Layer uses the services of the lower (N-1)-Layer and the (N)-Layer may also provide services to the (N+1)-Layer, or directly to an Application.

In figure 9-2, a small oval represents a service interface that is exposed to a higher layer. The ISO-BRM calls this a Service Access Point (SAP). From an application program point of view this is most often defined as an Application Program Interface (API). APIs are used to define the implemented interfaces exposed from one layer to another. The ISO-BRM sometimes represents the SAP for a protocol layer as a small oval, as shown here, but a pipe representation, as used in figure 9-1(a), may also be adopted if it is useful.

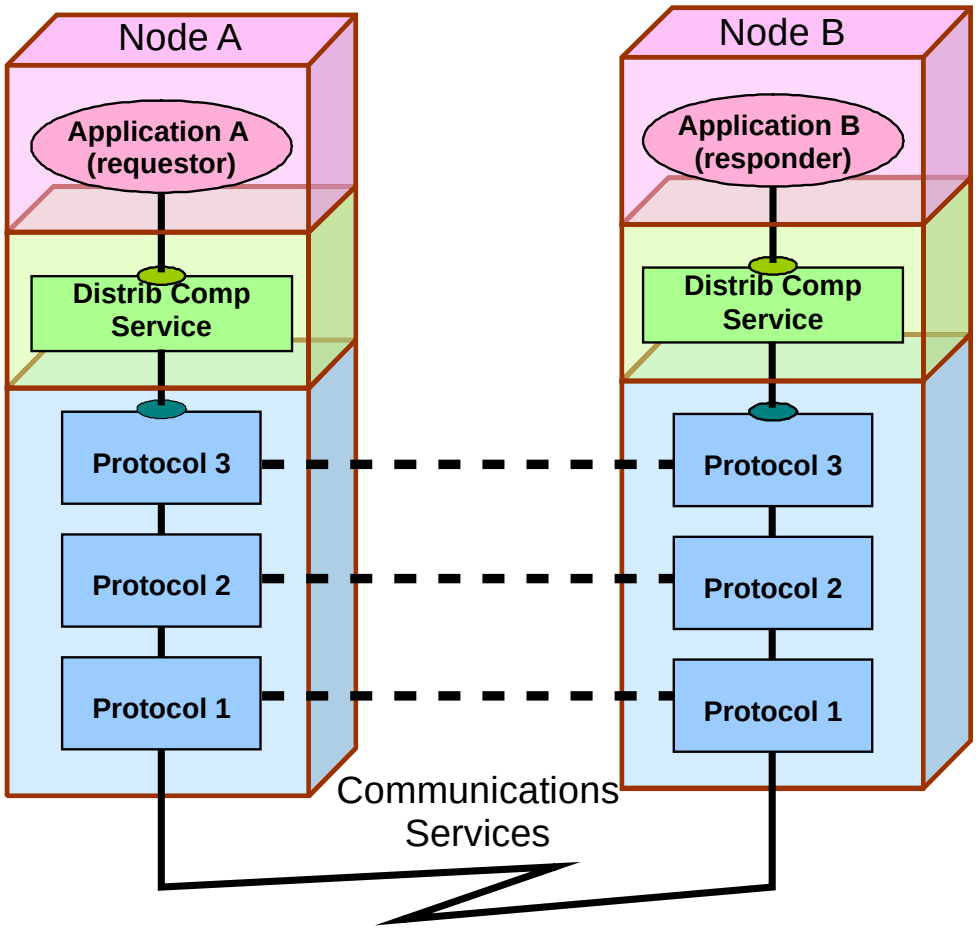


Figure 9-2: Simple Example of Layered View

Layered diagrams are most often used to distinguish between the application (layer 7) objects and communications protocol elements (layers 6 through 1). However, layered diagrams may be useful in other contexts as well, such as distinguishing applications logic from distributed computing infrastructure or from support libraries or operating systems routines. This is shown in this Communications View by indicating layering as separate colored elements within a Node. Any other approach that manages to represent the distinctions between layers (lines, colors, separate boxes) might also be adopted if it is clear and unambiguous. Layering may also be shown in Functional Viewpoint diagrams as well where useful. A layered Functional View might show the distinction between general support functions or infrastructure elements that are broadly used and the domain-specific application

functions that use them. Operating system, storage, and information management functions are other examples where a layered View might be useful.

This View is not a basic View because no new Object is introduced in this View.

9.4 EXAMPLE MAPPING FUNCTIONAL TO CONNECTIVITY VIEW

Abstract elements in a Functional View may be mapped to a Connectivity or Communications View to show how the functions are allocated and implemented. Such diagrams may be used when exploring engineering alternatives, such as evaluating how to implement the same basic functionality using a software, hardware, or a combination. As an example, consider how the telemetry framing and coding functions might be performed.

Figure 9-3 shows the basic functional flow of packets into coded symbols.

