The Consultative Committee for Space Data Systems

Draft Report Concerning
Space Data System Standards

INFORMATION ARCHITECTURE REFERENCE MODEL

DRAFT INFORMATIONAL REPORT

CCSDS 312.0-G-0

DRAFT GREEN BOOK

February 2006

AUTHORITY

Issue:	Draft Informational Report, Issue 0
Date:	February 2006
Location:	Not Applicable

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and reflects the consensus of technical panel experts from CCSDS Member Agencies. The procedure for review and authorization of CCSDS Reports is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*.

This document is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard 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.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (Roskosmos)/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 Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish Space Research Institute (DSRI)/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 & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- 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
1.0	Draft Information Architecture for Space Data Systems	02/01/2004	Draft
1.1	Draft Information Architecture for Space Data Systems	04/01/2004	Reviewed in Montreal, Canada
1.2	Draft Information Architecture for Space Data Systems	08/01/2004	Submitted to wider IAWG audience for review
1.3	Draft Information Architecture for Space Data Systems	02/05/2005	Revised Chapter 2 after IAWG meeting in Toulouse, France
1.4	Draft Information Architecture for Space Data Systems	03/03/2005	Organizational changes to the structure and content of chapter 2 after IAWG discussion.
1.5	Information Architecture for Space Data Systems Green Book Submission	06/01/2005	Document revised, in particular changes to section 2 to address comments by IPR Working Group after Spring meetings in Athens, Greece.
1.6	Reference Architecture for Space Information Management Green Book Revision after TIM at Goddard	09/01/2005	Draft CCSDS 312-0.G-1 revisions made after IAWG meeting in Atlanta, GA
1.7	Reference Architecture for Space Information Management Green Book Submission	11/10/2005	CCSDS 312-0.G-1
CCSDS 312.0-G-0	Information Architecture Reference Model, Draft Informational Report, Issue 0	February 2006	Current draft

CONTENTS

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

In the absence of sufficient information system standards for interoperability and cross support, we have seen systems developed that do not allow the exchange of information between the ground side of the domain and the flight side of the domain. These systems, often, do not allow for integrated exchange of information between components within these environments, let alone space agencies. The focus of this document is to present a reference space information management architecture (or information architecture in short) that encompasses the capture, management, and exchange of data for both flight and ground systems across the operational mission lifecycle. This includes identification of a set of conceptual functional components for information management, definition of their interfaces for information management, representation of these components and interfaces, and definitions of information processes (interactions between users and systems). The intent of this document is to provide a conceptual basis on which standards can be developed to support information management across the entire mission environment. This document, therefore, defines the necessary concepts and terminology for information architecture and leverages much of the past CCSDS work in the area. Part of this leveraging includes defining how existing standards can be assembled to fit into an information architecture for deploying space data systems. The information architecture covers problem areas associated with space data systems (such as organizational, functional, operational, and cross support issues).

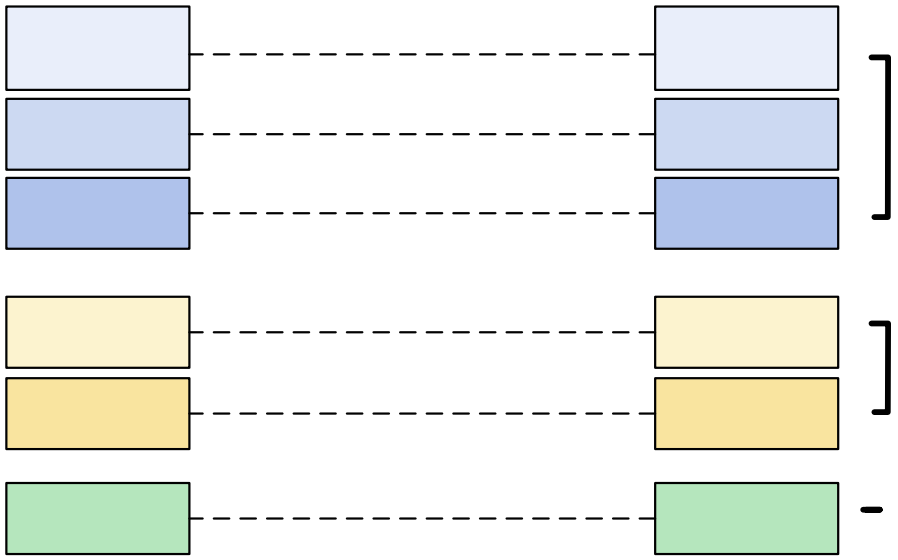


Figure 1-1: High-Level Abstract View of Interoperable Information Architecture

The information architecture presented within this document is layered. To achieve interoperability both within and across domains, and across applications built based on this document, each layer should be addressed. Figure 1-1 depicts this view and represents possible means of achieving interoperability at each layer. Each layer is critical to achieving interoperability. At the software and information levels, it is essential that common interfaces and meta-models for information and messages flowing between application interfaces be

defined along with common definitions for the information itself, in order to achieve system interoperability. This architecture document purposely separates into sections the information architecture and the information management components that implement that architecture. The separation of the information architecture from the software architecture promotes reuse provided that the software components can be configured by common meta-models. The concept of having multi-mission, but common meta-models is critical to achieving multi-mission, cross agency interoperability. For example, a common XML schema defined to annotate telemetry data files could be developed to support improved information management of telemetry systems. Structuring the XML document in such a way as to enable a common cataloging function to catalog the metadata in the XML document across missions would provide a multi-mission capability. If this XML schema is derived from a core meta-model, then it could support annotation of other data objects such as science data objects. This would enable the *same cataloging function*, deployed in a *completely different* part of the ground system, to catalog the science data. Architecting systems to consider the underlying models, how they are derived, and how they can be used by a core set of information components, will increase the longevity of software systems design and promote an infrastructure which enables improved utility of the data generated from international space missions.

1.1 SCOPE AND APPLICABILITY

This document is intended for those interested in understanding and developing information architectural elements for building space data systems. These elements include software components, such as registries, and repositories, and data components and interfaces. This document is most applicable in complex environments such as space, but clearly has the potential to provide a roadmap for information architecture in many types of data systems.

1.2 TERMINOLOGY

The following terminology is used throughout the document.

Model	A model provides a specification for representing objects and their relationships.
Metadata	Metadata is literally ‘data about data’, i.e., information that describes another set of data.
Meta-model	A meta-model is a model which describes another model.
Schema	A schema is a means for defining the structure, content and, to some extent, the semantics of data.
Application Information Object	An application information object (AIO) is an object containing an internal Data Object and a Metadata Object.

Application Information Architecture	An application information architecture is the notion of architecting information systems across system domains (e.g., space data systems, archiving systems, biomedical informatics systems) with a focus on both data architecture, and software architectural concerns.
Data Architecture	A data architecture is the specification the overall structure, logical components, and the logical interrelationships of data and information.
Software Architecture	A software architecture is the specification of overall structure, behavior, logical components, and logical interrelationships of a software system.
Data Product	A data product is the result of an active function which produces data. A data product may be simple and include just data value, or it may be complex and contain both data and metadata objects.

1.3 REFERENCES

[1] "Association, Aggregation and Composition." June 1998. *Object Orientation Tips*. <http://ootips.org/uml-hasa.html>

[2] Mandar Chitnis, Pravin Tiwari, and Lakshmi Ananthamurthy. "The UML Class Diagram: Part 1." May 2003. *Developer.com*. <http://www.developer.com/design/article.php/2206791>

[3] "The SIMBAD Astronomical Database." *Centre de Données Astronomiques de Strasbourg*. <http://cdsweb.u-strasbg.fr/Simbad.html>

[4] J. Blythe, E. Deelman, and Y. Gil. "Automatically Composed Workflows for Grid Environments." *IEEE Intelligent Systems* 19, no. 4 (July/August 2004): 16-23

[5] *Reference Model for an Open Archival Information System (OAIS)*. Recommendation for Space Data System Standards, CCSDS 650.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, January 2002.

[6] *The Data Description Language EAST Specification (CCSD0010)*. Recommendation for Space Data System Standards, CCSDS 644.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 2000.

[7] *Data Entity Dictionary Specification Language (DEDSL)—XML/DTD Syntax (CCSD0013)*. Recommendation for Space Data System Standards, CCSDS 647.3-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, January 2002.

[8] Ann Chervenak, et al.. "A Framework for Constructing Scalable Replica Location Services." In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (Baltimore, Maryland)*, 1-17. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002.

[9] Ann Chervenak, et al.. "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets." *Journal of Network and Computer Applications* 23 (2001): 187-200.

[10] D. Crichton, et al.. "A Component Framework Supporting Peer Services for Space Data Management." In *Proceedings of the 2002 IEEE Aerospace Conference (Big Sky, Montana)*, 2639-2649. Piscataway, NJ: IEEE, 2002.

[11] D. J. Crichton, J. S. Hughes, and S. Kelly. "A Science Data System Architecture for Information Retrieval." In *Clustering and Information Retrieval*, edited by W. Wu, H. Xiong, and S. Shekhar, 261-298. Network Theory and Applications. Norwell, Massachusetts, USA, and Dordrecht, The Netherlands: Kluwer, 2003.

[12] Ewa Deelman, et al.. "Mapping Abstract Complex Workflows onto Grid Environments." *Journal of Grid Computing* 1, no. 1 (2003): 9-23.

[13] Ewa Deelman, et al.. "Grid-Based Galaxy Morphology Analysis for the National Virtual Observatory." In *Proceedings of the 2003 IEEE Conference on Supercomputing (Phoenix, AZ)*. Los Alamitos, CA, USA: IEEE Computer Society, 2003.

[14] "EOSDIS Core System Data Model." January 6, 2006. *NASA Earth Science Data Systems SPG*. NASA/Goddard Space Flight Center. <http://spg.gsfc.nasa.gov/standards/heritage/eosdis-core-system-data-model>

[15] Roberto Puccinelli. "An Introduction to DataGrid." Illustrated by Aldo Stentella. March 2004. *The DataGrid Project*. <http://web.datagrid.cnr.it/LearnMore/index.jsp>

[16] *The Globus Alliance*. The University of Chicago/Argonne National Laboratory. <http://www.globus.org/>

[17] H. Gomaa, D. Menasc, and L. Kerschberg. "A Software Architectural Design Method for Large-Scale Distributed Information Systems." *Distributed Systems Engineering* 3, no. 3 (1996):162-172.

[18] *Hyperdictionary*. <http://hyperdictionary.com/>

[19] *Information technology—Metadata registries (MDR)—Part 1: Framework*. International Standard, ISO/IEC 11179-1:2004. 2nd ed. Geneva: ISO, 2004.

[20] C. Kesselman, I. Foster, and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." *The International Journal of High Performance Computing Applications* 15, no. 3 (Fall 2001): 200-222.

[21] *SpaceGRID Study Final Report*. SGD-SYS-DAT-TN-100-1.2. Issue 1.2. SpaceGRID Consortium, 2003.

[22] Chris A. Mattmann, et al.. "Software Architecture for Large-Scale, Distributed, Data-Intensive Systems." In *Proceedings of the 4th IEEE/IFIP Working Conference on Software Architecture (WICSA-4, Oslo, Norway)*, 255-264. Los Alamitos, CA, USA: IEEE Computer Society, 2004.

[23] R.W. Moore, et al.. "Data-Intensive Computing." In *The Grid: Blueprint for a New Computing Infrastructure*, edited by Ian Foster and Carl Kesselman, 105-130. San Francisco: Morgan Kaufmann Publishers, 1999.

[24] *Object Management Group*. <http://www.omg.org/>

[25] Lou Reich. "XML Packaging for the Archiving and exchange of Binary Data and Metadata." In *Proceedings of the 2003 Open Forum on Metadata Registries (Santa Fe, New Mexico)*. 2003. <http://metadata-standards.org/>

[26] Gurmeet Singh, et al.. "A Metadata Catalog Service for Data Intensive Applications." In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing (Phoenix, AZ)*. Los Alamitos, CA, USA: IEEE Computer Society, 2003.

[27] SPASE Consortium. *A Space and Solar Physics Data Model*. Version 1.0.1. N.p.: SPASE Consortium, January 2006. <http://www.igpp.ucla.edu/spase/data/makedoc.php>

[28] Tim Bray, et al., eds. *Extensible Markup Language (XML) 1.0*. 3rd ed. W3C Recommendation. N.p.: W3C, February 2004. <http://www.w3.org/TR/2004/REC-xml-20040204/>

[29] *XML Formatted Data Units*.[1]

[30] *Parameter Value Language Specification (CCSD0006 and CCSD0008)*. Recommendation for Space Data System Standards, CCSDS 641.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, June 2000.

[1] XFDU Structure and Construction Rules is a proposed CCSDS Recommended Standard.

