Consultative
Committee for
Space Data Systems

REPORT CONCERNING SPACE
DATA SYSTEM STANDARDS

REFERENCE ARCHITECTURE FOR SPACE INFORMATION MANAGEMENT

CCSDS 312.0-G-1

GREEN BOOK

September 2005

AUTHORITY

Issue:	Green Book, Issue 1
Date:	September 2005
Location:	Not Applicable

(When approved). 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 Recommendations is detailed in *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
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC20546, USA

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommendation 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.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency(JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- Russian Space Agency (RSA)/Russian Federation.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- 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.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Federal Science Policy Office (FSPO)/Belgium.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space and Astronautical Science (ISAS)/Japan.

- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- 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 and Issue	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 working group for review
1.3	Draft Information Architecture for Space Data Systems	02/05/2005	Revised Chapter 2: Mattmann and Hughes
1.4	Draft Information Architecture for Space Data Systems	03/03/2005	Organizational changes to the structure and content of chapter 2 after discussion with Crichton.
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 Greece.
1.6	Reference Architecture for Space Information Management Green Book Submission	09/01/2005	CCSDS 312-0.G-1

CONTENTS

Section Page

FIGURES

TABLES

1 INTRODUCTION

In the absence of information system standards for interoperability and cross support, we have seen systems developed that do not allow the exchange of information throughout ground and flight systems and across agency data systems. 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.

Another goal of this document is to define 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). This document also serves to discuss in detail the Information Viewpoint and Information Management Objects defined by Reference Model for Open Distributed Processing (RM-ODP) and further by the Reference Architecture for Space Data Systems (RASDS) [25] being developed by the CCSDS System Architecture Working Group.

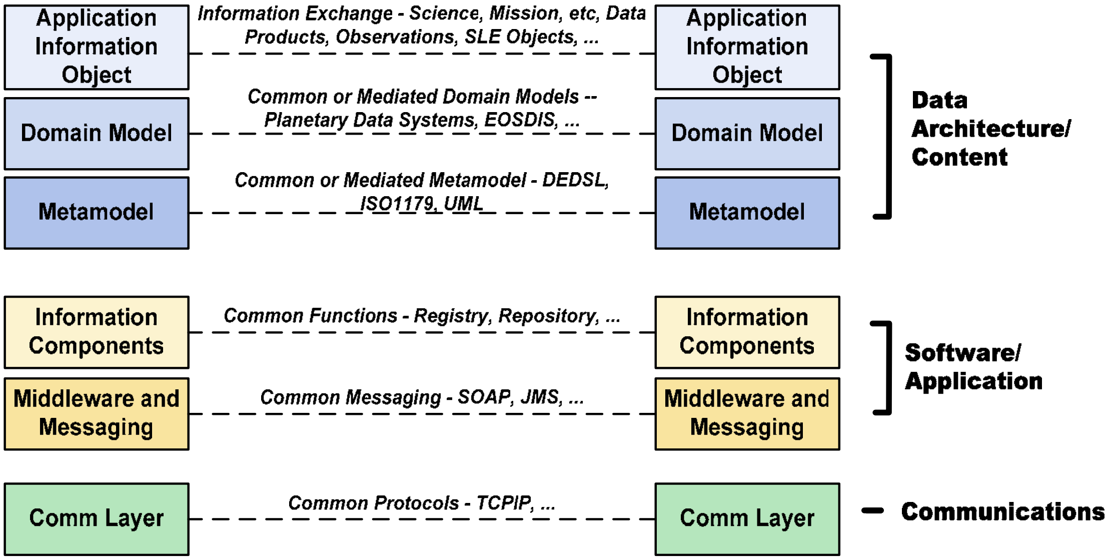
Application Information Object
Information Exchange - Science, Mission, etc, Data Products, Observations, SLE Objects, ...
Application Information Object
Domain Model
Common or Mediated Domain Models -- Planetary Data Systems, EOSDIS, ...
Domain Model
Metamodel
Common or Mediated Metamodel - DEDSL, ISO1179, UML
Metamodel
Data Architecture/ Content
Information Components
Common Functions - Registry, Repository, ...
Information Components
Middleware and Messaging
Common Messaging - SOAP, JMS, ...
Middleware and Messaging
Software/ Application
Comm Layer
Common Protocols - TCPIP, ...
Comm Layer
Communications

Figure 1. High-level abstract view of interoperable information architecture

This document introduces a layered view of the information architecture. To achieve interoperability both within and across domains, and across applications built based on this document; conformance at each layer must be achieved. Figure 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 the information products and messages flowing between application interfaces be defined along with common definitions for the information objects themselves, in order to achieve system interoperability. This architecture document purposely separates into chapters the information architecture from the information management components that implement that architecture.