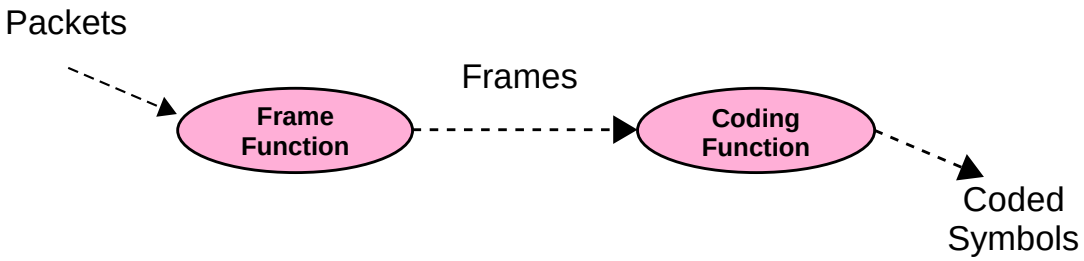
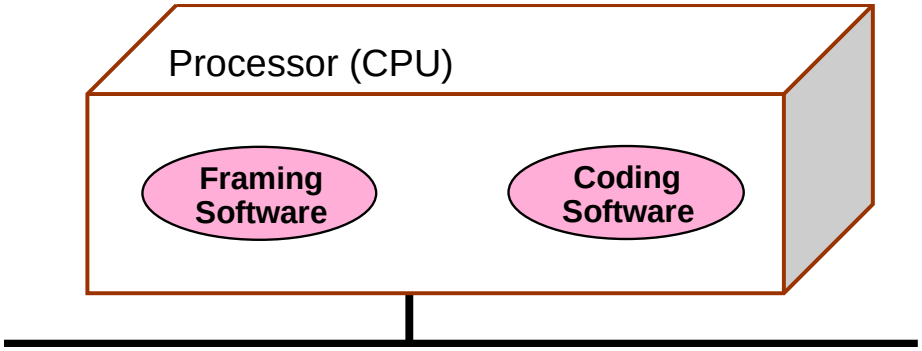


Figure 9-3: Example of Functional View of Framing and Coding

During the engineering of the system that is to implement these functions (and many others) the designer may consider the trades of where to perform these functions and whether to use hardware, software, or some combined approach. The trades will involve evaluation of cost, available processor performance, required bandwidth, and mass, among others. Figure 9-4 shows two alternative engineered approaches, one using software running on a computer and the other using just hardware.

These two approaches provide the same basic functionality but differ in terms of the loading placed on the system CPU and the throughput of the resulting component. It could be said that the trade is not based upon functionality (which is the same) but upon capability (throughput) and the related resource load, balanced against cost and mass.

Software Only Connectivity View



Hardware Only Connectivity View

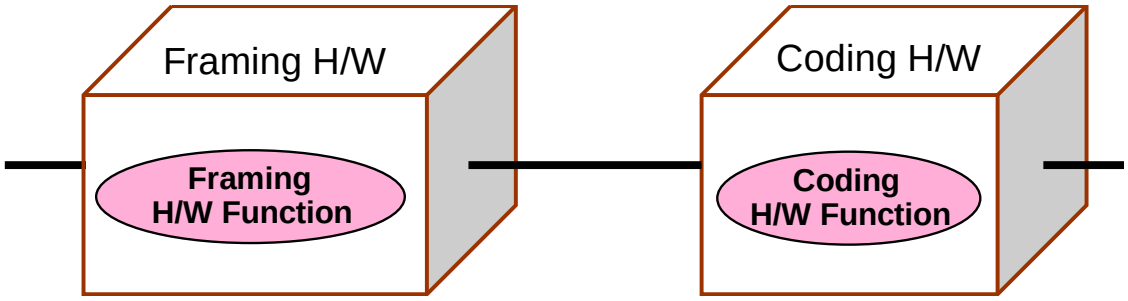


Figure 9-4: Example of Trade Study Mapping Functional to Connectivity View

Note that other Connectivity View implementation approaches are also possible, with one element implemented in software and the second implemented in hardware, as driven by performance requirements, mass, cost, or other design considerations.

ANNEX A

NOTES ON USE OF RASDS

A1 INTRODUCTION

A detailed treatise on how to actually ***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; Rechtin and Maier is very well regarded (Mark W. Maier and Eberhardt Rechtin, *The Art of Systems Architecting*, 2nd edition, CRC Press, 2000). The RASDS may be applied in a number of different venues as part of a system architecture description or design process. Not all of the Viewpoints of RASDS need to be used for all problems, and the first questions to ask are, 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 RASDS only two or three Viewpoints may be needed, and only one or two Views may be needed to address different concerns.

For each Viewpoint Specification RASDS defines the typical stakeholders and concerns, and defines the required set of objects and relationships that may appear in any view on a system. During development of system views 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 is specified in sections 4-8. Not all attributes are needed in all views for any given Viewpoint, and not all attributes are relevant for all objects in a view. Furthermore, RASDS does not provide any specific 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, and in still others they may be shown in a separate table. More formal methods of capturing these views, as discussed in the other annexes, offer excellent means for capturing these object attributes.

The following subsections provide a brief example of the sorts of heuristics that one can apply while using RASDS 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. For instance, 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 composition of those Nodes.

A2 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 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.

A3 EXAMPLE TECHNIQUES AND METHODS FOR FUNCTIONAL VIEW:

- 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.

A4 RELATED EXAMPLE TECHNIQUES AND METHODS FOR CONNECTIVITY VIEW

A methodology similar to that of the Functional View is followed for the Connectivity View. All of 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 ability of the performance of the engineered elements of the system to meet the 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.

ANNEX B

FORMAL METHODS AND TOOLS

B1 OVERVIEW

As noted in the introduction to this document, one of the primary motivations for the RASDS is to provide a kit of architect's tools that domain experts can use to describe and construct many different specific space data system architectures for complex systems. The RASDS can be used very effectively in its current form to provide a vocabulary for describing systems architectures, 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 or suite of tools to support design, the consistent use of these concepts and representational formalisms can help enormously in clarifying architectural descriptions and design.

The methods identified are derived from the RM-ODP and adapted as necessary to deal with the realities of systems that operate in space. This methodology has been validated through use in describing real space data systems, and these initial evaluations have been quite successful. A number of different missions and projects have used the RASDS to describe their high-level architectures, and some CCSDS working groups have successfully used the RASDS, in its draft form. Feedback from these activities has helped to refine the present document.

Although RASDS is useful even if it is used only to guide selection of useful views and their contents in a set of 'design drawings', in order for RASDS 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.

Developing a comprehensive tool suite is a major undertaking in its own right and would require significant resources. In order to minimize development costs and to minimize the costs of adoption it is desirable to leverage broadly supported methods and tools wherever possible. At present, there are two identified approaches that can utilize current active developments that support software and system engineering: Unified Modeling Language (UML) 2.0 and Systems Modeling Language (SysML). Requirements on these environments and tools, and specifications on the selected formal methods, have been described in a separate document (reference [19]).