2 INFORMATION ARCHITECTURE

2.1 INTRODUCTION

Software systems of today are growing in complexity, dynamicity, and heterogeneity, and are becoming increasingly more costly to operate. Space data systems are a representative example of this emerging trend. Data systems in the space systems application domain are highly distributed, complex software entities that must manage information from its inception at a scientific instrument to its distribution via one of many existing CCSDS protocols (e.g., CFDP, Proximity-1), to its arrival at a ground station on Earth, to its delivery to a science processing center, and ultimately to its archival in a long term archiving center for preservation. In a sense, the space data system should be driven by the *model*s of the information that it must process, distribute, and manage. This could mean models of an image on a spacecraft. It could mean models of engineering data that needs to be sent to a control operations center. It could also mean *models* of other *models*. There are many different models that need to be managed across an end-to-end space data system. To avoid rigidity, however, software used by a space data system should be flexible: it should be driven by the models that it operates on, and not *vice versa*.

For the most part, however, current space data systems are not flexible, and include software implementations that *are* extremely tied to the information that they operate in. Sadly, space data systems are not unique in this regard. Space data systems serve as a prime example of many existing information system application domains. Bio-medical informatics systems, science processing systems, and space flight operation systems all exhibit the same austere structure: software and model tied together. A change in the model requires a change in the software; a change in the software leads to a change in the model.

In this document, the *application information object*[1] is described. The application information object is the cornerstone of defining and constructing a data-driven system where models and software function in unison, but are separate entities. An application information object is an independent, flexible model of the data and corresponding metadata in an information system, and is meant to be reusable across many information system domains. The main guiding principle of the information object is to separate the models of information (e.g., data, metadata, etc.) from the actual implemented system code. In this fashion the software system and the models that describe the information in the system may both evolve independently of one another. Modularity, separation of concerns, and dynamic evolution of information system components are only a representative cross-section of the benefits that this model provides.

The information object is composed of the *data object*, a sequence of bits responsible for physically representing data, and the *metadata object*, information about the data object including, but not limited to, structure, semantic, and preservation information (reference [6]). This section starts by providing key definitions and is followed by a small taxonomy of

[1] Also used and described throughout the document as an *information object*.

information object types commonly used in information architecture and a set of Standard Information Object examples across domains for clarification. The section concludes with definitions of *meta-models, domain models, and data dictionaries,* which play a key role in the description of information objects.

2.1.1 DATA OBJECTS

Data objects are either physical objects or digital objects as illustrated in figure 2-1. A physical object is a tangible thing (e.g., a moon rock) together with some representation information bringing to light the fact that any object that can be described with data is a data object. On the other hand, a digital object is a sequence of bits, representing a thing that is not tangible (e.g., an electronic document, image file, a 'folder' of files). This document focuses on the digital object specialization of the data object; the physical object specialization is not considered.

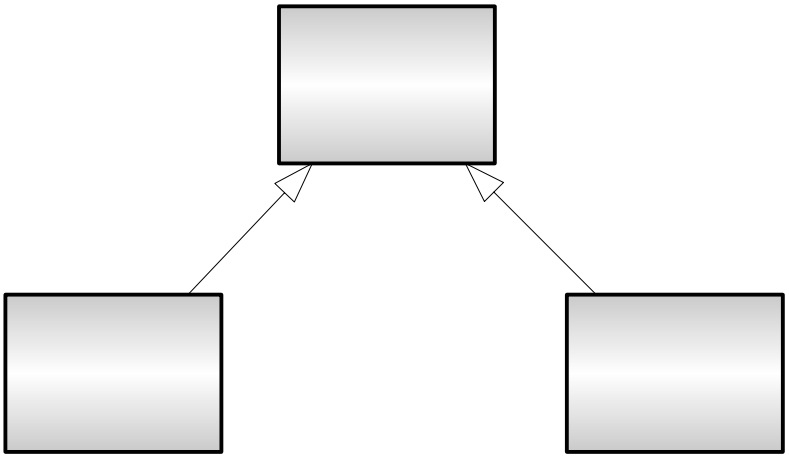


Figure 2-1: A Data Object

2.1.2 METADATA OBJECTS

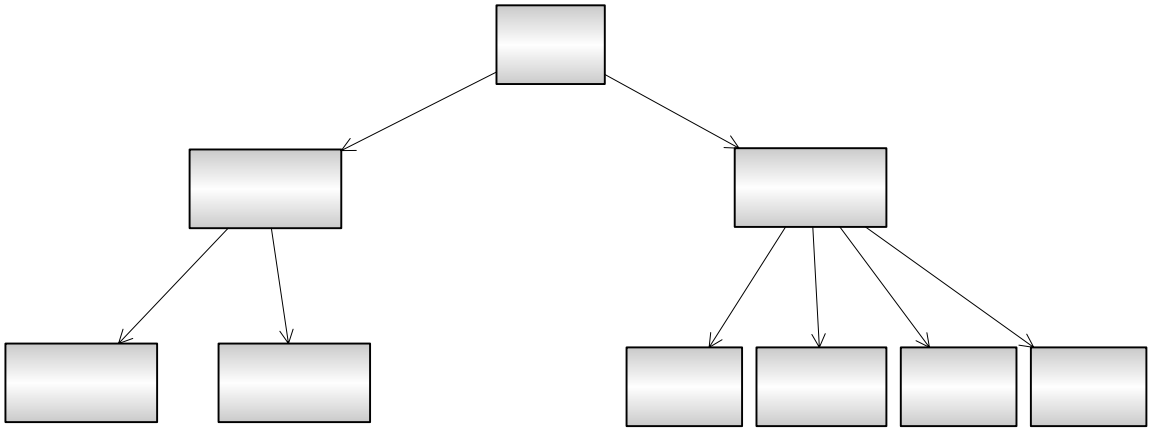


Figure 2-2: A Metadata Object, Adapted from Reference [5]

Metadata objects in this document provide information (or *metadata*) about the data object. Similar to the OAIS reference model (reference [5]), a metadata object in this document comprises representation and preservation description information as two broad classifications of metadata. As shown in figure 2-2 representation information includes structure (syntactic) and semantic information and preservation information includes reference, provenance, fixity, and context information. Also, the metadata objects described in this document might be atomic or comprised of a set of metadata sub-objects. Data objects and metadata objects are highly interdependent. Without the metadata object, essentially the data object is just a self-contained sequence of bits about which nothing is known: systems cannot unlock its information. When a metadata object *and* data object are present (e.g., an information object), a myriad of capabilities are available to the user (or system). If the data object is an image, most likely the metadata object will describe what *kind* of image (JPEG or 'raster' for example). If the metadata object mandates that the data object has a field called *pixel*, an examination of a specified (by the metadata object) location within the data object will reveal the value of the pixel.

Represen
Informa

Structure
Information

Se
Info

2.2 INFORMATION OBJECTS

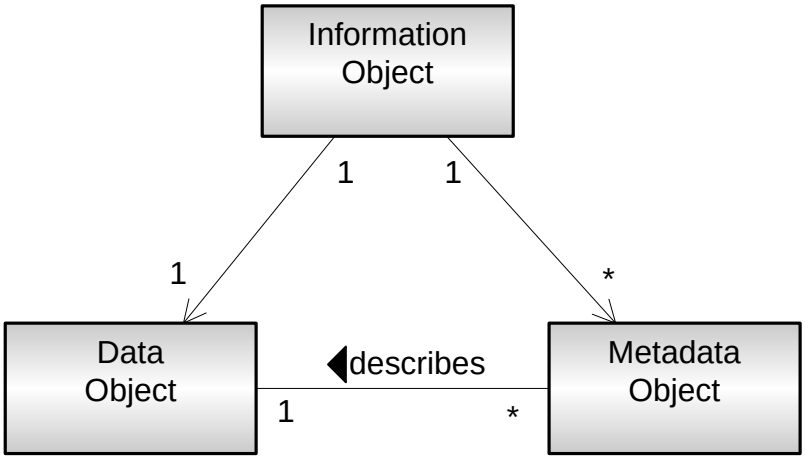


Figure 2-3: An Information Object

Information objects (shown in figure 2-3) build upon the data and metadata object by logically associating them together. Information objects are components in information architecture that model both a granule of information (i.e., the *bits*) and its corresponding *metadata*. An information object consists of a data object and one or more metadata objects: the latter models the aforementioned information and metadata properties. The metadata object can describe the data object's *structure*, such as what fields it is composed of, the fields' *valid values* (e.g., in the case of 'Uplink Speed', the data may have a controlled list of available speeds such as 1MB or 2MB/sec), and the *semantic relationships* between the structural elements (such as 'Uplink Speed **must always equal** Downlink Speed').

2.2.1 TAXONOMY OF INFORMATION OBJECTS

For the purposes of comparing different information objects, this subsection identifies a set of information object *classes*. They are detailed below.

2.2.1.1 Primitive Information Object

A *primitive information object* is an information object with simple metadata information that contains a small amount of metadata with a data object. Simple metadata indicates that the only metadata captured for a particular data object are primitive attributes such as name format, and modification date. These are attributes typically associated with a file in a file system and seldom provide any information about content or relationships.

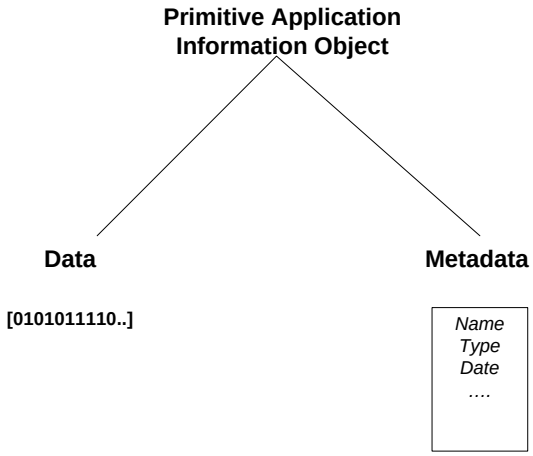


Figure 2-4: Primitive Information Object Example

An example of a primitive information object is a data file managed in a solid state recorder. Minimal metadata exists for it other than basic properties that define its name, type, and size. A name often is used to denote specialized information about an object. In practice, it is preferable to separate the name of an object from other information such as creation date, sequence numbers, etc. Many space data systems have typically focused on the management of primitive information objects and have not made metadata objects first-class citizens.

2.2.1.2 Standard Information Object

A *Standard Information Object* is defined as an information object that has well-defined metadata and a data object. The metadata is an instance of one or more domain models. The data object can be null. A number of data systems throughout the space agencies have Standard Information Objects as part of their system design. These have been predominately used within archive and science processing data systems. The metadata for these information objects are often defined by some data description language like XML and may be stored in an online registry or database to enable effective search and browsing. Increasing emphasis on constructing end-to-end mission information system architectures will require that Standard Information Objects be used at a variety of stages including observation planning, execution, processing, and distribution across the mission pipeline. Standard Information Objects are applicable across this entire pipeline since it is a mechanism to enable interoperability between systems as long as the information objects and their associated models are planned.

2.2.1.3 Complex Information Object

Complex information objects (shown in figure 2-5) are information objects that encapsulate one or more information objects, coupled with a metadata object containing *packaging information*. Similar to the OAIS reference model (reference [5]), packaging information is the set of information, consisting primarily of package descriptions, which is provided to data management

to support the finding, ordering, and retrieving of information holdings by consumers. Additionally packaging information is the information that is used to bind and identify the components of an information package. For example, it may be the ISO 9660 volume and directory information used on a CD-ROM to provide the content of several files containing content information and preservation description information. It also can describe the algorithms and formats of the package structure itself (e.g., whether or not the package was compressed, which compression algorithm was used, such as ZIP, TAR,[1] etc.).

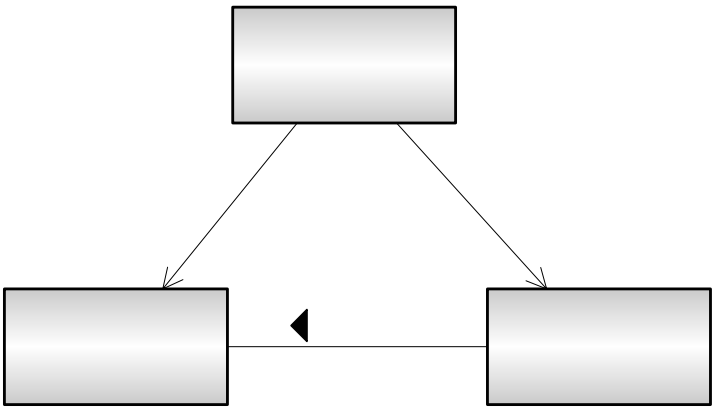


Figure 2-5: A Complex Information Object

Each information object in a complex information object includes its own metadata that may or may not correlate with other metadata from the other information objects in the package. This makes it difficult to interpret and compare information objects, even ones that come from the same repository, unless they conform to a standard meta-model, e.g., such as the XFDU packaging model (reference [25]).