1.1 SCOPE AND APPLICABILITY

This document is intended for those interested in using 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. They will be most valuable in complex environments such as space, but is by its very nature not limited to use in space, and 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	literally "data about data" information that describes another set of data
Meta Model	a model which describes another model
Schema	a means for defining the structure, content and to some extent, the semantics of data
Application Information Object	An object containing an internal Data Object and a Metadata Object.
Application Information Architecture	The notion of architecting information systems, with a focus on both data architecture, and software architectural concerns.
Data Architecture	The specification the overall structure, logical components, and the logical interrelationships of data and information.
Software Architecture	The specification of overall structure, behavior, logical components, and logical interrelationships of a software system.

Data Product	The result of an active function which produces data. The Data Product may be simple, and just include data value, or it may be complex, and contain both data, and metadata objects.

1.3 REFERENCES

1. The SIMBAD astronomical database (http://cdsweb.u-strasbg.fr/Simbad.html), 2004.
2. Blythe, J., Deelman, E. and Gil, Y. Automatically Composed Workflows for Grid Environments. *IEEE Intelligent Systems*, *19* (4). 16-23.
3. CCSDS. CCSDS OAIS Reference Model, 2002.
4. CCSDS. The Data Description Language (EAST) Specification, 2000.
5. CCSDS. Data Entity Dictionary Specification Language (DEDSL), 2000.
6. Chervenak, A., Deelman, E., Foster, I., Guy, L., Hoschek, W., Iamnitchi, A., Kesselman, C., Kunszt, P., Ripeanu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K. and Tierney, B., Giggle: A Framework for Constructing Scalable Replica Location Services. in *IEEE Supercomputing Conference*, (Baltimore, MD, 2002), IEEE Computer Society.
7. Chervenak, A.L., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*. 1-12.
8. Crichton, D., Hughes, S., Kelly, S. and Ramirez, P., A Component Framework Supporting Peer Services for Space Data Management. in *IEEE Aerospace Conference*, (Big Sky, Montana, 2002), IEEE Computer Society.
9. Crichton, D.J., Hughes, J.S. and Kelly, S. A Science Data System Architecture for Information Retrieval. in Wu, W., Xiong, H. and Shekhar, S. eds. *Clustering and Information Retrieval*, Kluwer Academic Publishers, 2003, 261-298.
10. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Blackburn, K., Lazzarini, A., Arbree, A., Cavanaugh, R. and Koranda, S. Mapping Complex Workflows onto Grid Environments. *Journal of Grid Computing*, *1* (1). 25-39.
11. Deelman, E., Plante, R., Kesselman, C., Singh, G., Su, M., Greene, G., Hanisch, R., Gaffney, N., Volpicelli, A., Annis, J., Sekhri, V., Budavari, T., Nieto-Santisteban, M., O'Mullane, W., Bohlender, D., McGlynn, T., Rots, A. and Pevunova, O., Grid-Based Galaxy Morphology Analysis for the National Virtual Observatory. in *IEEE Conference on Supercomputing*, (Phoenix, AZ, 2003), IEEE Computer Society.
12. EOSDIS. ECS Core Data Model (http://spg.gsfc.nasa.gov/standards/heritage/ECSDM_brief), 2005.
13. EU-DataGrid. The DataGrid Project (http://web.datagrid.cnr.it/LearnMore/index.jsp), 2004.
14. Globus. The Globus Alliance, 2004.
15. Gomaa, H., Menasce, D. and Kerschberg, L. A Software Architectural Design Method for Large-Scale Distributed Information Systems. *Distributed Systems Engineering*.
16. Hyperdictionary.com. Definition of Compression, web, 2004.
17. ISO/IEC. Framework for the Specification and Standardization of Data Elements. ISO/IEC ed., Geneva, 1999.
18. Kesselman, C., Foster, I. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputing Applications*. 1-25.