Some initial experiments have been done using SysML and xADL (reference [13]) formalisms to represent RASDS. These have proven promising, and future work is likely to bear fruit. The SysML, which is based upon UML 2.0, is the mostly likely platform because it already provides formalisms for requirements, verification, viewpoints, describing hardware and software objects, and handling of discrete and continuous data flows.

Producing the SysML specification required extensions to (and also reductions in) the basic UML metamodel, which were defined using the Meta Object Facility (MOF, reference [8]).

The RM-ODP working group in ISO has itself developed some formal methods, but tools that implement the required functionality for these methods are not yet mature. Current efforts are underway in ISO/IEC JTC1/SC7/WG19 to use UML to provide a formal methodology to describe RM-ODP systems (reference [15]). In the long run this may prove to be a useful basis on which to build a RASDS formalism.

B2 SysML AND UML

Many current mission design efforts use document-driven approaches, employing common word processing and presentation software packages. While these are typical of the practice, they have significant limitations in that any models or views that are depicted are essentially just drawings with implied, but not explicit, semantic content. The state of the practice, at least for UML-based software architecture efforts, has been changing and a number of commercial and open source tools are available that support UML modeling. Information about SysML specifications and a current list of vendors may be found at the SysML Web sites, http://www.omgsysml.org/ and http://www.sysml.org. These vendors support UML profiles for full modeling as well as some simple extensions to drawing tools which permit graphical representations, but do not support full model development and semantic checking.

SysML, and the UML from which it is derived, provide a set of techniques and formalisms for modeling systems and software. Both of them bring a rich set of modeling constructs that permit description of systems from a number of 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 UML the model is in the tool; the drawings are just an external representation of the model.

b) These diagrams, in order to adequately represent stakeholder viewpoints, must be adapted with carefully defined user semantics for each view.

The SysML methodology extends UML 2.0 by adding requirements, verification, and parametrics to the UML suite of diagram types, as shown in figure B-1. It also eliminates several UML constructs that are of more value during software implementation, some of which would be of value during system design. 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 system engineering and architecting processes.

The SysML 1.0 specification has been finalized between the OMG and INCOSE, and commercial UML and simple drawing tools that support it are now emerging. As it stands, most of the diagram types that have been defined in SysML are useful, but they are far too generic, allow too many degrees of freedom, and provide too little guidance to the space data system architect.

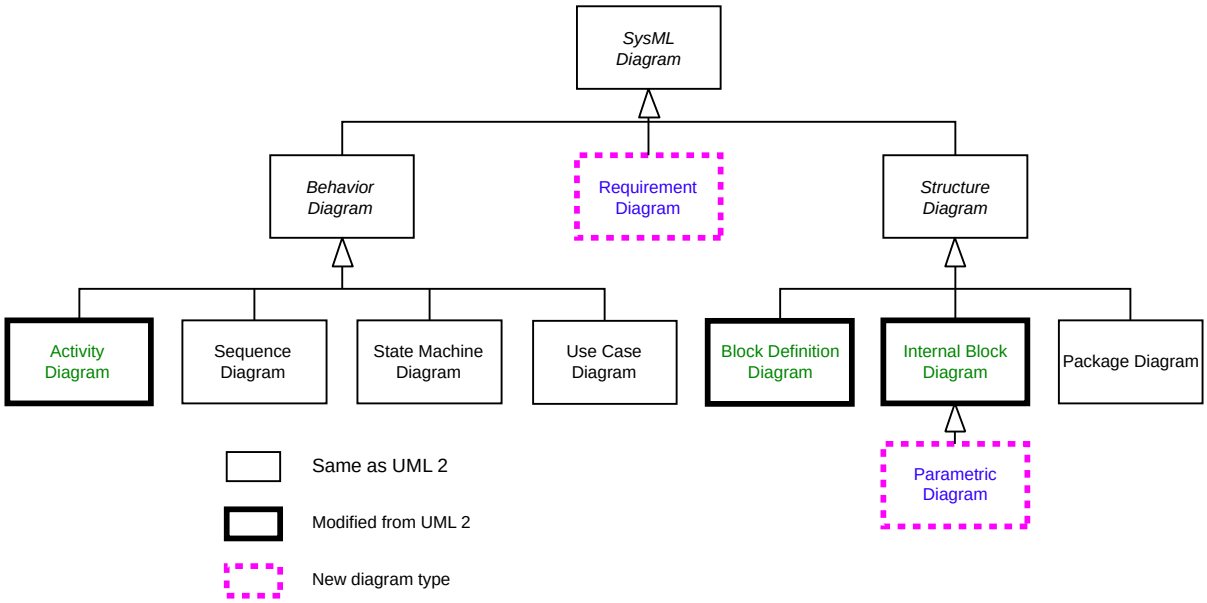


Figure B-1: SysML Diagram Types (SysML Partners)

SysML also has incorporated support for the sorts of viewpoint and view constructs that are required to define the kind of domain specific reference models that are described in this document. These definitions are, by design, largely compliant with IEEE 1471-2000. Members of the CCSDS System Architecture Working group (SAWG) worked with the SysML Submission Team to ensure that these concepts were adequately supported. What is needed next is a domain specific space system profile that will extend any tools supporting SysML to include explicit support for the sorts of viewpoint specifications and views that have been identified in RASDS. This work must be done in order to make these tools as useful as they can be.

As part of the SAWG studies an analysis was done of the capabilities defined within SysML and how it would map to the kinds of constructs needed by RASDS. The biggest realization was that many of these diagram types could be used for more than one viewpoint, but that what differentiated the viewpoint diagrams were the object types that were represented in any given viewpoint and the nature of the relationships that would be depicted. What would be needed to produce a useful adaptation of a UML or SysML tool environment would be to develop a profile that added RASDS–domain-specific viewpoint stereotypes to the generic SysML constructs. This is similar to the effort now underway within ISO to develop UML for RM-ODP.