The purpose of the complex information object is to provide the aggregation of related data to the user. It is assumed that the user typically knows how to use each information object within the set. If the user does not know how to correlate the information, then descriptive information related to the complex information object (such as *index information* regarding the individual information objects in the complex information object) can be used to deduce package properties.

2.2.2 EXAMPLES OF INFORMATION OBJECTS

This subsection explores information objects through several illustrative examples in the context of different application domains. For ground data systems, a spacecraft command

[1] See reference [18] for definitions of ZIP and TAR.

message file information object is discussed. For archive data systems, a planetary data system information object is discussed. Finally, for space data systems, a service link exchange (SLE) information object is discussed.

2.2.2.1 Spacecraft Command Message File

Table 2-1: Information Object View of a Spacecraft Command Message File

Data Object		**Metadata Object**		
Name	***Type***	***Data Element***	***Data Element Type***	***Semantic Constraints***
Command	Sequence of bits	Ground Station Name	String	None
		Packet-Sent Time	Timestamp	≤ Current System Time
		Instrument Name	String	Value:= $a \mid a \in \{\text{spectrometer, hi-resolution imager}\}$

A spacecraft command message file is a telemetry uplink packet sent from a ground station to a spacecraft. It can be modeled using an information object. The information object is made up of a sequence of bits representing the command to be sent to the spacecraft. This bit sequence is mapped to an application information object consisting of one data object, *command sequence*. The associated structural information for the telemetry uplink packet consists of three data elements, *ground station name* (representing the ground station that sent the command to the spacecraft), *instrument name* (representing the instrument on-board the spacecraft that this sequence of commands is intended for), and *packet sent-time* (a timestamp representing the exact time the packet was sent from ground to space). Semantic information about these three data elements consists of *valid values* for the data element *instrument name* (e.g., spectrometer, or hi-resolution imager), and *min value* for the timestamp, which states that the timestamp for *packet sent-time* should be less than or equal to the current time on the sending system. This example is summarized in table 2-1.

Table 2-2: Information Object View of a Planetary Data System Product

Data Object		**Metadata Object**		
Name	***Type***	***Data Element***	***Data Element Type***	***Semantic Constraints***
SPICE files	Set of ancillary spacecraft data files	File Name	String	Must exist in the volume
		Orbit Numbers	Long Integer	Must be valid orbit within the mission
		Mission Name	String	Must be valid PDS Mission
Image Files	Raster Image	Image Dimensions	W x H image dimensions	Dimensions must not exceed 1024 pixels by 768 pixels

2.2.2.2 Planetary Data System Product

A Planetary Data System (PDS) product is an *archive* structure consisting of one or more science data files (e.g., image files, calibration files, SPICE files) and a PDS Label file in the ODL format. It can be represented using the information object construct. The information object consists of a set of data objects, such as *image or SPICE files*. Each data object is described by a metadata object, the PDS Label. For the SPICE files, metadata objects defined data elements such as *file name* to identify the name of the SPICE file, *Orbit Numbers* to identify the spacecraft orbit numbers that the SPICE file data covers, and *Mission Name* for which the SPICE file describes the navigation data. Each of the data elements for the SPICE files has semantic constraints. For instance, the mission name element's value must be a valid PDS mission. The orbit number element's value must be a valid orbit number from the mission. The file name of the SPICE file must exist in the PDS volume. For each the image file data objects, there are single metadata objects containing the data element *image dimensions*, which describes the width and height of the image in pixels. There is a single semantic constraint on this element; for example, in this case, the width of the image must not exceed 1024 pixels, and the height must not exceed 768 pixels. This example is summarized in table 2-2.

Table 2-3: Information Object View of an SLE Service Management Object

Data Object		**Metadata Object**		
Name	***Type***	***Data Element***	***Data Element Type***	***Semantic Constraints***
Trajectory Prediction Operations Constraints	TBD	Maximum Size Storage	Long integer	Maximum CM allowed size of storage does not exceed a pre specified value.
		Allowed trajectory formats	List	Formats conform to standard mimeType specifications.
		Operation Timeout Limits	List	Timeouts cannot exceed pre specified value.
Service Agreement	Aggregation of SLE objects	Service agreement identifier	URN	Identifier URN should be within an accepted SLE namespace

2.2.2.3 Space Link Extension (SLE) Service Management Objects

CCSDS is developing standards to support automation of requests between agencies for managing space link and SLE services known as 'SLE-SM'. SLE-SM defines a set of information objects called *service management objects* (shown in figure 2-6) for automating the exchange of SLE-SM information. The service request includes the *service agreement*, *configuration profiles*, *trajectory predictions*, and *service packages.* The SLE Service Management Objects can be modeled as an information object in the same fashion shown in previous examples. The SLE Service Management information object would consist of a set of data objects including a service agreement, trajectory prediction constraints, a forward carrier agreement, and the rest of the objects shown in figure 2-6. Each data object would have a corresponding metadata object. For example, two of the data objects from figure 2-6 are shown in table 2-3 above, leaving out the rest of the examples for brevity. In the above table, the Trajectory Prediction Constraints data object has a metadata object associated with it that contains three data elements: *Maximum Size Storage*, of type long integer, that represents the maximum amount allowed for storage by a content manager (CM); *Allowed Trajectory Formats* is a list of acceptable CCSDS and non-CCSDS formats that this service agreement defines; and *Operation Timeout Limits* is a list of timeout values on operations involved in this service agreement. Examples of semantic constraints in the above metadata objects would be verifying that the operation timeout limits do not exceed pre-specified values, that the formats correspond to known mimeType specifications, and checking to ensure that the maximum size storage does not exceed a pre-specified value.

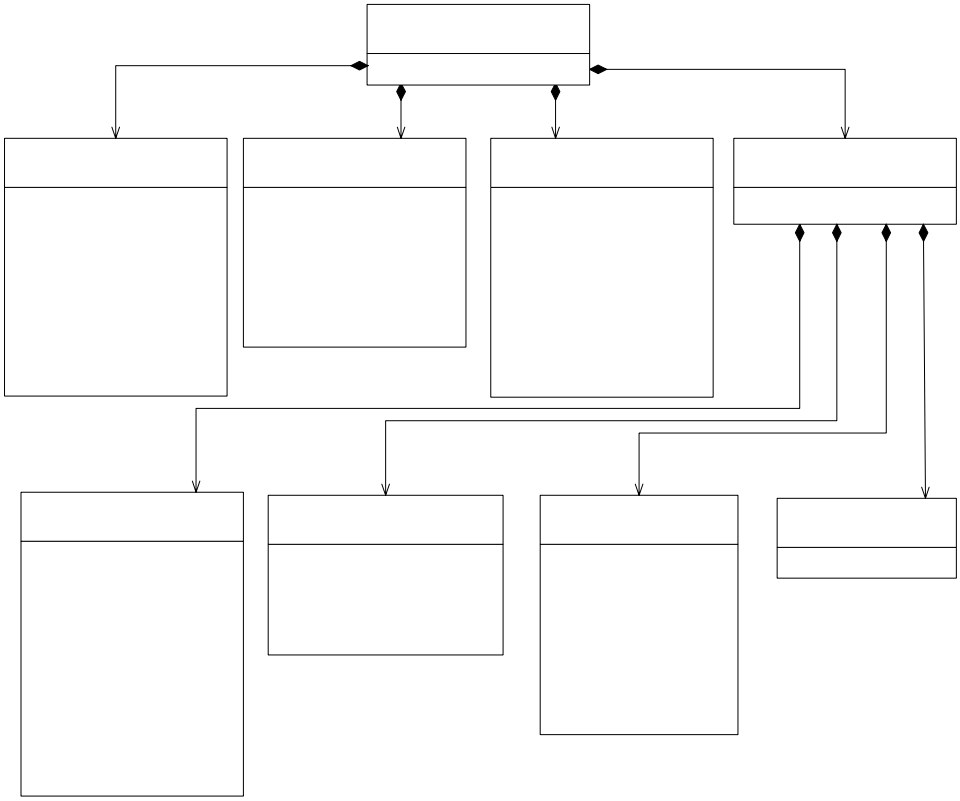


Figure 2-6: Service Agreement Information Model Overview

2.3 MODELING CONCEPTS

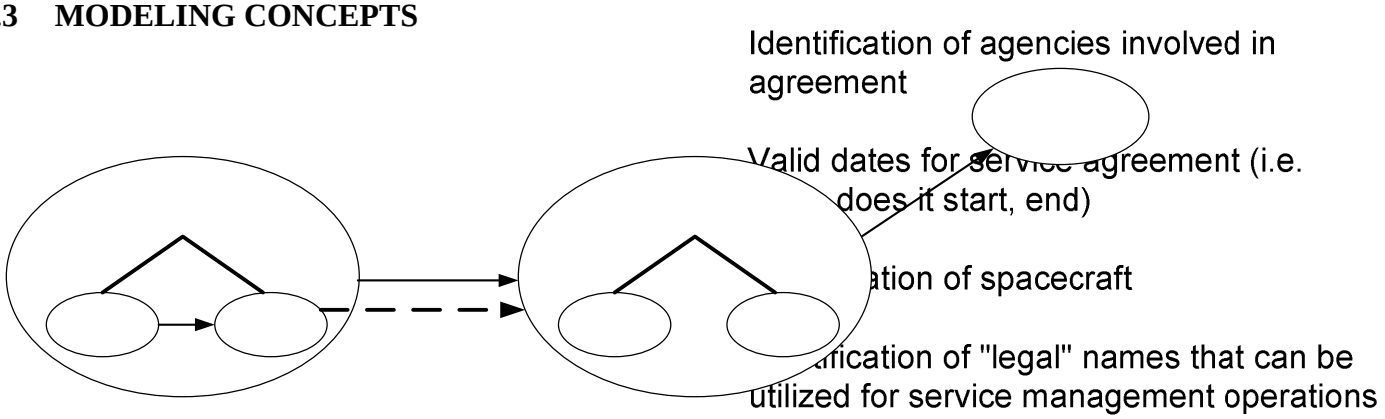


Figure 2-7: Information Object in Context

Models are important in information architecture because they provide means to describe and use objects. Without explicit models, objects cannot be examined, understood, or changed accurately. They also cannot be compared or integrated with other objects. These capabilities are critical in space data systems because they facilitate the correlative use and exchange of data.

Service Package Operations

General Agreement

The basic relationship between information architecture models and the objects they describe are illustrated in figure 2-7. The data object is described by the metadata object, both components of an information object. The metadata object is an instance of a class of objects that are prescribed by (one or more) *domain model*s (e.g., a 'preservation domain model'). The domain model in turn is an instance of a class of objects that are prescribed by a meta-model.

The hierarchical relationship between objects, models, meta-models, and even meta-meta-models can be simplified using a generalization proposed by OMG (reference [24]). Namely, when considering the hierarchy of models illustrated in figure 2-8, any object at level *n* can be described by an instantiation of an object from a class in level *n-1*. For example, the domain model at level M1 can be described by an instance of a UML model at level M2.

Models interact within and across levels. For example, ISO/IEC 11179 can be used as a model for a data dictionary. In turn, the data dictionary could be used as a component of a domain model.

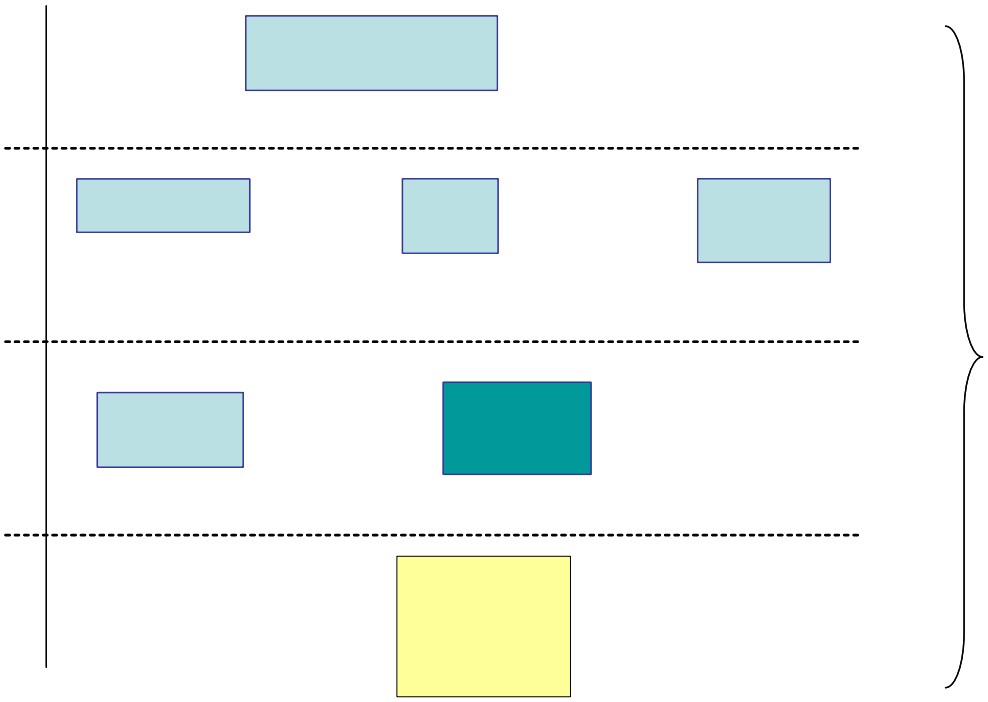


Figure 2-8: Model Hierarchy, Adapted from Reference [24]

In information architecture, the focus is on identifying a set of standard models that meet the requirements for developing information management systems. At the M3 level, MOF has been identified as the key model. At the M2 level, UML, the XML meta-model, and

ISO/IEC 11179 have been identified as key models for system and domain models, data interchange structures, and metadata registries, respectively.