19. Marchetti, P.G., Ansari, S., Koning, H.-P.d., Fusco, L., Kreiner, G., Evans, H., Daly, E., Beco, S., Perrin, V., Perry, C., Vielcanet, P., Rodgers, D., Lei, F. and Morris, C. SpaceGRID Study Final Report. ESA ed. *ESA Technical Report*, ESA, 2002, 1-84.
20. Mattmann, C., Crichton, D., Hughes, J.S., Kelly, S. and Ramirez, P., Software Architecture for Large-scale, Distributed, Data-Intensive Systems. in *4th IEEE/IFIP Working Conference on Software Architecture (WICSA-4)*, (Oslo, Norway, 2004), IEEE Computer Society, To appear.
21. Moore, R.W., Baru, C., Marciano, R., Rajasekar, A. and Wan, M. Data-Intensive Computing. in Kesselman, C. and Foster, I. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman Publishers, 1999.
22. OMG. 2004, http://www.omg.org.
23. PDS. SFDU Usage (http://pds.jpl.nasa.gov/documents/sr/Chapter16.pdf), 2003.
24. Reich, L., XML Packaging for the Archiving and exchange of Binary Data and Metadata. in *Open Forum on Metadata Registries*, (2003).
25. Shames, P. and Yamada, T., Reference Architecture for Space Data Systems. in *5th International Symposium on Reducing the Cost of Spacecraft Ground Systems (RCSGO)*, (Pasadena, CA, 2003).
26. Singh, G., Bharathi, S., Chervenak, A., Deelman, E., Kesselman, C., Manohar, M., Patil, S. and Pearlman, L., A Metadata Catalog Service for Data-Intensive Applications. in *IEEE International Conference on Supercomputing*, (2003).
27. SPASE. A Space and Solar Physics Data Model (http://www.igpp.ucla.edu/spase/data/makedoc.php), 2005.
28. W3C. Extensible Markup Language (http://www.w3.org/TR/2004/REC-xml-20040204/), 2004.

2 INFORMATION ARCHITECTURE

2.1 INTRODUCTION

The *application information object*, used and described throughout the document as an *information object* is the cornerstone of defining and constructing a data-driven system through which data is described, exchanged, and retrieved.

The information object (shown in Figure 2) 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 [6]. This section starts by providing key definitions and is followed by a small taxonomy of information object types commonly used in information architecture and a simple example in the space data systems domain 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

According to the OAIS standard [3], *data objects* are either a physical object or a digital object as illustrated in Figure 3. The data object may be expressed as either a physical object (e.g., a moon rock) together with some representation information or it may be expressed as a digital object (i.e., a sequence of bits) together with the representation information giving meaning to those bits. This document uses a definition of data object that omits the physical object component. This makes the data object in this document equivalent to the OAIS digital object. In other words, in this document, a data object constitutes data as it is physically represented using a sequence of bits.

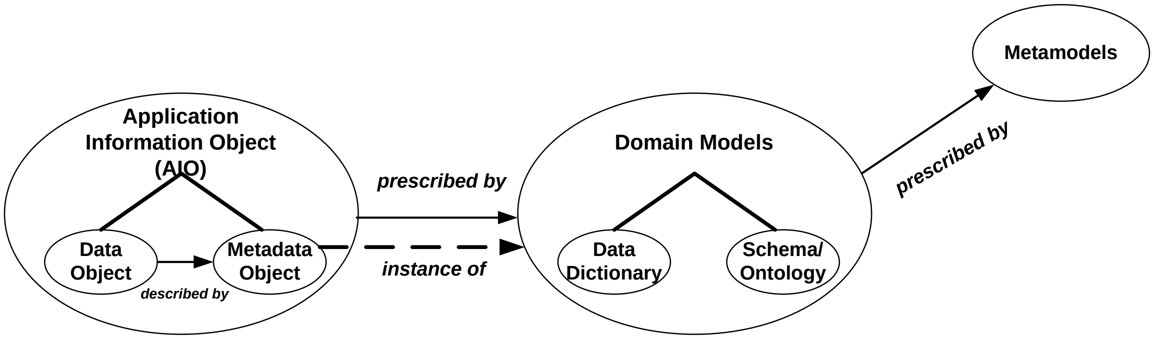

Figure 2. Information Object in Context

2.1.2 METADATA OBJECTS

Metadata objects in this document are a special class of data objects (or bits) that provide information about the data object. The OAIS reference model [6] defines representation, preservation description, and packaging information as three broad classifications of metadata. As shown in Figure 3, representation information includes structure (syntactic) and semantic information and preservation information includes reference, provenance, fixity, and context information. The scope of a metadata object used in this document includes both representation and preservation description information; only packaging information is excluded (shown in Figure 4). Also, the metadata objects described in this document might be atomic or comprised of a set of metadata sub-objects, each with a scope of one OAIS defined information subclass.

It is important to understand the relationship between data objects and metadata objects. Without the metadata object, essentially the data object is just a sequence of bits about which nothing is known: systems cannot unlock its information.