An effort within the CCSDS SAWG to pursue development of such a formalism for RASDS has been halted due to a shift of priorities within the CCSDS Management Council (CMC), but there are some earlier studies of this topic. The following description, while not complete, is intended to provide guidance to any group that wishes to develop such a profile:

– 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 or UML diagram types and the kinds of constructs that are needed for RASDS Viewpoints. However, there is a sensible mapping for all of the RASDS constructs into at least one of the SysML types once the stereotyping for the particular RASDS object and relationship classes has been made. The fact that there is much more precision of expression possible with SysML 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 data system architectures.

As resources are made available the CCSDS System Architecture Working Group (SAWG) that developed this specification hopes to continue this work to finalize a set of formalisms and an adapted tool suite. Groups who are interested in this approach are encouraged to consider how it can be advanced and standardized so that commercial SysML or UML tools can be used in the development of these architectural models.

B3 RELATIONSHIP TO SOFTWARE ARCHITECTURE METHODS

This section briefly describes the relationship between common system and software architecture representations and RASDS. Most of the concepts defined in the RASDS viewpoints, objects, and relationships will be familiar to many practitioners in any system or software engineering and architecture community. At the same time, it is clear that for some

of these concepts there is already a diverse set of languages and modeling approaches in use. This is particularly the case for the software architecture elements associated with the functional and connectivity viewpoints.

For example, in the software architecture domain a frequently used analysis approach is the Krutchen 4+1 view model (reference [23]). These views are logical, process, physical, development, and use case.

- The logical view shows how the system is decomposed into a set of behavioral abstractions and it may use class, collaboration, or sequence diagrams. This is essentially identical with what is called the functional viewpoint in RASDS.

- The process view allows description of the systems processes, as implemented, and how they communicate. Activity diagrams are often used in this view. This view aligns with part of the connectivity viewpoint, in an allocated Engineering Object view.

- The development view describes the modules in the system and how they are organized and associated into packages and classes. This may also a part of the software view in the connectivity viewpoint, but no explicit method for displaying this is provided in RASDS.

- The physical view describes how the implemented software application is installed and operates in one or more computers. It may use a UML deployment diagram showing nodes that may contain one or more components. This is directly analogous to the connectivity viewpoint and to the allocation view.

- The use case view includes scenarios, and it is used to describe the required functionality of the system. It may employ use cases, activity diagrams, or descriptions of required actions of the system. This is aligned with the enterprise viewpoint, but RASDS provides no specific view representation.

While RASDS provides a quite clean mapping for the basic logical, process, and physical concepts described in the Krutchen 4+1 approach, it does not provide much guidance for the software development process itself. The field of software development is a rich one and it brings a wealth of new methods such as agile programming, design patterns, and other methods for risk reduction. What RASDS provides is a framework for describing the overall system architecture that accommodates these other software design processes and artifacts and relates them to the other elements in the system in a much more complete way than these other, more domain-focused methods, are able to do.

The Rational Unified Process (RUP-SE, reference [24]) and the related Model Driven System Design (MDSD, reference [23]) process also uses the Krutchen 4+1 model, but adds two additional viewpoints, one for workers and operational interactions, and one for geometric assemblages. To place this into a lifecycle context RUP-SE also introduces the concepts of model levels, which they describe as context, analysis, design, and implementation. These ‘levels’ largely relate to lifecycle phases. Note that RUP-SE has also adopted the IEEE-1471 language for describing viewpoints and acknowledges leveraging RM-ODP.

The RASDS does not explicitly model the operational interactions captured in RUP-SE nor many of the related operational views modeled in the DoDAF. Where these are required the available UML mappings can be used in a way that would integrate with the rest of any formal modeling framework.

See annex C for more discussion of the mappings between RASDS and DoDAF.

ANNEX C

RASDS AND DODAF COMPARISON

C1 INTRODUCTION

The RASDS has been specifically crafted to provide useful guidelines for the description of space data system architectures. As such it is a domain-specific methodology that provides means for describing the architectures of data systems that operate in space. Other methods, such as the DoD Architecture Framework (DoDAF, reference [11]) or the Systems Modeling Language (SysML, reference [9]), are formalized methods that are coming into wide use and could potentially be used for this same purpose. As was noted in annex B some work has already been done to use the SysML (and the underlying UML) formalisms to describe RASDS models.

A number of tools that directly support DoDAF modeling are available. In fact, several UML tool vendors now offer extensions to their development environments to support DoDAF views within the UML environment, leveraging UML methods where they are applicable. This is a very powerful approach in that these models of complex systems can really benefit from extensive tool support for managing the underlying information set and relationships.

This annex provides a brief introduction to the DoDAF views and products and provides a mapping of their constructs into RASDS. Information about ANSI/IEEE 1471 and Systems Engineering may be found in reference [22], which provides a useful comparison of IEEE 1471, RM-ODP, and some other frameworks.

C2 DoDAF VIEWS AND PRODUCTS

The DoDAF was designed by the U.S. Department of Defense to assist in the design, development and acquisition of hierarchically structured systems and systems of systems. As stated in the DoDAF Definitions and Guidelines (reference [11]):

> … architectures continue to provide a critical mechanism for understanding operational concepts and their relation to capabilities, anticipating changes in operational concepts or changes in automated capabilities, and acquiring both materiel and non-materiel assets. The DoD Components have made significant progress in using architectures. Examples of using architectures to support budgeting, identification of capability gaps, acquisition, and operations include the Air Force Task Force capability-based analysis, Navy's Mission Capability Package analysis approach, and OSD/Joint Staff concept of improving interoperability through focusing on key interfaces.