2.3.1 META-MODELS

A meta-model is simply a model that prescribes another model. For example in the generalization illustrated in figure 2-8, UML at level M2 is a meta-model that can be used to develop a domain model at M1.

In information architecture, meta-models are important because they prescribe how elements can be compared and examined across domains. If elements did not conform to a particular meta-model, then it would be impossible to guarantee the ability to compare and examine them even within the same domain. Since the ability to compare elements is critical to enabling interoperability of data exchanged between systems, it is necessary that common and/or compatible meta-models be used to describe domain elements both within and across domains. For example, the PDS defines the data element Mission Name described earlier using a meta-model that requires the data element name, a description and a set of valid values. In order for a query using Mission Name as a constraint to find data in both the PDS and another space science domain it is critical that the second domain have a compatible meta-model in order to find the equivalent mission constraint that will produce valid results.

In the following subsections, several standard meta-models will be briefly described. These include the ISO/IEC 11179 standard for the specification and standardization of data elements (reference [19]), the CCSDS Data Entity Dictionary Specification Language (DEDSL) (reference [7]), and the XFDU (reference [25]) model for describing information packages.

2.3.1.1 ISO/IEC 11179

In the realm of meta-models, the ISO/IEC 11179 (reference [19]) standard framework for the specification and standardization of data elements provides a basic foundation for meta-models, metadata registries and how to use them. It specifies general registry functions such as definition, identification, naming, administration, and classification. Practically it provides an accepted base set of attributes needed to describe data elements. As an international standard it also provides a global basis for data element definition and classification and supports data dictionary interoperability. The specification classifies the basic set of attributes into four categories: *identifying*, *definitional*, *representational*, and *administrative*.

2.3.1.2 Data Entity Dictionary Specification Language (DEDSL)

The Consultative Committee for Space Data Systems (CCSDS) Data Entity Dictionary Specification Language (DEDSL) provides a specification for the construction and interchange of data entity dictionaries using XML, and its conformance to ISO/IEC 11179 has been documented in (reference [7]).

2.3.1.3 XFDU

The XML Formatted Data Unit (XFDU) Structure and Construction Rules is a set of CCSDS recommendations for the packaging of data and metadata, including software, into a single package to facilitate information transfer and archiving. It also provides a detailed specification of core packaging structures and mechanisms that meets current CCSDS agency requirements. (See reference [29].)

2.3.2 SPACE DOMAIN MODELS

A *domain model* describes objects belonging to a particular area of interest. The domain model also defines attributes of those objects, such as name and identifier. The domain model defines relationships between objects such as 'instruments *produce* data sets'. Besides describing a domain, domain models also help to facilitate correlative use and exchange of data between domains. Below we briefly mention some common space domain models.

2.3.2.1 Planetary Science

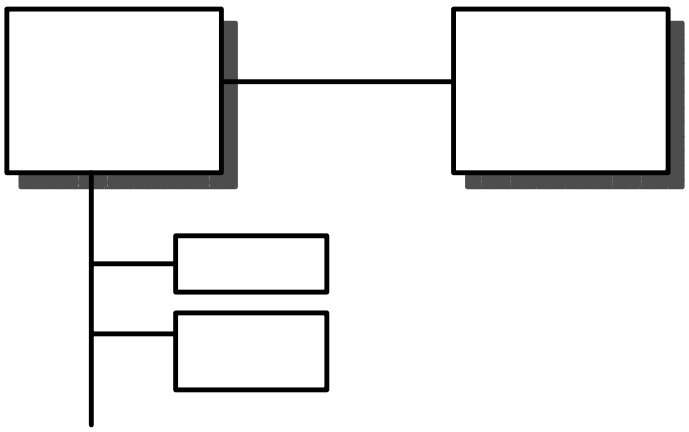


Figure 2-9: Example Planetary Domain Model (Simplified)

NASA's Planetary Science domain model defines objects such as *instruments* and *data sets* and *science users* and their associated relationships (such as instruments produce data sets and data sets are distributed to science users). This is illustrated in figure 2-9. The planetary science domain model was defined in PVL/ODL (described in Section 2.3.3.1).

2.3.2.2 SPASE

SPASE (reference [27]) is a space and solar physics domain model being developed by an international working group with participation from several national agencies, universities, and industrial affiliates. The SPASE model attempts to define relationships between ancillary data, images, and plots for space and solar physics data products, such as images and data collected about photons and particles.

2.3.2.3 EOSDIS and EOS Core Data Model

The EOSDIS (reference [17]) domain model defines data product types, a knowledge base, and a global thesaurus for Earth science terminology to interpret the data products collected in Earth observing systems. Data products include sea surface temperature measurements, global climate measurements, and many other Earth science data products. The ECS Core Data Model (reference [14]) was developed as an extension to the earlier EOSDIS domain model in order to specify relationships necessary to handle the sheer data volume (nearly two terabytes a day) that is regularly captured in the EOSDIS system. In the ECS Model data is represented as collections of smaller units, called granules. Collections define a series of attributes including, but not limited to, spatial coverage, temporal coverage, and contents.

2.3.3 DATA DESCRIPTION LANGUAGES

Data description languages are notations used for representing semantic and syntactic data. As such, they provide the necessary implementation level facilities to manipulate and exchange application information objects, and to implement meta-models, domain models and information. Some common examples of data description languages are listed below.

2.3.3.1 PVL/ODL

The Parameter Value Language (PVL) (reference [30]) is a CCSDS Recommended Standard for the specification of a standard keyword value type language for naming and expressing information objects. It defines a language that is both human readable and machine readable. This keyword value type language has been used to document domain models in a way conceptually similar to the approach taken by the W3C (World Wide Web Consortium)'s Resource Description Framework (RDF). The Object Description Language (ODL) is a subset of PVL.

2.3.3.2 EAST

The Data Description Language EAST Specification (reference [6]) is a CCSDS Recommended Standard that defines a language and syntax for the expression and exchange of information objects, in the form of *Data Description Records* (DDRs). The idea behind a DDR is to provide enough information about data (e.g., its format, size, etc.) to be able to interpret and exchange it in an automated fashion.

2.3.3.3 XML

The Extensible Markup Language (XML) (reference [28]) is a World Wide Web Consortium (W3C) specification and syntactic format for data objects formatted in XML, which is a subset, or restricted form of the popular Standard Generalized Markup Language (SGML). XML defines information objects called *entities* which capture data (parsed or unparsed) delimited by XML *tags*, which are named value attributes enclosed by a '<' and '>' symbol

respectively. Entities may have sub-entities, and *attributes*, which describe related information about a particular entity object, such as its name, or its id.

2.4 INTEROPERABILITY

Data dictionary interoperability is a key facet of enabling heterogeneous data systems to exchange and compare information. Ultimately, since domain models contain data elements that model a particular domain, and because data elements for a domain model originate from the data dictionary for a particular domain, the data dictionary plays an important role in making data systems exchange information.

It is important to have a common meta-model for data dictionaries so that they can be captured and exchanged in a common way. This is critical for building things like metadata registries and for capturing and sharing data elements across projects. Further, it is important to recognize that data dictionaries cannot be constructed without a domain model.

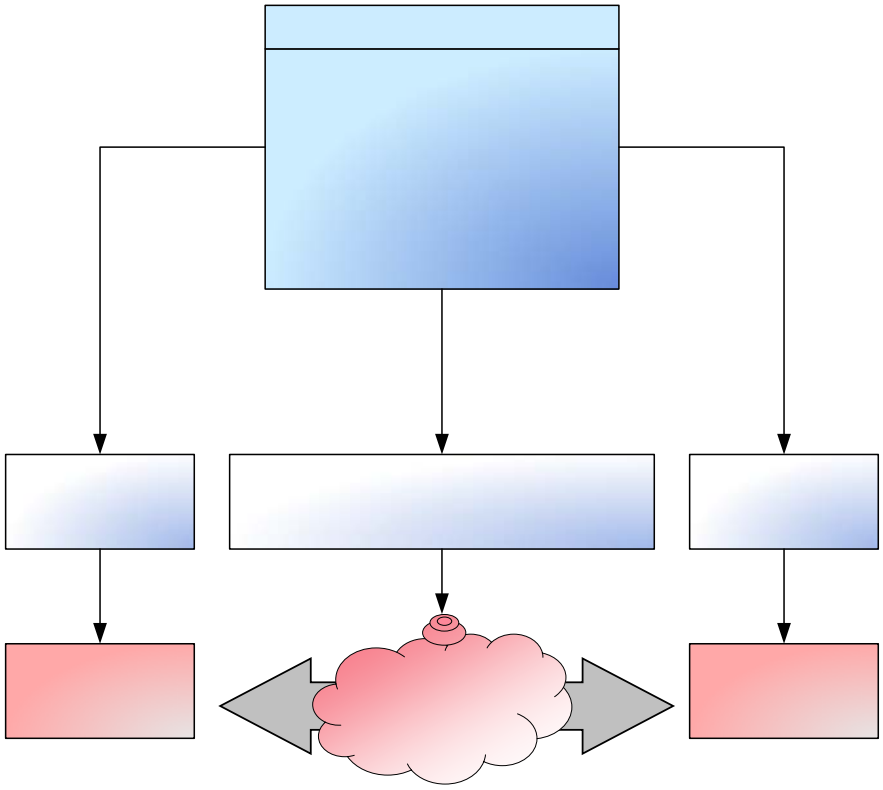


Figure 2-10: Data Models, Meta-Models, and Domains

The key requirement to enable data system interoperability is to have common or at least compatible data elements across the respective domain data models. In figure 2-10, two domains and their respective data models are illustrated. The two domains can interoperate, or exchange information, when knowledge exists about data element commonality at the data model level. For example, if both domain data models contain the data element *target name* with 'Mars' as a valid value, then the two domains can exchange information about Mars.

The knowledge about data element commonality, depicted as the *interchange model* in the figure, is difficult to acquire and requires domain experts to compare elements from their respective domains for similarities. Often elements will have similar attributes such as *name* and *valid value*s but significantly different interpretations and definitions. These similarities and differences must be understood and documented.

The process of comparing data elements is made much easier if a single data model (or meta-model) is used to capture the domain data models. It provides a standard notation, syntax, and semantics so that data elements from two different domains can be contrasted and compared. For example, the ISO/IEC 11179 recommendation for the specification of data elements (reference [19]) provides a comprehensive set of attributes for describing data elements and provides a good basis for data dictionary development.

3 SOFTWARE COMPONENTS FOR INFORMATION ARCHITECTURE

This section describes Information Management Objects (IMOs)[1] used for the access, distribution, capture, and management of information objects. Two types of IMOs are identified: *Primitive Information Management Objects (pIMOs)* and *Advanced Information Management Object (aIMOs)*. Generally, aIMOs are constructed from one or more pIMO components. pIMOs are active objects capable of *put*ting, *get*ting and *find*ing information from the underlying data stores. aIMOs are complex objects, composed from one or more pIMOs, that enable basic capabilities of information architecture, including *ingestion*, *retrieval*, *processing*, *distribution,* and *querying* of data objects, metadata objects, and information objects. Although this set of capabilities is not the entire set of capabilities that could be derived, it is meant to be a framework of building blocks from which further complex capabilities are defined.

3.1 PRIMITIVE INFORMATION MANAGEMENT OBJECTS

Primitive information management objects are simple functional components capable of manipulating their underlying data storage using *put*, *get*, and *find* operations. There are two types of pIMO: *Data Store Object (DSO)*, and *Query Object (QO).* These objects (components) are called *primitive* since this document does not explicitly identify any of their architecturally relevant sub-components. Both of these pIMOs operate on a *Physical Data Storage* component.