A metadata object does not have to describe an electronic resource, such as a data object. It could simply carry information, such as the description of a spacecraft. In this case, it simply provides information about a *thing* but can never return that *thing* to the user. When a metadata object and optional 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*, a mere examination of the correct location within the data object (specified

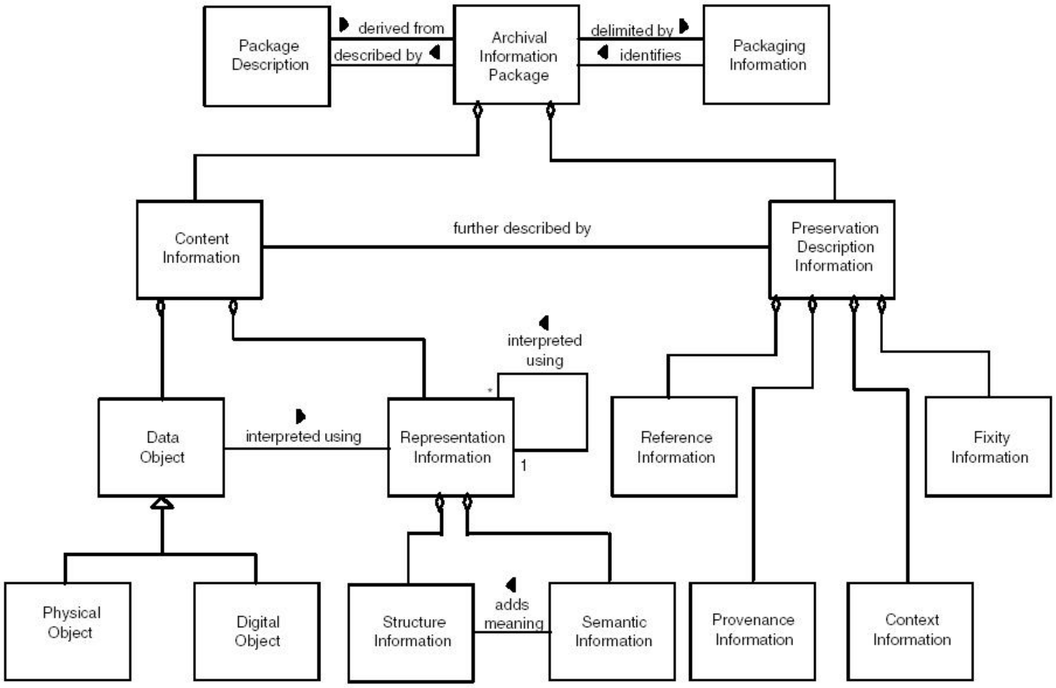

Figure 3 . OAIS Archival Information Package

by the metadata object) will reveal the value of the pixel.

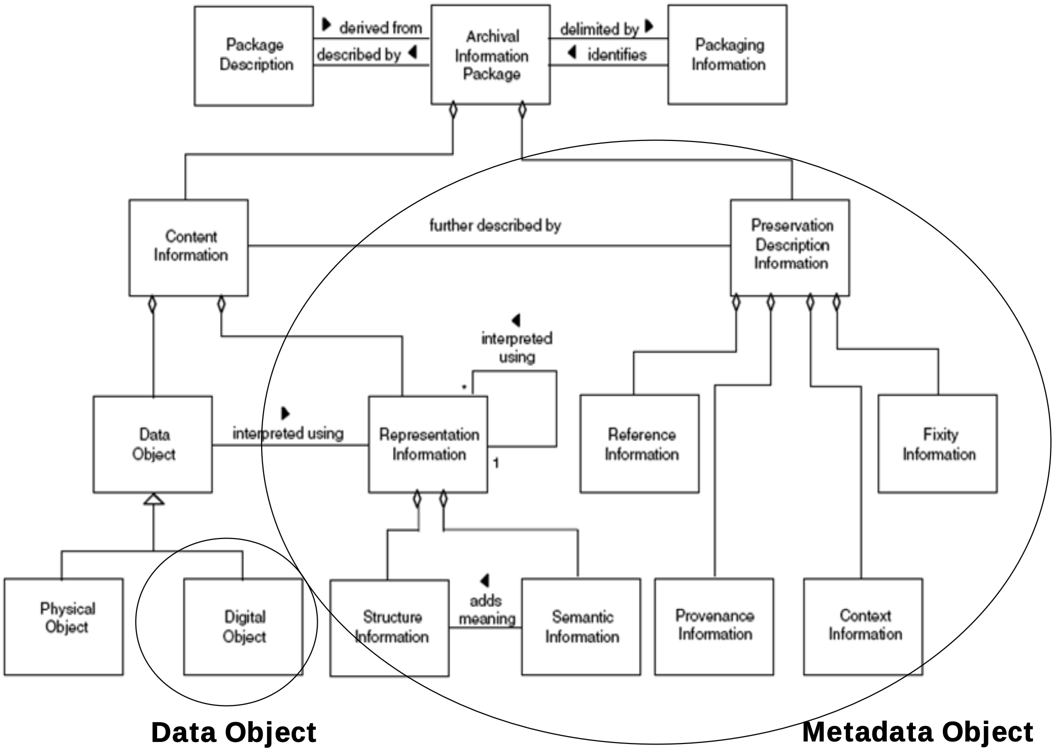

Figure 4. Information Architecture view of the OAIS Archival Model

In addition, because the metadata object is itself a data object, it also has to be described by a metadata object to indicate that it contains metadata (data that describes other data). This is seldom strictly implemented. However various mechanisms exist to address this issue, including the SFDU [23] concept, organizing metadata into registries, using file extensions to indicate metadata files, or simply parsing data objects to determine whether they can be interpreted as metadata. The taxonomy of information objects that follows helps address these issues.