The framework supports the development of interoperating and interacting architectures. It defines three related views of architecture: Operational View (OV), Systems View (SV), and Technical Standards View (TV). The following are quoted from the DoDAF Definitions and Guidelines:

1.3.1 Definition of the Operational View

The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. DoD missions include both war-fighting missions and business processes. The OV contains graphical and textual products that comprise an identification of the operational nodes and elements, assigned tasks and activities, and information flows required between nodes. It defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges.

1.3.2 Definition of the Systems View

The SV is a set of graphical and textual products that describes systems and interconnections providing for, or supporting, DoD functions. DoD functions include both war-fighting and business functions. The SV associates systems resources to the OV. These systems resources support the operational activities and facilitate the exchange of information among operational nodes.

1.3.3 Definition of the Technical Standards View

The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed. The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture.

Note that DoDAF uses the term ***view*** differently from how it is defined in IEEE 1471 or in RASDS. In DoDAF parlance there are three views, each of which contains one or more architecture products. RASDS would call these three Viewpoints, with an associated set of Views. A strict use of definitions from IEEE-1471 would say that the DoDAF defines twenty-six different viewpoints. These twenty-six different products (All – 2, Operational – 9, System – 13, Technical – 2) may be tables, pictures, N2 diagrams, or others sorts of documentation artifacts. Similar to RASDS, not all of these products are needed or expected for any given architecture description.

C3 DoDAF AND RASDS MAPPING

In table C-1 all of the DoDAF product types are named and briefly described and shown in relationship to the corresponding RASDS viewpoints. As will be obvious, for a number of the DoDAF products there is no corresponding RASDS View. Because of its intended use DoDAF is much stronger than RASDS in describing operational interactions, a full third of the products relate almost exclusively to enterprise and organizational concerns. At the same time, RASDS is much stronger in describing the technical architectures of system elements and has the definitions and views to deal with physical space systems, their interactions with the environment, and the protocols used in their construction, all largely lacking in DoDAF.

Table C-1: DoDAF Views and Products and RASDS Viewpoints

DODAF Views and Product	RASDS Viewpoints
Overview and Summary Information (AV-1) - Scope, purpose, intended users, environment depicted, analytical findings	None specified in RASDS
Integrated Dictionary (AV-2) - Architecture data repository with definitions of all terms used in all products	None specified, but an ontology could be used for this
High-Level Operational Concept Graphic (OV-1) - High-level graphical/textual description of operational concept	None specified in RASDS
Operational Node Connectivity Description (OV-2) - Operational nodes, connectivity, and information exchange need lines between nodes	Enterprise Viewpoint includes all of this information
Operational Information Exchange Matrix (OV-3) - Information exchanged between nodes and the relevant attributes of that exchange	Information Viewpoint includes information attributes; Enterprise Viewpoint includes exchanges; no matrix view is specified
Organizational Relationships Chart (OV-4) - Organizational, role, or other relationships among organizations	Enterprise Viewpoint includes this information; chart representation may be useful
Operational Activity Model (OV-5) - Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information	Enterprise Viewpoint includes this information. Specific activity-modeling diagrams may prove useful in some circumstances
Operational Rules Model, State Transition, and Event Trace (OV-6a, b, c) - Three products used to describe operational activity— business rules that constrain operation, business process responses to events, traces of actions in a scenario or sequence of events	None specified, but these may be of use in some space architecture-modeling efforts that focus on operational issues
Logical Data Model (OV-7) - Documentation of the system data requirements and structural business process rules of the Operational View	Information Viewpoint includes data models, Enterprise Viewpoint includes requirements, but business rules are not specified

DODAF Views and Product	RASDS Viewpoints
Systems Interface Description (SV-1) - Identification of systems nodes, systems, and system items and their interconnections, within and between nodes	Connectivity Viewpoint includes all of these elements and relationships
Systems Communications Description (SV-2) - Systems nodes, systems, and system items, and their related communications lay-downs	Communications Viewpoint includes all of this and more, defines protocol stacks and interactions absent in SV-2
Systems-Systems Matrix (SV-3) - Relationships among systems in a given architecture; can be designed to show relationships of interest, e.g., system-type interfaces, planned vs. existing interfaces, etc.	None specified in RASDS; may be useful in certain circumstances
Systems Functionality Description (SV-4) - Functions performed by systems and the system data flows among system functions	Functional Viewpoint includes all of this
Operational Activity to Systems Functionality Traceability Matrix (SV-5) - Mapping of systems back to capabilities or of system functions back to operational activities	No matrix specified explicitly in RASDS, but relationships are identified, (see figure 2-8)
Systems Data Exchange Matrix (SV-6) - Provides details of system data elements being exchanged between systems and the attributes of that exchange	Information Viewpoint includes all of this information; see also correspondences to Functional and Connectivity viewpoints
Systems Performance Parameters Matrix (SV-7) - Performance characteristics of Systems View elements for the appropriate time frame(s)	Connectivity Viewpoint includes performance characteristics of system elements and interactions with environment
Systems Evolution Description (SV-8) - Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation	None specified in RASDS
Systems Technology Forecasts (SV-9) - Emerging technologies and software/hardware products that are expected to be available in a given set of time frames and that will affect future development of the architecture	None specified in RASDS

DODAF Views and Product	RASDS Viewpoints
Systems Rules, State Transition, and Event Trace Descriptions (SV-10a, b, c) - Three products used to describe system functionality— identify constraints that are imposed on systems functionality, responses of a system to events, system-specific refinements of critical sequences of events described in the Operational View	None specified explicitly in RASDS; may be of use in Functional or Connectivity Viewpoint analyses
Physical Schema (SV-11) - Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema	Information Viewpoint includes all of these elements
Technical Standards Profile (TV-1) - Listing of standards that apply to Systems View elements in a given architecture	Communications and Information Viewpoints partially cover these, specifically for protocol and information elements, but not other standards
Technical Standards Forecast (TV-2) - Description of emerging standards and potential impact on current Systems View elements, within a set of time frames	None specified in RASDS

The DoDAF also has a concept of an integrated architecture description that is intended to explicitly relate elements in one view to elements defined in another view:

> An architecture description is defined to be an *integrated architecture* when products and their constituent architecture data elements are developed such that architecture data elements defined in one view are the same (i.e., same names, definitions, and values) as architecture data elements referenced in another view. The term *integrated architecture* refers to an architecture description that has integrated Operational, Systems, and Technical Standards Views. That is, there are common points of reference linking the OV and the SV and also linking the SV and the TV. For example, SV-5 relates operational activities from OV-5 to system functions from SV-4; the SV-4 system functions are related to systems in SV-1, thus bridging the Operational and Systems Views.