A physical data storage component is a hardware or software component responsible for storing data. Devices such as tape drives, hard disks, solid state recorders, RAM, flash memory, and the like are all examples of physical data storages. There are two basic parts of physical data storages:

- **Memory**—the physical location of the data in the data storage (labeled as ‘D’ in figure 3-1);
- **Local Identifiers**—the index catalog of pointers to memory containing data objects (labeled as ‘H’ in figure 3-1).

These parts enable low-level access to physical data storages to (1) place data objects into memory locations; and (2) index those locations for use in search and retrieval process. The organization of a physical data storage is shown in figure 3-1.

[1] The words ‘objects’ and ‘components’ are used interchangeably in this context.

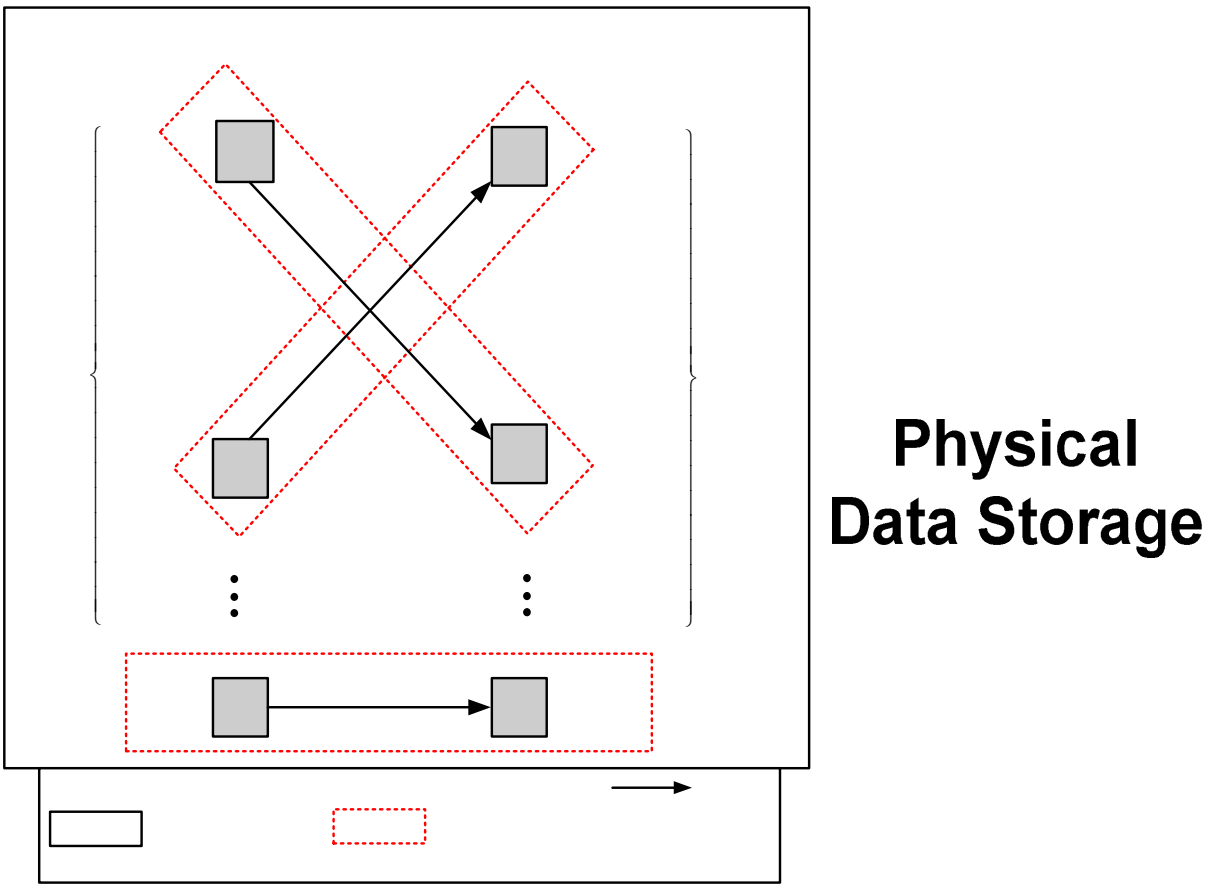


Figure 3-1: The Internal Structure of a Physical Data Storage

3.1.1 DATA STORE OBJECT

The *data store object* (shown in figure 3-2) is attached to a physical data storage and supports *putting* and *getting* information. Figures 3-3 and 3-4 depict the *put* and *get* operations of the DSO, respectively. The *get* operation takes a *local identifier* as input (ranging from a simple memory address to a string identifier) and returns the data object residing in the addressed memory location as an output. The *put* operation takes a data object as input and, upon completion, places the data object in a free memory location (labeled as 'local identifier' in figure 3-3) determined by the catalog and ingestion process of the underlying physical data storage. The local identifier is then returned back to the caller.

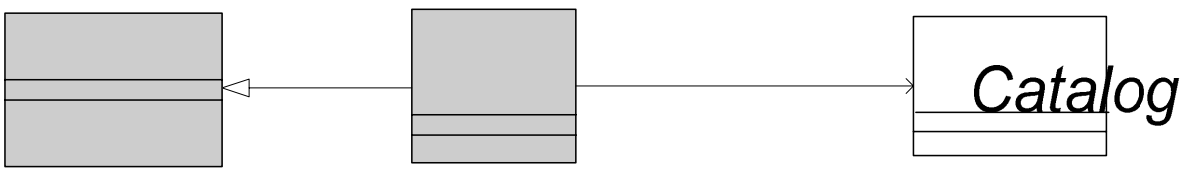


Figure 3-2: A Data Store Object

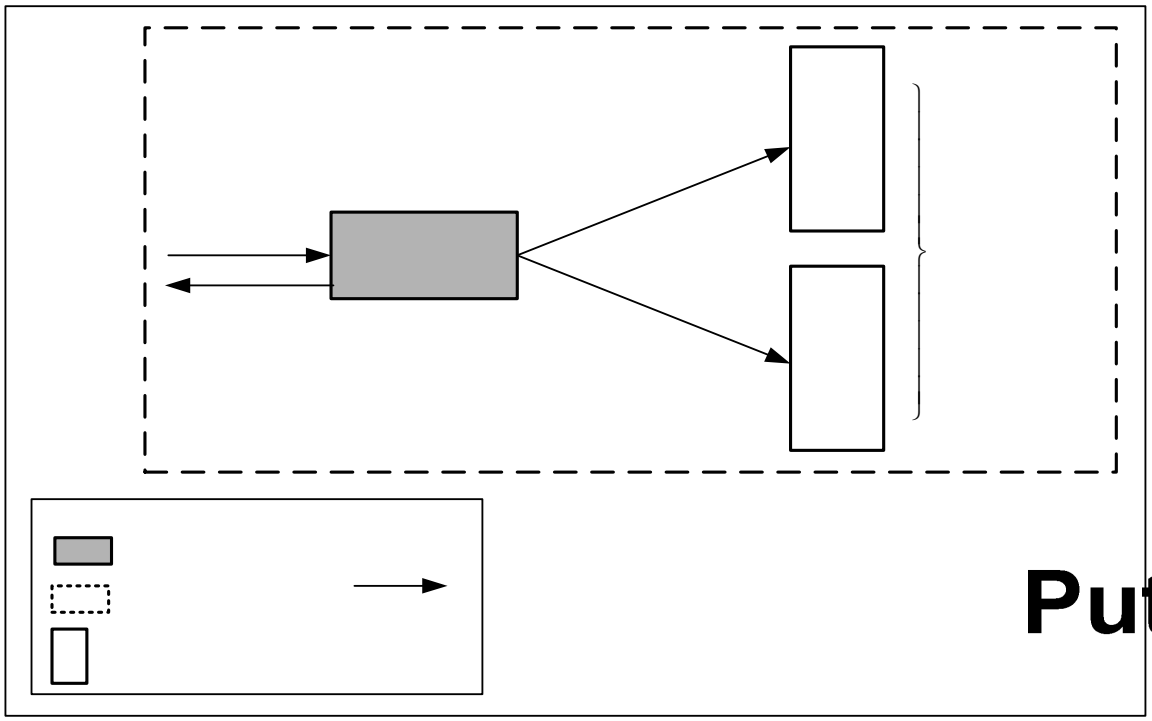


Figure 3-3: The *Put* Operation of the Data Store Object

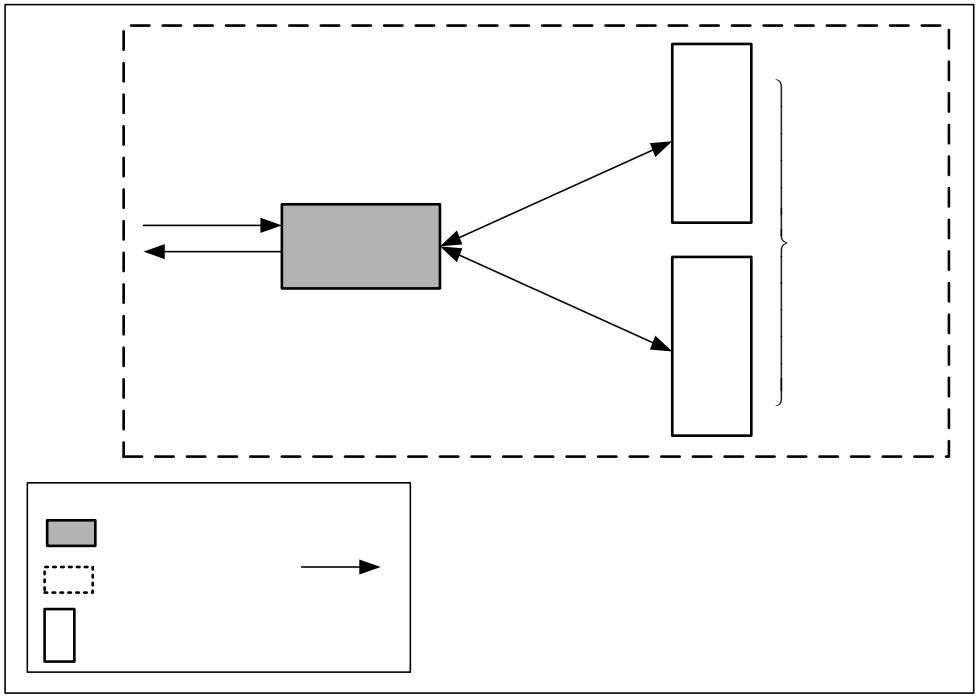


Figure 3-4: The *Get* Operation of the Data Store Object

3.1.2 QUERY OBJECT

The *query object* shown in figure 3-5 enables retrieval of data objects. Data objects (shown as DOs in figure 3-5) are retrieved using the *find* operation. The find operation takes an *expression* parameter representing a specific search criterion for the underlying physical data storage. Each matching data object is then returned to the caller of the find operation. A find invocation may return zero or more data objects. Figure 3-6 visually describes an example of the find operation and the data flow between the query object component and the respective physical data stores it communicates with.