2.2 INFORMATION OBJECTS

Information objects (shown in Figure 5) 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 a metadata object: 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*).

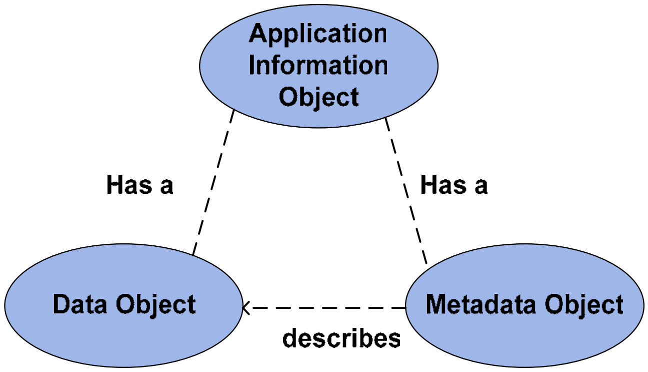

Figure 5. The Model of a Simple Information Object

2.2.1 CARDINALITY

Conceptually, an information object consists of a data object (sequence of bits) and a metadata object (sequence of bits that describes another sequence of bits). However, depending on how a data and metadata object are defined, the relationship is best identified as a one-to-many relationship, typically with one data object and many metadata objects. Logically, for the purposes of conceptually describing the information architecture, this document will show it as a *one* to *one* relationship. In practice, this relationship can be implemented in several different ways. Some projects will define a set of metadata objects for one data object, and another will define a single, aggregate metadata object for one data object. A third might define a set of metadata objects for a set of data objects.

2.2.2 TAXONOMY OF INFORMATION OBJECTS

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

2.2.2.1 Primitive Information Object

A *primitive information object* is an information object with an unstructured metadata type that is not allowed to be compositional in nature and that contains a small amount of metadata with a data object. Unstructured 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.

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. Space data systems have typically focused on the management of primitive information objects, and have not made metadata objects first-class citizens.

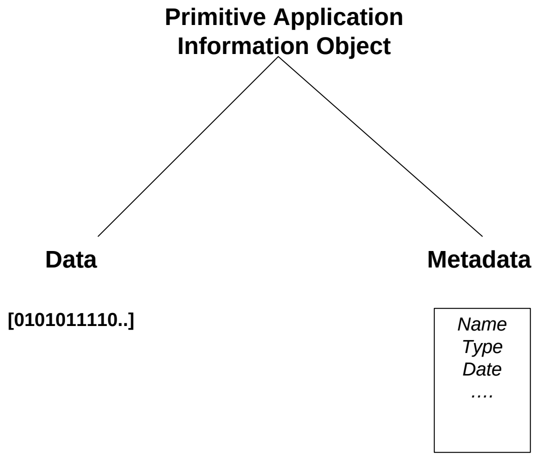

Figure 6. Primitive information object example

2.2.2.2 Simple Information Object

A *simple information object* is defined as an information object that has a structured metadata type defined by one or more domain models. It is not compositional in nature, and contains metadata with a data object. The data object can be null. A number of data systems throughout the space agencies have simple information objects as part of their system design. These have been predominately used within archive and science 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 of data products. Increasing emphasis on constructing end-to-end mission information system architectures will require that simple information objects be used at a variety of stages including observation planning, execution, processing, and distribution across the mission pipeline. Simple 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.

2.2.2.3 Information Package

Information packages (shown in Figure 7) are information objects that encapsulate one or more information objects, coupled with a metadata object containing *packaging information*. Defined in the OAIS reference model [3], 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 OAIS 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 [16], TAR [16] , etc).