Similar to RASDS, there is a defined set of relationships in DoDAF between the elements defined in one view and the elements defined in another. Compare the partial example of DoDAF elements, shown in figure C-1, with figure 2-8, which shows the RASDS top-level objects and relationships.

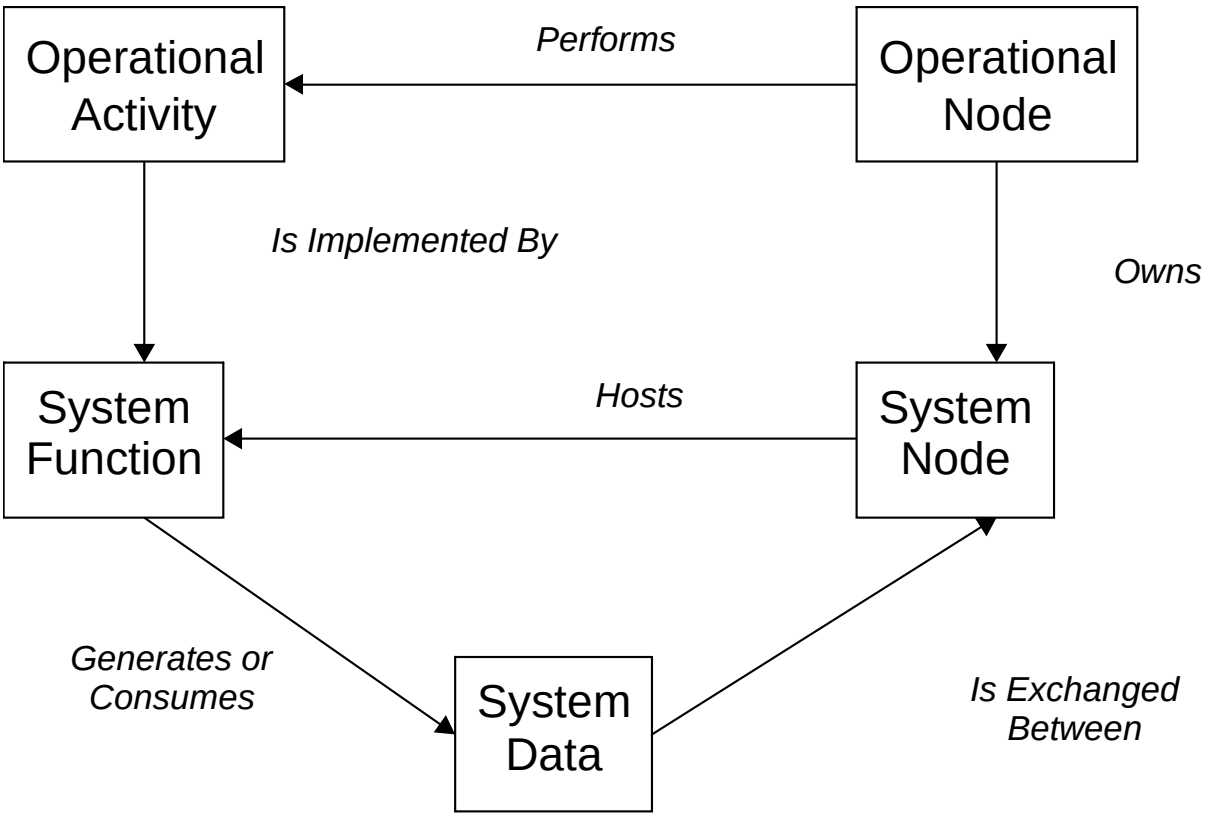


Figure C-1: DoDAF Elements and Relationships (Partial)

From this comparison it should be clear that for the elements and views that are most important for space data system architecture descriptions that RASDS provides all of the essential capabilities that DoDAF does, and more. This is particularly the case for technical architecture description, interactions with the environment, and description of protocol structures, use, and design. At the same time DoDAF is much stronger in its support for high-level overviews (AV-1, OV-1) and for the provision of detailed descriptions of organizational interactions and matrices of organizational and functional interactions that may be useful during design analysis. The DoDAF products also provide useful views for presenting evolving systems and standards, which RASDS has omitted. For applications where such design views are useful and predominate, architects should be encouraged to adopt the DoDAF methods, since they are supported by a number of existing tools.