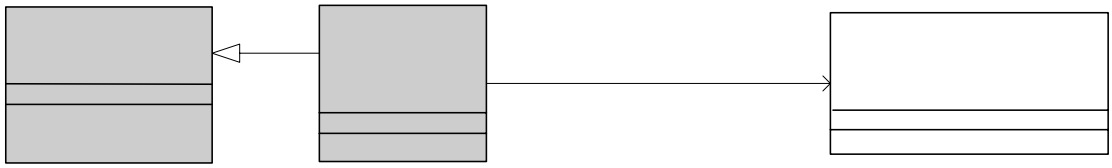


Figure 3-5: A Query Object

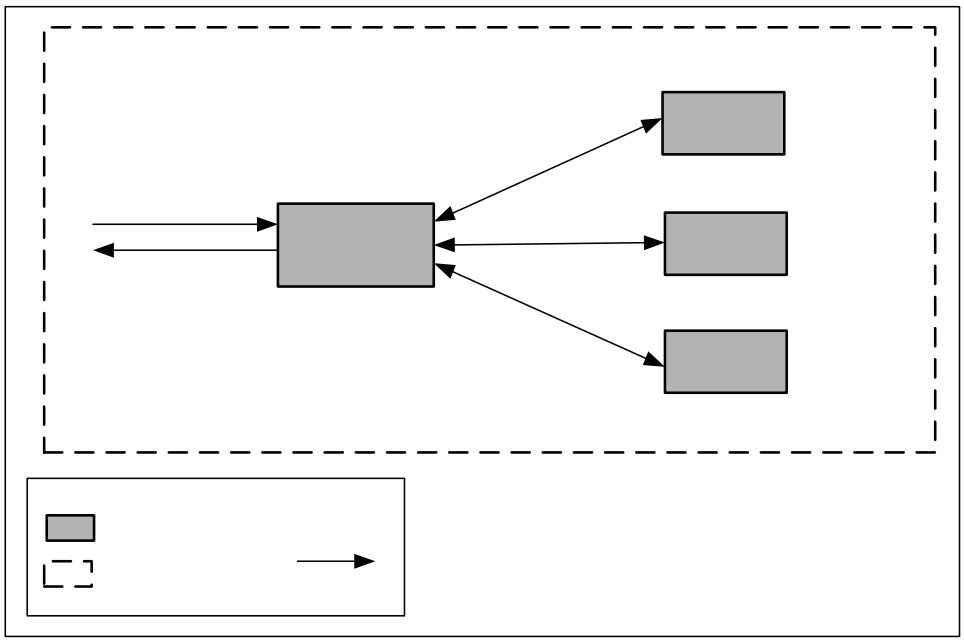


Figure 3-6: The *Find* Operation of the Query Object

3.2 ADVANCED INFORMATION MANAGEMENT OBJECTS

Advanced Information Management Objects (aIMOs) are components composed from one or more pIMOs. aIMOs leverage pIMOs' primitive data store and retrieval functions to arrive at complex capabilities. Examples of these capabilities include ingestion of data into repositories, federated search across heterogeneous repositories using registries, and the like. The set of aIMOs presented in this document is not meant to be comprehensive. There are

other aIMOs, but the set presented here represents a sound cross-section of advanced components that span the typical usage scenarios involved in data systems. In the rest of this section, the following aIMO components are discussed in more detail: *Repository Service Objects*, *Registry Service Objects*, *Product Service Objects*, *Archive Service Objects*, and *Query Service Objects.*

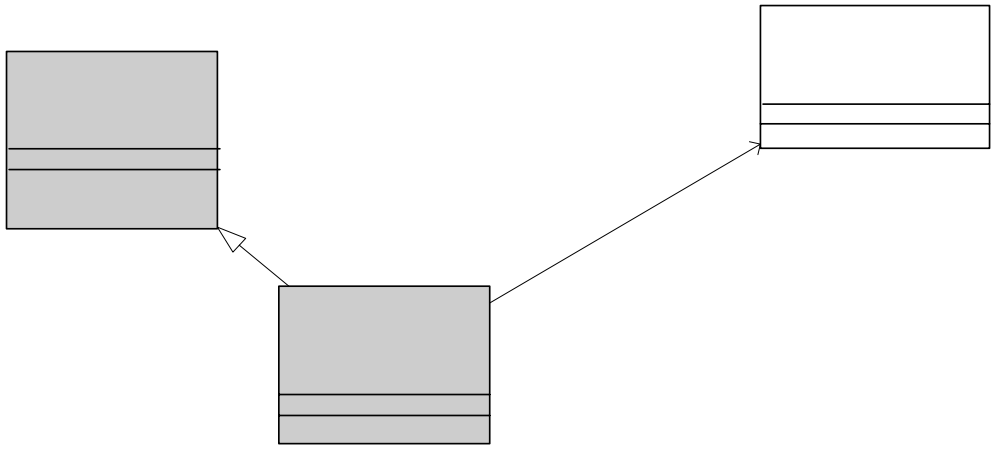


Figure 3-7: Repository Service Object

3.2.1 REPOSITORY SERVICE OBJECT

The *repository service object* component is depicted in figure 3-7. Repository service objects are responsible for management of an underlying data store object or the physical data store. The repository service object differs from a data store object by a myriad of properties that are typically considered *non-functional*. These properties include *scalability*, *dependability*, *uniformity*, and other quality attributes. In this context, repository service objects provide the same *get* and *put* methods that the data store object provides. However, whereas a data store object may not scale across many underlying physical data stores, may not be dependable 24×7, and may not provide a uniform software interface, a repository service object is responsible for delivering non-trivial quality of service in each of these non-functional properties.

Its primary interface is a *repository request* that can be used to manage information objects (IOs). Information objects can be retrieved from the repository via the repository request interface, and a response from the repository is provided. The repository service object also provides basic *get* and *put* capabilities of information objects using the capabilities of its associated data store object.

3.2.1.1 A Taxonomy of Repository Service Objects

Information architecture makes a distinction among different types of repository service objects, along several dimensions. There are three main dimensions in a repository service object taxonomy: *repository object type*, *object properties*, and *object description*, each of which are further explained in this section.

First, repository service objects are identified via their *type*. Type provides a quantifiable grouping for a family of repositories with similar functional and non-functional properties. This document identifies three key repository types: *data Store*, *operational archive*, and *long-term archive*. The *object properties* dimension serves as a general grouping of various functional and non-functional properties a repository might have. At the time of preparing this document, the properties dimension covers the entire scope of properties for a particular repository. In the long term however, properties will be categorized as dimensions of comparison and classification between different repository service objects. Potential dimensions of repositories include *compositionality*, referring to the lower-level and higher-level organization of the sub-components of a repository; *supported data objects*, referring to the type of data objects that a repository is responsible for storing; *permanence*, referring to the non-functional property of how long the data is guaranteed safe and reliable shelter within a repository; and finally *interface richness*, referring to the repository's ability to natively handle either primitive get/put operations, or higher level operations possibly requiring both querying and processing of data being returned. The last dimension in the current taxonomy, *object description*, identifies key services and responsibilities of the repository when deployed together with a set of other software components. Table 3-1 lists the current taxonomy and classification of repositories.

Table 3-1: A Taxonomy of Repository Service Objects

Repository Object Type	Object Properties	Object Description
Data Store	Primitive Component (e.g., DBMS, and File system).	Basic Data Store component described in 3.1 sits behind Data Store Object and supports Repository Interface to *get* and *put* data (lower level data such as streams and bits).
Operational Archive	Component that stores data products and higher level products, possibly including metadata. Supports retrieval of data products through possibly complex methods, and processing. No support for permanence. Stores products for short term (e.g., less than 10 years), and allows retrieval of products.	Advanced Component supporting retrieval of possibly complex data products, including their metadata. Repository where writes are frequent and reads are frequent. Data products stored in this type of archive will be updated and versioned. Examples of products stored in this archive are command sequence products sent using spacecraft telemetry.
Long-term Archive	Stores products for long term archiving, and supports basic archive functionality.	Archive for long-term preservation of data products, and data permanence. Supports basic archive functional interfaces (e.g., get, put).

3.2.2 REGISTRY SERVICE OBJECT

The *registry service object* component provides an interface to retrieve metadata objects. There are two special types of metadata objects which most current registries are able to return, other than the basic *metadata object* described in section 2. The first type is a *service description* metadata object. A service description is some metadata document that describes the basic components of a service, such as its interface and its accepted parameters and values; a Web Services Description Language (WSDL) document would be an example of this. The second type of metadata object returned by most registry service objects is the *resource* metadata object. A resource metadata object is typically simple keyword-value paired information about an information object, such as an individual science data product, or a science data set. The registry service object returns metadata objects which satisfy a particular query expression provided by the user of the *metadataQuery* interface. Figure 3-8 depicts a registry service object.

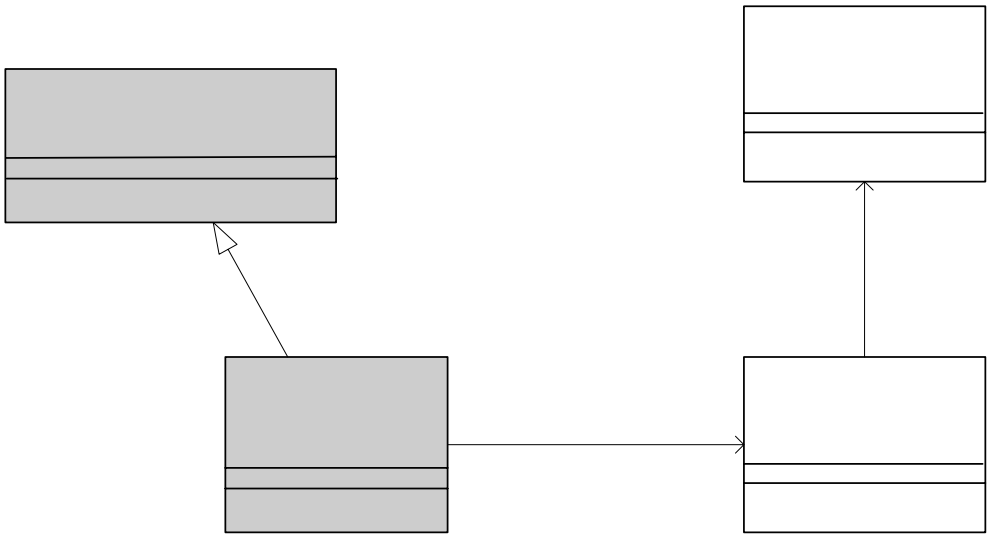


Figure 3-8: A Registry Service Object

Similar to the repository service object, there also exist different *classes* of registry service objects. A representative subset of these classes is identified below.

3.2.2.1 A Taxonomy of Registry Service Objects

This taxonomy identifies three main classes of registries and then classifies them along a particular set of dimensions: the *registry type*, the *return object types*, and *query interface parameters*.

The three main types of registries are *metadata registry, service registry*, and *resource registry.* The metadata registry returns structural information describing the structure of the metadata. This is sometimes referred to as a meta-meta-model. Subsequently, the object returned from a metadata registry is a meta-metadata object. Queries to the metadata registry are formulated via specification of constraints and values assigned to a set of data elements.

Constraints and values are specified either implicitly by querying the data element properties, or explicitly by specifying the data element's ID (see reference [19]).

The service registry provides an interface to search for functional services that perform a needed action specified by a user. Service registries manage descriptions of service interfaces (called *service descriptions*), including their respective locations, methods, and method parameters. New technological standards such as WSDL (reference [28]) provide an implementation-level facility for service descriptions. An additional implementation of a service description and its respective service registry exists in the form of the *Profile Server* and *Resource Profile* components specified in references [10], [11], and [22]. Service descriptions are important because they describe software methods, software systems, and Web resources using metadata. Because of this, they can be queried to retrieve a *service endpoint* (essentially a pointer to the service's location), and metadata describing how to invoke the particular service. This helps to facilitate the use and consumption of services dynamically via software rather than explicit invocations and requests.

The third type of registry, the resource registry, while capable of describing any resource or object, is used specifically for describing information objects such as science data products and data sets. Science catalogs such as the SIMBAD Astrophysics Catalog (reference [3]) are examples of resource registries that serve information objects. Resource registries can also point to other resource registries to enable discovery of information objects across distributed registries.

The classification dimensions introduced here effectively categorize the functional properties of each type of registry, leaving the non-functional classification unspecified at this point. This type of classification of non-functional registry service properties is very important, and this contribution is an element of on-going work within this document and within the Information Architecture (IA) Working Group. The taxonomy of registry service objects is summarized in table 3-2.

Table 3-2: A Taxonomy of Registry Service Objects

Registry Type	Return Object Types	Query Interface Parameters
Metadata Registry	Data Dictionaries, Data Elements	Query for Data Element properties, or Data Element IDs, or Data Dictionary IDs
Service Registry	Service Endpoints, Service Metadata (interface properties, interface type, return schema)	Query for Service properties
Resource Registry	Data Products, Resource Registry Locations	Data Resource properties

3.2.3 PRODUCT SERVICE OBJECT

The next aIMO is the *product service object*. The product service object contains a repository service object, coupled with a query object, and a domain processing or transformation object. The *domain processing* object is a functional component that provides specialized processing of a data object to transform it from one object type to another. This is critical in the era of providing on-the-fly processing of data to other users and systems and allows for specialization of a core software infrastructure on a product-type specific basis. In fact, domain processing objects can be externalized and registered on a product-type basis so as to require that the object is called as part of the retrieval process. Processing can involve functions such as science level processing, compression, decompression, scaling (in the case of an image), format conversion, and many other transformations

The product service object serves as a common interface to heterogeneous data sources and allows for the querying the information objects (shown as IO in figure 3-9) via a query expression. The query expression is passed along to the internal query object, which in turn evaluates the query expression and transfers it into a sequence of *get* calls to the repository service object, including execution of any specialized data processing objects. A product service object is shown in figure 3-9.