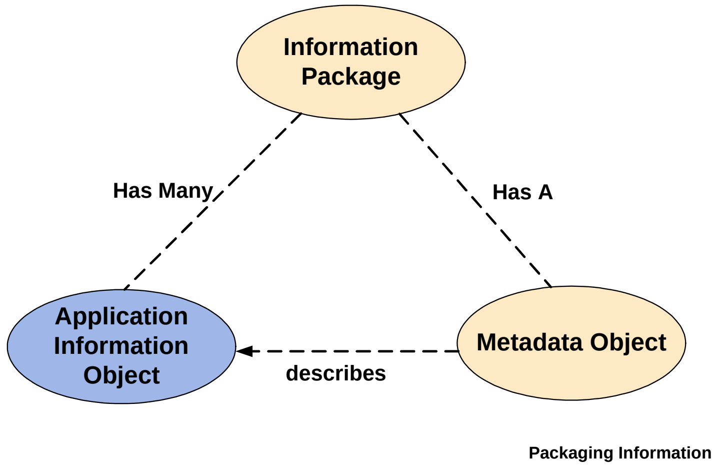

Figure 7. An Information Package

Each information object that makes up the package includes its own metadata object that may or may not correlate and cross-compare with other representation information 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.

The purpose of the information package 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 package (such as *index information* regarding the individual information objects of the package) can be used to deduce package properties. Recent work involving packaging has resulted in the development of a CCSDS Working Group dedicated to studying packaging, and to the development of the proposed CCSDS XFDU packaging model [24] .

Table 1. Information Object View of a Telemetry Uplink Packet

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

2.2.3 INFORMATION OBJECT EXAMPLE

This section gives an example of a space data systems information object using the concepts just discussed. The information object is a telemetry uplink packet sent from a ground station to a spacecraft. The telemetry 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 field, *command sequence*, of type *long integer*. 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 of 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 greater than, or equal to the current time on the sending system. This example is summarized in Table 1.

2.3 MODELING CONCEPTS

Models are important in information architecture because they provide the means to describe objects. Without models, objects cannot be examined, understood, or changed. 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.

The basic relationship between models and the objects they describe are illustrated in Figure 2. 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 models* (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 [22]. Namely, when considering the hierarchy of models illustrated in Figure 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.

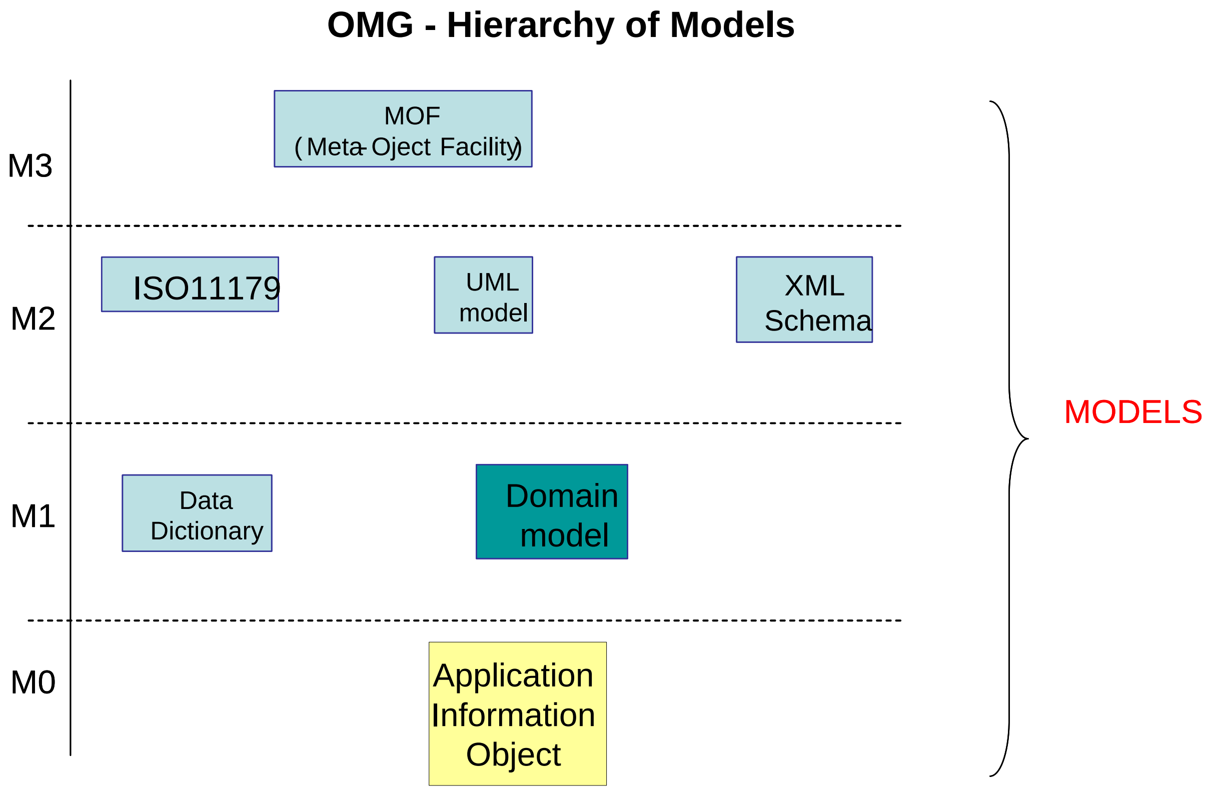

Figure 8. Model Hierarchy, adapted from [22]

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.