C4 DoDAF AND RASDS ANALYSIS

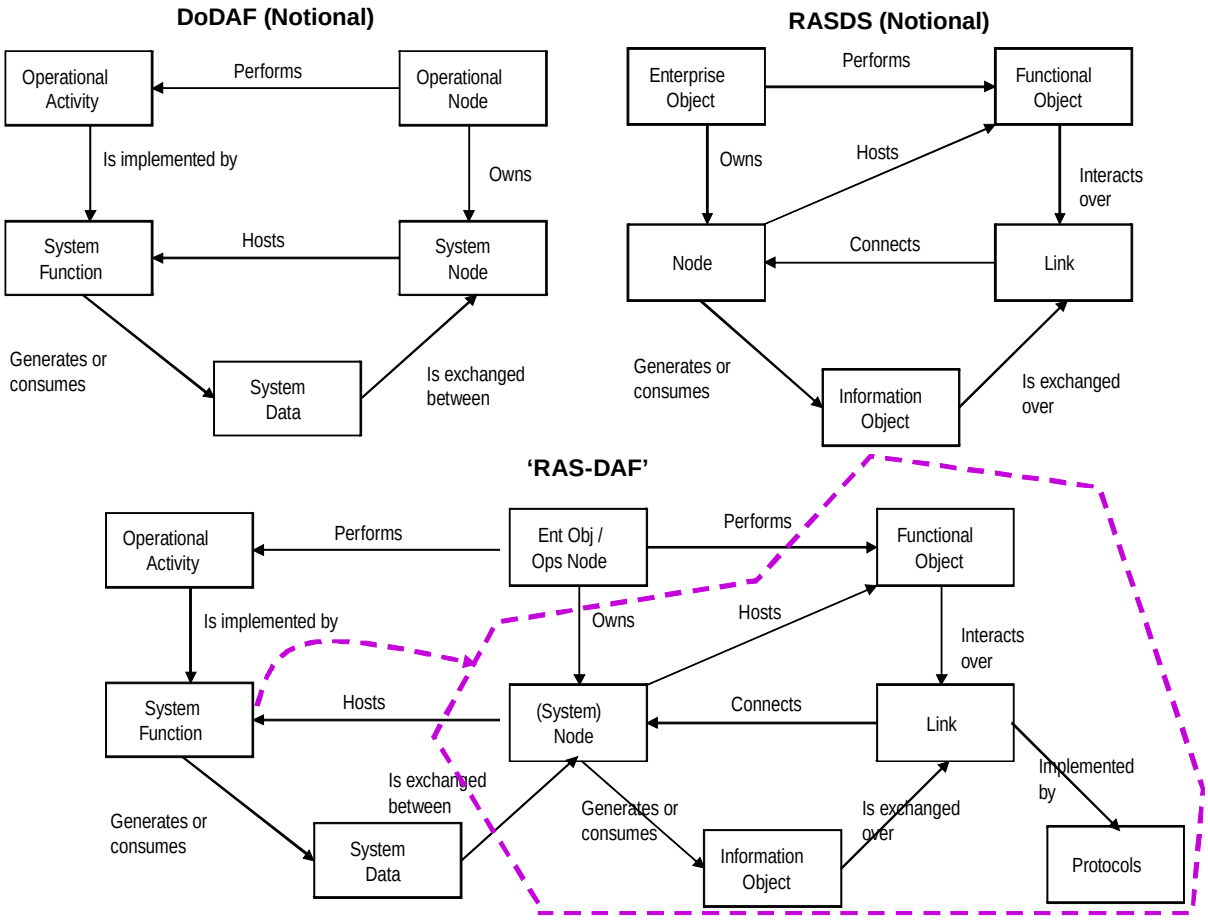
Some space data system design projects have attempted to use DoDAF to describe space data system architectures and discovered that it has limitations for use in this domain. The primary issue is that while DoDAF provides useful views for describing organizational interactions and information exchanges, its ability to provide adequate descriptions of technical system architectures, environmental interactions, and protocols is very limited.

Although DoDAF was not used for developing RASDS, a close mapping can be defined between RASDS elements and DoDAF elements and between RASDS Viewpoints and DoDAF views/products, as presented in table C-1. Similar to, but distinct from, the discussion of RASDS and SysML, with DoDAF there are also multiple mappings of constructs. One example is that the kind of information described by the RASDS information viewpoint appears in both the operational information exchange matrix (OV-3) and the system data exchange matrix (SV-6) of DoDAF. This is not unlike the correspondences between information objects in the Functional and Information Views in RASDS.

In DoDAF, relationships between many of the elements in different views/products are clearly defined (e.g., operational activities are implemented by system functions at system nodes) and the users must specify the relationships between instances of different elements. These relationships are very similar to those defined in RASDS.

RASDS does not have a standard way of describing operational and system behaviors (OV-6 and SV-10), system performance parameters (SV-7), systems incremental evolution (SV-8), or systems and technical standards forecasts (SV-9 and TV-2). These are all made explicit in DoDAF, and if these views are important to a designer in a particular application, the DoDAF products could be directly adapted. Similarly, if a carefully detailed view of operational activities and organizational information exchanges is important in any given architecture analysis, the DoDAF OV-3 and OV-5 could be adopted.

Most of the information called out in DoDAF products can equally well be represented in RASDS views. Where DoDAF has particular strength is in its ability to represent organizational interactions and to support planning for large-scale system acquisitions, which ability was one of its primary motivations. Where RASDS has strengths is in its ability to represent technical architectures and protocols, particularly of space data systems and the complexities involved in operating in space. With attention to use of terminology and representations, system architecture descriptions can be developed that combine both methods and draw on the strengths of both.



RASDS/DoDAF => 'RAS-DAF'

Figure C-2: DoDAF and RASDS

ANNEX D

GLOSSARY AND ACRONYMS

D1 ACRONYMS

ANSI	American National Standards Institute
AOS	Advanced Orbiting Systems
API	Application Program Interface
AV	All View (of DoDAF)
BWEM	BandWidth Efficient Modulation
C&DH	Command and Data Handling
CCSDS	Consultative Committee for Space Data Systems
CEOS	Committee on Earth Observation Satellites
CFDP	CCSDS File Delivery Protocol
CMC	CCSDS Management Council
CPU	Central Processing Unit
DBMS	Data Base Management System
DEDSL	Data Entity Dictionary Specification Language
DoD	(U.S.) Department of Defense
DoDAF	DoD Architecture Framework
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
H/W	Hardware
IEC	International Electrotechnical Committee
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Standards Organization