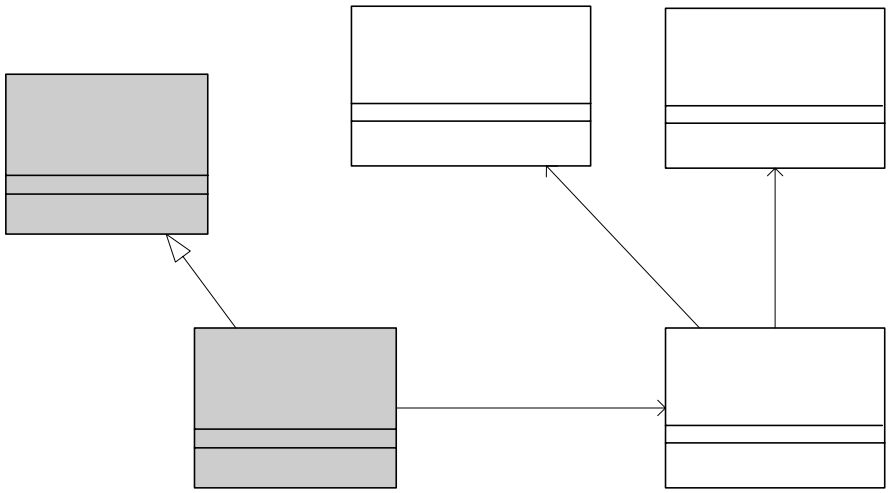


Figure 3-9: A Product Service Object

3.2.4 ARCHIVE SERVICE OBJECT

Archive service objects are responsible for (a) ingestion of data objects into a repository, and (b) ingestion of metadata objects into an accompanying registry. The ingestion of both metadata and data objects can be performed using a task processing approach: the users define *tasks* formulating the ingestion process of information objects (shown as IO in figure 3-10). These tasks can then be managed via a rule-based policy which, given a set of criteria such as time, task type, ingestion type, etc., determines when a particular task, or set of tasks, should be executed for a given ingestion. This rule-based task processing is often referred to

as *workflow* (see references [4], [12], and [13]) and can execute externalized objects such as the *Domain Processing Object* discussed in 3.2.3. This would enable a workflow-oriented archive service to construct a pipeline for ingestion and processing of level science products from missions. The externalization of a domain processing object would allow science data processors to run on appropriate scalable hardware, such as computational clusters, constructing an architecture for science processing and archive. In fact, this component was implemented for the SeaWinds Earth science instrument that was part of the payload for the ADEOS II satellite. This type of ingestion process is shown as the *ingest service object* component in figure 3-10.

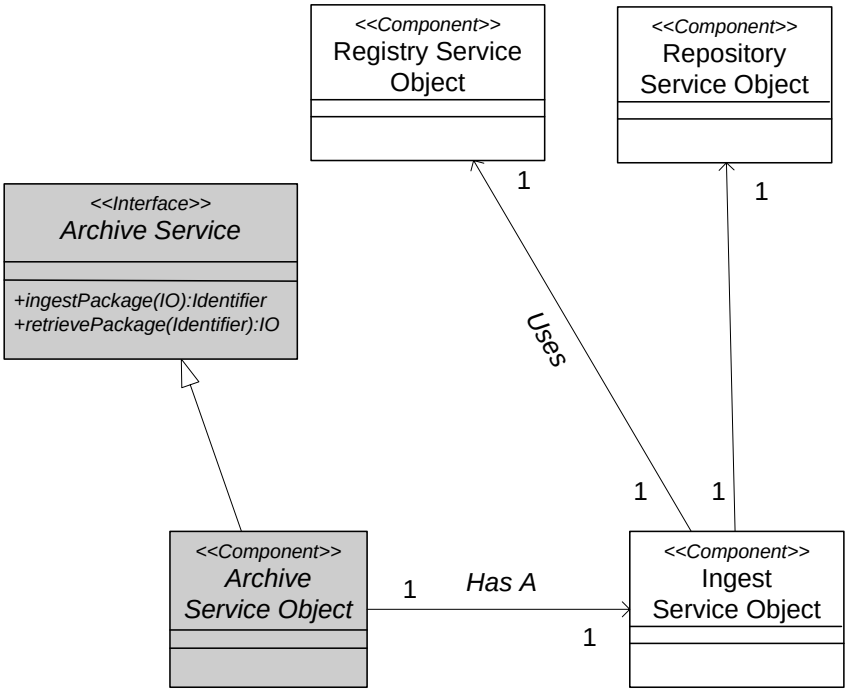


Figure 3-10: An Archive Service Object

Archive service objects also have the capability of handling transaction-based ingestion of data and metadata objects, similar to the ingestion interface described in the OAIS model (reference [5]). This type of transaction capability would be provided by the ingest service object in figure 3-10, managing all aspects of ingesting an object into the archive (e.g., validation, registration, etc.). An archive service object is shown in figure 3-10.

3.2.5 QUERY SERVICE OBJECT

The final aIMO defined in this document is the *query service object*. The query service object manages routing of queries in order to discover and locate product service objects, repository service objects and registry service objects which contain information to satisfy user queries. Routing is accomplished by querying registry service objects in order to discover the location of the appropriate repository, or product service objects to ultimately locate the information objects (shown as IOs in figure 3-11) that satisfy a user's query. Once the service objects have returned the information objects that satisfy the query, the information objects are aggregated and returned to the query service object. At that point, the query service object can perform processing such as packaging, translations to other formats, and other types of advanced processing. These advanced processing capabilities are shown as the *domain processing object* in figure 3-11 and discussed in 3.2.3 as an externalized component of the product service. Figure 3-11 depicts a query service object.

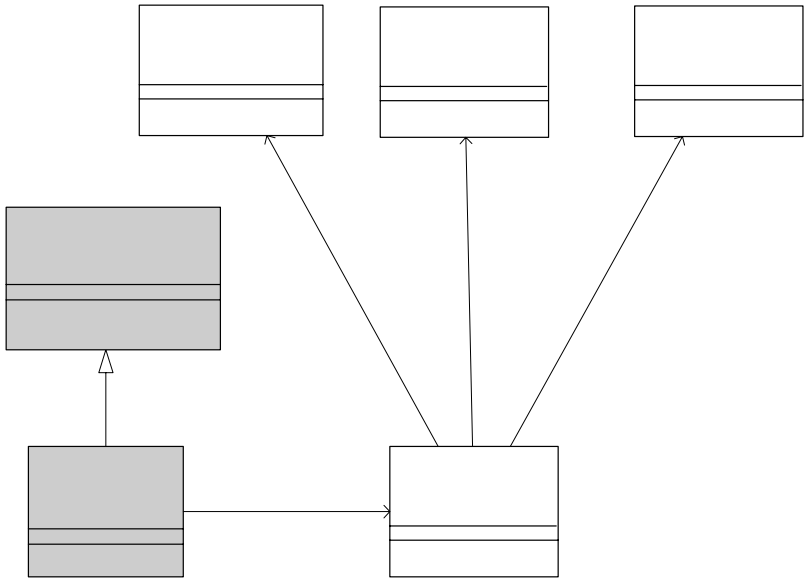


Figure 3-11: A Query Service Object

4 SPACE DATA SYSTEMS

This section provides information about related space data system projects which use components of the information architecture described in this document. The use of information architecture components in each project is summarized in table 4-1.

Table 4-1: Example Projects Using Related RASIM Concepts

Project	Information Architecture Concepts Used
OAIS	CCSDS reference model describing information objects, information packages, archive service object.
SPACEGRID	Uses concept of information objects and registry service objects.
EOSDIS	Uses concepts including meta-models, domain models, metadata objects, information objects for a national Earth science program within NASA.
European Data Grid	Uses concept of information objects, information packages, archive service object, registry service object for a national grid system.
National Virtual Observatory	Uses concepts of information objects, information packages, archive service object, registry service object for an international astrophysics interoperability effort.
Planetary Data System	Uses concepts of information objects, information packages, archive service object, registry service object for a national planetary science grid system within NASA.

4.1 OAIS

The CCSDS OAIS reference model (reference [5]) has made metadata a key element in terms of the ability to validate ingestion of data products and understand data product format, which is a key element of information architecture. OAIS defines the notion of an 'open archive'. An open archive is an archive service object that interacts with three main outside entities: *Producers*, *Consumers* and *Management*. In general,

- producers produce Submission Information Packages (SIPs) to send to the OAIS compliant archive;

- consumers consume Dissemination Information Packages (DIPs) that they retrieve from the OAIS compliant archive;
- management constitutes outside entities responsible for managing data within the archive and are not involved in the day-to-day operations of the component.

In addition to SIPs and DIPs, OAIS archives also deal with Archival Information Packages (AIPs), which are created within the OAIS archive from SIPs. With respect to information architecture, the OAIS DIPs, SIPs, and AIPs could all be considered information objects conforming to each respective package format specified in reference [5].

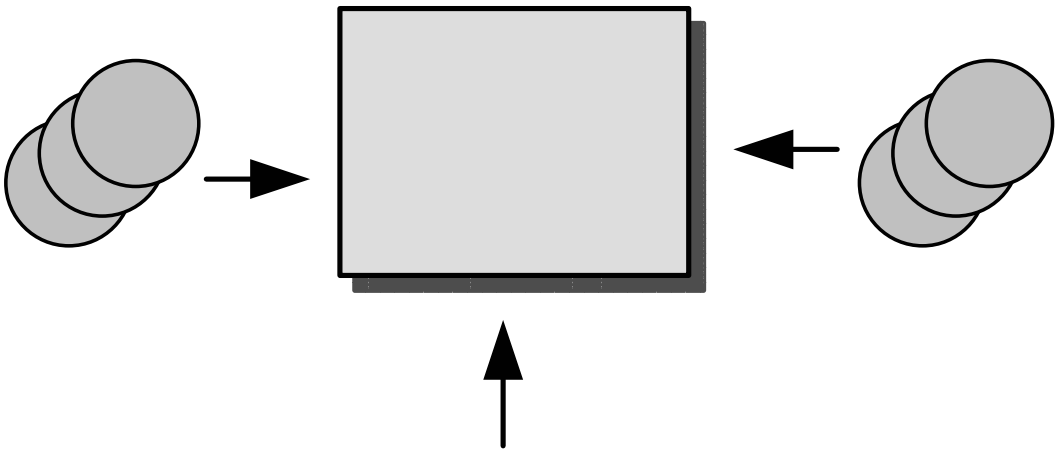


Figure 4-1: The Open Archival Information System Reference Model

OAIS-compliant archives are in the business of preserving, providing, managing, and collecting information. Inherently they are most related to the archive service object described in 0; however, since the OAIS reference model defines the standard data structures that an OAIS archive should use, which are all domain specific instantiations of information objects, OAIS archives could utilize the information objects described in this document.

4.2 GRIDS

Recent work in the *grid* community (see reference [16]) has characterized a class of distributed data interoperable systems as *data grids* (references [8], [9], [23], and [26]). Data grids involve the identification of (different classes of) metadata (reference [26]), used to make heterogeneous software systems interoperable. In the next paragraphs, some overviews of grid projects at various space agencies are listed. Each subsection details how each grid project uses the components of information architecture.

Producers

4.2.1 SPACEGRID

ESA's SpaceGRID Study [21] commenced in 2001 and concluded in 2003 with the goal of assessing how ESA could infuse grid technology into various Earth observing and space missions to support (1) distributed data management, (2) data distribution, (3) data access, and (4) a common architectural approach to designing, implementing, and deploying software to support such activities. The study spanned several different disciplines including Earth Observation, Space Research, Solar System Research, and Mechanical Engineering. Results of the study included identification of 240 user requirements for grids, 146 of which were considered 'common', denoting the fact that the requirement was considered useful for at least three of the study domains. Of the 146 requirements, a cross-section of design areas were identified, and user-desired requirements of grids were listed as:

- Flexibility;
- Portal;
- Security;
- Distributed Access;
- Human Computer Interface;
- Virtual Organization;
- Collaborative Environment;
- Reliability.

Figure 4-2 depicts the proposed SpaceGRID infrastructure, which is very similar to the service objects and architectural model described in this document. It is a layered architectural model, with client applications at the top-most layer making calls through an organizational API. The organization's API makes use of grid services, which in turn use grid infrastructure to access both 'hard' (hardware-based) and 'soft' (software-based) distributed resources.

The data that is made available by grid infrastructure in the ESA report is searched using metadata catalogs. These catalogs can be thought of as storing metadata objects, which in turn point to data objects desired by the user. Effectively, the grid infrastructure described in the SpaceGRID report is distributing, searching, and delivering information objects to users.