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 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.

In the following sections, several standard meta models will be briefly described. These include the ISO/IEC 11179 standard for the specification and standardization of data elements [17], the CCSDS Data Entity Dictionary Specification Language (DEDSL) [5], and the XFDU [24] model for describing information packages.

2.3.1.1 ISO/IEC 11179

In the realm of meta models, the ISO/IEC 11179 [17] 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 namely *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 [5].

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. [36].

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

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 9. The planetary science domain model was defined in PVL/ODL.

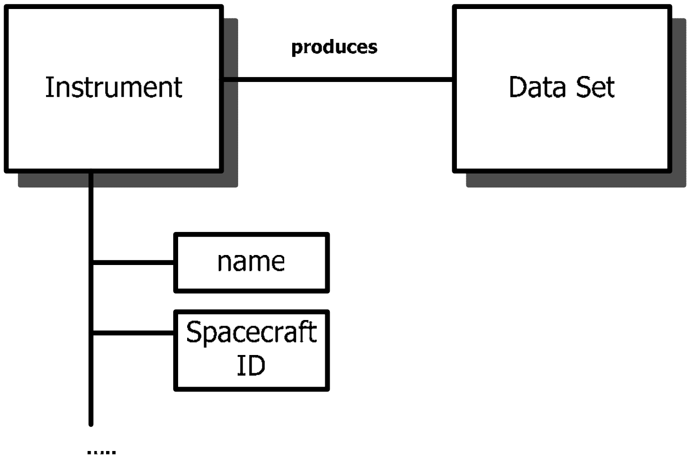

Figure 9. Example Planetary Domain Model (Simplified)

2.3.2.2 SPASE

SPASE [27] is a space and solar physics domain model being developed by an international working group with participation from several NASA 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

The EOSDIS [15] 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.

2.3.2.4 EOS Core Data Model

The ECS Core Data Model [12] was developed as an extension to the earlier EOSDIS domain model in order to specify relationships necessary to handle the sheer data volume (nearly 2 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) [34] is a CCSDS recommendation 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 RDF. The Object Description Language (ODL) is a subset of PVL.

2.3.3.2 EAST

EAST [4] is a CCSDS recommendation for space data system standards. It defines a language and syntax for the expression and exchange of information objects, in the form of *data description records*. 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) [28] is a W3C specification and syntactic format for data objects formatted in XML, which is a subset, or restricted form of the popular SGML language used to exchange hypertext in the early days of the internet.

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 to 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.

The key requirement to enable data system interoperability is having common or at least compatible data elements across the respective domain data models. In Figure 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