ISO-BRM	ISO/IEC Basic Reference Model
JTC	Joint Technical Committee (of ISO)
MDSD	Model Driven System Design
MOF	Meta Object Facility
N2	'N Squared' diagram
ODP	Open Distributed Processing
OMG	Object Management Group
OS	Operating System
OSLP	Overview of Space Link Protocols
OV	Operational View (of DoDAF)
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
RF	Radio Frequency
RFC	Request for Comment (Internet standards)
RM-ODP	Reference Model-Open Distributed Processing
ROM	Read Only Memory
RTLT	Round Trip Light Time
RUP-SE	Rational Unified Process – System Engineering
SAP	Service Access Point
SAWG	System Architecture Working Group
SC	Subcommittee (of ISO)
SCPS	Space Communications Protocol Specification

SLE	Space Link Extension
SOA	Service Oriented Architecture
SoS	System of Systems
SV	System View (of DoDAF)
S/W	Software
SysML	System Modeling Language
TC	Telecommand
TOGAF	The Open Group Architecture Framework
TV	Technical View (of DoDAF)
UML	Unified Modeling Language
UPMS	UML Profile and Metamodel for Services
WG	Working Group
xADL	Extensible Architecture Description Language

D2 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]

Abstract Data Architecture Meta-Models—Models for specification and standardization of data elements (e.g., ISO/IEC 11179, DEDSL). [8.3.4]

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]

Activity—A specification of behavior described as a sequence of actions. [3.2.3]

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. [6.3.4]

Application—one or more pieces of software designed to perform some specific function; a configuration of interacting implemented software Engineering Objects. [6.3.4]

Application Programming Interface—A set of definitions of the ways one piece of computer software communicates with another; a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software. [7.3.4]

Architecting—The process of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture. Architecting 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; 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]

Artifact—Any tangible object made, modified, or used by people, or produced during system design, development, testing, or operations. [8.3.4]

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]

Communications Viewpoint—An engineering and technology view on a space data system that focuses on the protocols and mechanisms of information transfer performed by that system. [7.2]

Community—An entity (e.g., Earth Science) that may exist within one Space Enterprise or across multiple Space Enterprises. A community 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. [4.3]

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 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. [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]

Connectivity Viewpoint—An engineering viewpoint on a space data 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. [6.2]

Constraint—A limitation or implied requirement that limits the design solution or implementation, is not changeable by the enterprise, and is generally nonallocable. [3.2.3]

Correspondence—A function such that for elements in one viewpoint there is a related element in another viewpoint; the condition of being in conformity from elements in one viewpoint to elements in another. [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. [4.2]

Data Architecture—Models of the structure and relationships among the data elements used within a system. [8.3.4]

Data Models—Schema and structure definitions. [8.3.4]

Data Objects—Information objects, either physical or digital. [8.3.4]

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. [4.3]

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. Engineering Objects may be implemented as hardware (Node) or as software (application or software component). [6.3.4]

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. [4.2]

Enterprise Viewpoint—A view of a space data 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. [4.2]

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 modeling it is reserved to refer to things in the universe of discourse being modeled. [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]

Facility—A physical infrastructure element that supports the use of services and other resources. [4.2]

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. [4.3]

Firmware—Software that is contained in a read-only memory (ROM) device. Firmware is typically treated as software unless there is a reason for showing the hardware component itself. [6.3.4]

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 data system, or to support actions of another Functional Object, and to transfer, generate, or process data in performing those actions. [5.2]

Functional Viewpoint—A view on a space data 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. [5.2]

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. [6.3.4]

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. [5.4.4]

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. [8.2]

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. [8.2]

Information Viewpoint—A view of a space data 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. [8.2]

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. [8.3.4]

Interaction—An action performed by an object with participation of another object or with its environment. [3.2.3]

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. [3.2.5]

Location—A point or extent in space. [3.2.4]

Logical Object—An abstract entity that may be considered separately from any particular implementation or deployment. A Logical Object has no physical manifestation except as part of a model, but it may have associated behaviors and interfaces. [3.2.4]

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]

Link—The locus of relations among Nodes. A Link provides interconnections between Nodes for communication and coordination. A Link may be implemented by a wired connection or with some RF or optical communications media. A Link 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. [6.3.4]

Metadata—data about data; the information that describes content. Metadata 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. [8.2]

Meta-model—An explicit model of the constructs and rules needed to build specific models within a domain of interest. [8.2]

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 in the model for other concepts that are related to these layers, for example, **(N)-protocol**, **(N+1)-service**. [7.3.4]

Node—A model of a space data 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. [6.3.4]

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]

Operations Concept—A verbal or graphic statement, in broad outline, of assumptions or intent in regard to 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. [4.4.4]

Organization—A formal group of people with one or more shared goals. [4.3]

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. [4.2]

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]

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. [6.3.4]

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. [7.3.4]

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. [7.3.4]

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. [7.3.4]

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. [8.3.4]

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. [8.3.4]

Relationship—The way in which two or more entities can be associated with one another. [3.2.3]

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]

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, 4.2]

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. [4.4.4]

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. [8.3.4]

Semantics—Rules by which syntactic expressions are assigned meaning. [3.2.3]

Service—A provision of an interface of an object to support actions of another object. [3.2.3]

Service Access Point—The point at which (N-layer)-services are provided by an (N-layer)-protocol-entity to an (N+l-layer)-protocol-entity. [7.3.4]

Software or computer programs—the components of information systems that provide operating instructions for specific task-based applications that run on computing hardware. [6.3.4]

Space Enterprise—A top-level autonomous entity (e.g., NASA) that is dedicated to the exploration and/or exploitation of space. A Space Enterprise has its own objectives, resources, and policies, and it is not a component of any other Space Enterprise. [4.3]

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. [7.3.4]

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. A ***State Table*** is an alternative tabular representation of the same information. [7.3.4]

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]

Type—Specification of 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. [3.2.3]

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 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 data 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]