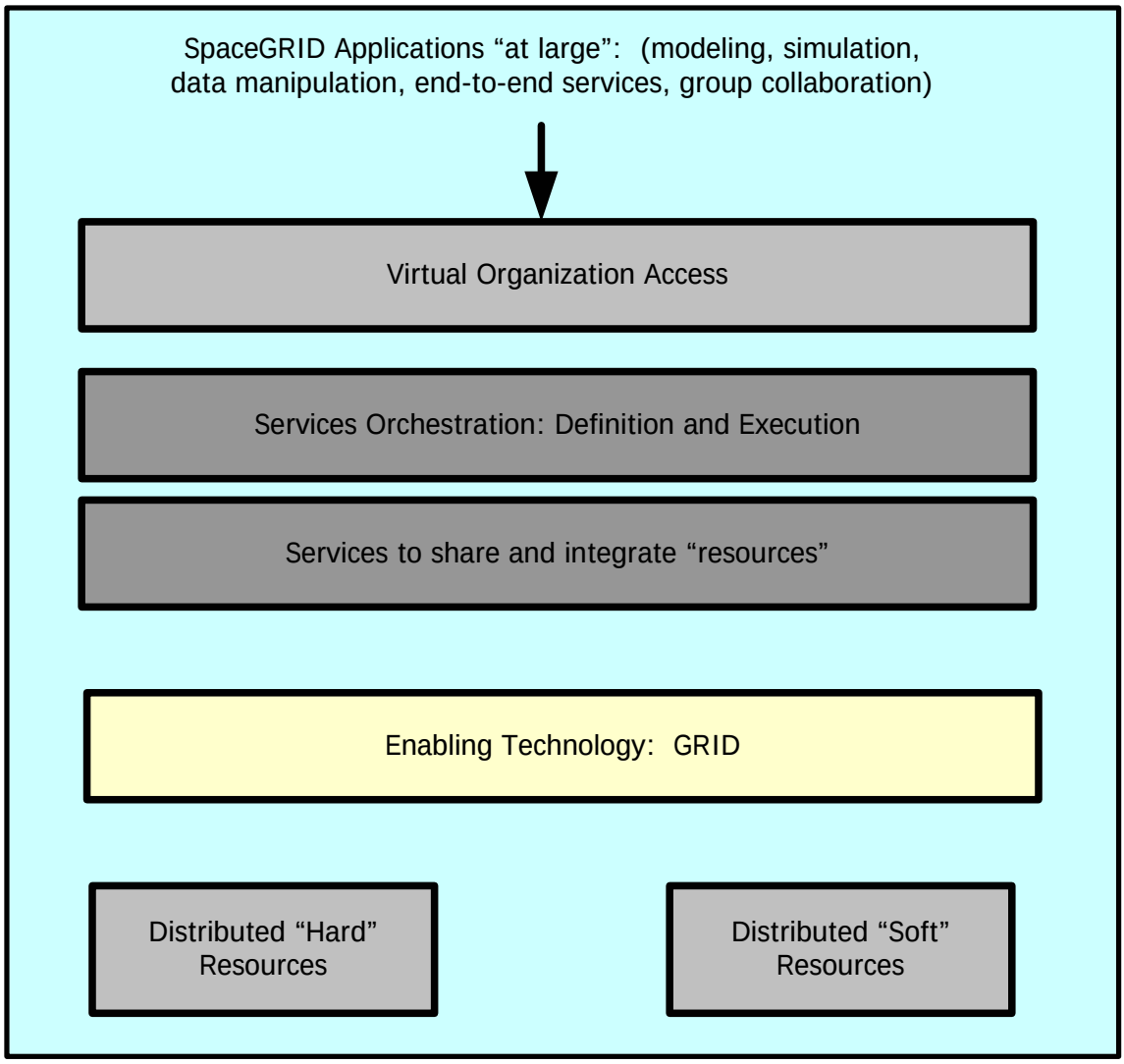


Figure 4-2: SpaceGRID Proposed Infrastructure

4.2.2 EOSDIS

NASA's Earth Observing System Data and Information System, or EOSDIS, was a preliminary investigation into how NASA could support data distribution, processing, archival and storage of Earth science data sets produced by Earth observing missions. EOSDIS was an excellent early example of the problems with state-of-the-art information systems technology circa 1996. So-called "one-off" data systems were being produced across the country, and viable data sets could not be accessed, distributed and ultimately used. This required sending data on removable media and ultimately increased the amount of time necessary to engage in science. The goal of EOSDIS was to bridge the gap between existing Earth science data systems, and unlock their data, and make it available to scientists.

Many of the conclusions from EOSDIS were early precursors to the study and ultimate adoption and acceptance of the *grid paradigm*. The relation between EOSDIS and this

document lies in the fact that EOSDIS is a domain-specific example of (1) Earth science specific information objects, (2) Earth science meta-models, (3) Earth science metadata objects, and (4) Earth science domain models and ontologies.

4.2.3 EUROPEAN DATA GRID

The European Data Grid (EDG) is an EU- and ESA-funded project aimed at enabling access to geographically distributed data and computational resources (see reference [15]). EDG uses Globus Toolkit technology to support base grid infrastructure and then builds data-specific services on top of the underlying grid infrastructure. These data-specific services are services such as *replica management*, *metadata management*, and *storage management*. Because of its focus on data and metadata, EDG is highly related to this document. The EDG system manages, distributes, processes, and archives information objects. The metadata objects are stored in metadata catalogs, and the data objects are stored transparently in an underlying storage system. Users use software components, similar to those described in section 3, to query for and retrieve application information objects and information packages made available by the EDG system.

4.2.4 NATIONAL VIRTUAL OBSERVATORY

The National Virtual Observatory, or NVO, is an NSF-funded project whose goal is to enable science by greatly enhancing access to data and computational resources. NVO uses the Globus Toolkit (see references [16] and [20]) grid middleware infrastructure to distribute, process, retrieve, and search for astrophysical science data. The components of NVO are essentially the components described in this document: (1) a well defined information architecture, including information objects (or astrophysical data products), (2) common models to describe the information objects, and (3) software service objects (in the form of grid services) to exchange science data.

4.2.5 PLANETARY DATA SYSTEM

The Planetary Data System (PDS) is a NASA-funded program that is responsible for distributing, archiving, and managing planetary science data collected from all NASA funded planetary missions. The PDS consists of seven ‘discipline nodes’ and an engineering and management node. Each node resides at a different U.S. university or government agency and is managed autonomously.

For many years PDS distributed its data and metadata (i.e., its information objects) on physical media, primarily CD-ROM. Each CD-ROM was formatted according to a ‘home-grown’ directory layout structure called an *archive volume*, which later was turned into a PDS standard. PDS metadata objects were constructed using a common, well-structured set of approximately 1200 metadata elements, such as ‘Target Name’ and ‘Instrument Type’, that were identified from the onset of the PDS project by planetary scientists. Beginning in the late 1990s the advent of the WWW and the increasing data volumes of missions led

NASA managers to impose a new paradigm for distributing data to the users of the PDS: data and metadata were now to be distributed electronically, with a single, unified Web portal as the gateway to the information. This posed a challenge due to the geographically distributed nature of the data. Consequently, PDS adopted a distributed software framework called the Object Oriented Data Technology (OODT) (see reference [11]) framework which provided a set of services for capture and distribution of data within a distributed environment. As a result a Web portal and accompanying infrastructure to distribute PDS data and metadata over the Internet was built in 2001 using the OODT middleware. OODT provides an implementation of several of the information management objects described in this document, in particular, a registry service object, an archive service object, and a product service object. Several of these components were provided to PDS as a distributed architecture called 'PDS-D'. Scientists and the user community download PDS 'products' via a unified portal connected to a set of common infrastructure services that are geographically distributed and connected to the planetary science repositories located at the PDS nodes.

ANNEX A

APPLICABLE STANDARDS USED IN THIS DOCUMENT

This annex details the applicable standards used in this document, such as the modeling notation standards, and the governing CCSDS standards that apply to this document.

A1 UML MODELING STANDARDS

All of the UML figures drawn in this report are drawn using the UML 1.0 notation. UML was chosen because of its broad applicability and use in the design of modern software-intensive systems.

UML is used to represent data concepts in this document, along with software object concepts and relationships. The following UML diagrams were used extensively:

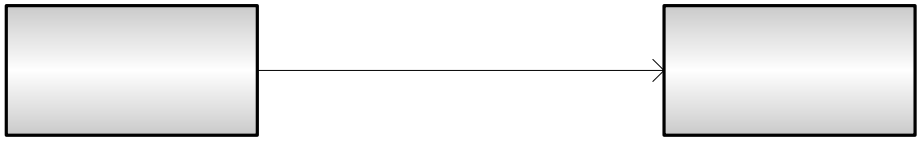
A1.1 UML CLASS DIAGRAMS

The UML Class diagram is used to show objects and relationships amongst objects. In particular UML defines *classes*, which are entities in a system that interact and interface with other entities. Classes can have attributes, interfaces, and relationships with other objects.

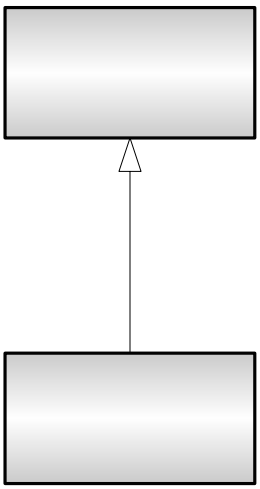
Primarily, the *association* relationship is used in this document. An association has a *name* and *direction*, which describes the name of the association (e.g., *has a*), and the direction (either uni- or bi-) from which the association originates and ends. Associations can have *roles* at each of their ends. Roles define how one end of an association will behave in certain situations. Associations can also have *cardinalities* at each end of the relationship, specifying how many of each class connected to each end of the association participate in the association. In this document, the cardinality **1** denotes a single participant from a class. The cardinality ***** denotes more than one, or *many* participants from a class.

Using the example below, we know that there is a *Soda Machine* class and a *Student* class, with an association between them with a name of *dispenses sodas to*, originating from the soda machine class and terminating at the student class. We know that for one soda machine, there can be many students that it dispenses sodas to. Note that there are no roles defined in this example.

The association relationship shown above between soda machines and students can also be shown using the following notation:



There is one other type of relationships used in the UML modeling notation in this document to represent relationships between two objects, or classes. The type of relationship is a *generalization* relationship. Generalization relationships indicate that one class is a specialization of a more generic class, or that one class is a child of another class, which is its parent. The generalization relationship is usually shown with an unfilled arrow end at the end of an association line. In the example below, we have the parent class *Bird*, and a child class, *Parrott*, which is a type of bird; therefore, it is a specialization of the more generic parent class. As such, there is a generalization relationship between the two classes.



Soda Machine

Please see reference [1] and [2] for further clarifications on the UML modeling notation used in this document.

A2 CCSDS

Table A-1: CCSDS Information Standards Mapped to Information Architecture Concept

Information Architecture Concept	CCSDS Standard
Meta-Model Specification **(section 2)**	*DEDSL (Data Entity Dictionary Specification Language)* http://www.ccsds.org/documents/647x1b1.pdf
Archive Ingestion Model **(section 3)**	*Reference Model for an Open Archival Information System (OAIS)* http://www.ccsds.org/documents/650x0b1.pdf
Data Element Semantics and Specification **(section 2)**	*The Data Description Language EAST Specification (CCSD0010).* Blue Book. Issue 2. November 2000. http://www.ccsds.org/documents/644x0b2.pdf
Specification of Application Information Object Format **(section 2)**	*Information Interchange Specification* http://www.ccsds.org/documents/642x1g1.pdf
Data Value Representation **(section 2)**	*Parameter Value Language Specification (CCSD0006 and CCSD0008).* Blue Book. Issue 2. June 2000. http://www.ccsds.org/documents/641x0b2.pdf
Packaging Specification (**section 2**)	*XML Formatted Data Unit (XFDU) Structure and Construction Rules.* White Book, Issue 2, September 2004. http://www.ccsdsrg/docu/dscgi/ds.py/Get/File-1912/IPRWBv2a.doc
Data Object Format Specification (**section 2**)	*Standard Formatted Data Units — Control Authority Data Structures.* Blue Book. Issue 1. November 1994. http://www.ccsds.org/documents/632x0b1.pdf

This section presents a mapping of existing CCSDS Standards to the data and software components and ideas discussed in this document.

ANNEX B

ABBREVIATIONS AND ACRONYMS

aIMO	advanced information management object
AIO	application information object
AIP	archival information package
CFDP	CCSDS File Delivery Protocol
DDR	data description record
DEDSL	data entity dictionary specification language
DIP	dissemination information package
DO	data object
DSO	data store object
EAST	Enhanced Ada SubseT
ECS	EOSDIS Core System
EDG	European data grid
EOS	Earth Observing System
EOSDIS	EOS Data and Information System
ESA	European Space Agency
IMO	information management object
IO	information object
MOF	meta-object facility
NSF	National Science Foundation
NVO	National Virtual Observatory
OAIS	Open Archival Information System
ODL	Object Description Language

OMG	Object Management Group
OODT	object oriented data technology
PDS	Planetary Data System
pIMOs	primitive information management objects
PVL	Parameter Value Language
QO	query object
RASIM	Reference Architecture for Space Information Management
RDF	resource description framework
SGML	standard generalized markup language
SIP	submission information package
SLE	Space Link Extension
SPASE	Space Physics Archive Search and Exchange
UML	Unified Modeling Language
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XFDU	XML Formatted Data Unit
XML	Extensible Markup Language