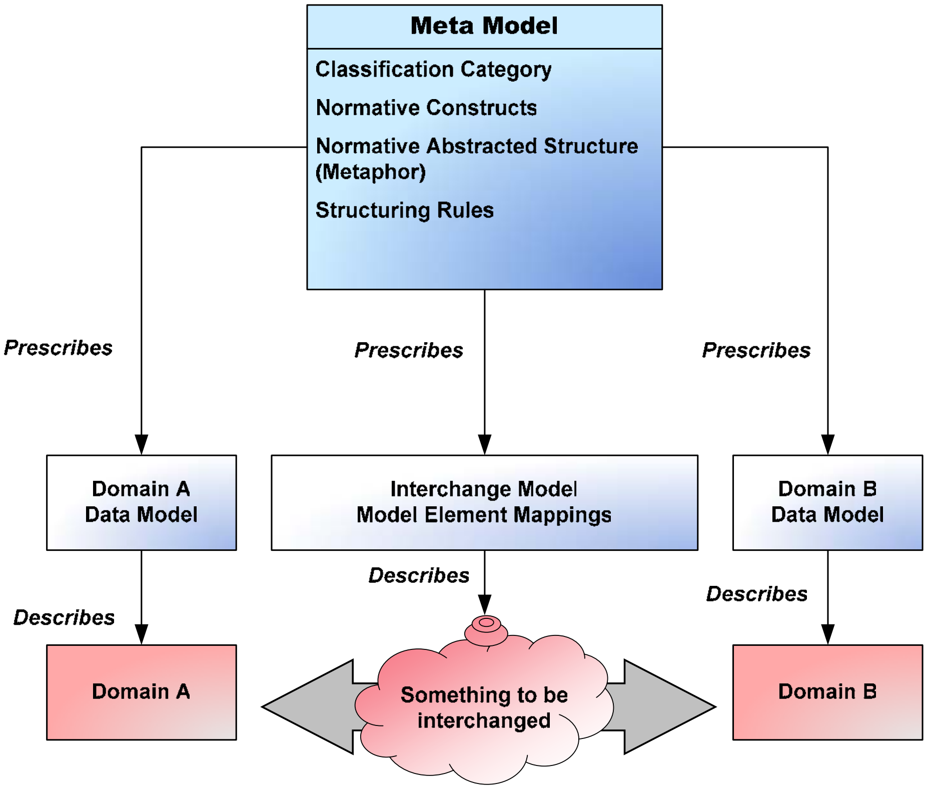

Figure 10. Data Models, Meta Models and Domains

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 [17] 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 chapter 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 various key capabilities of information architecture including *ingestion*, *retrieval*, *processing*, *distribution,* and *querying* of data objects, metadata objects, and information objects.

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

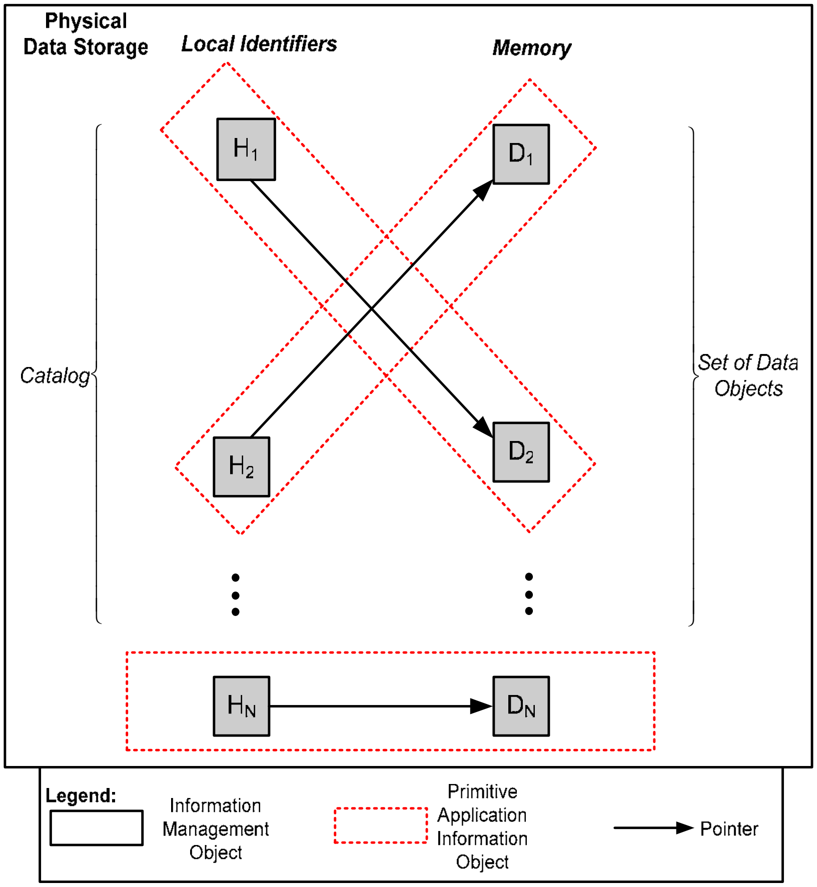

Figure 11. The Internal Structure of a Physical Data Storage

---

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

(components) are called *primitive* since they are assumed to have no recognizable 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:

1. **Memory.** the physical location of the data in the data storage (labeled as “D” in Figure 11)
2. **Local Identifiers.** the index catalog of pointers to memory containing data objects (labeled as “H” in Figure 11)

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 11.

3.1.1 DATA STORE OBJECT

The *data store object* (shown in UML in Figure 12) is attached to an underlying physical data storage and supports *putting* and *getting* information. Figure 13 and Figure 14 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 13) determined by the catalog and ingestion process of the underlying physical data storage. The local identifier is then returned back to the caller.

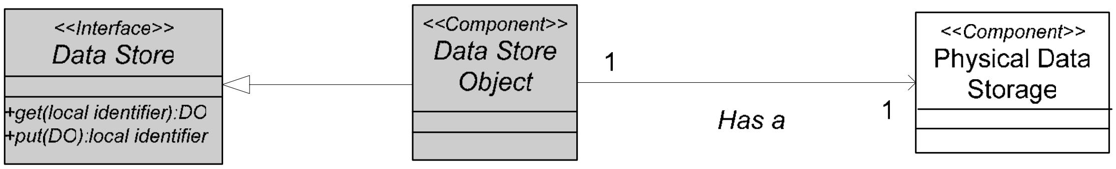

Figure 12. A Data Store Object and its Corresponding UML View

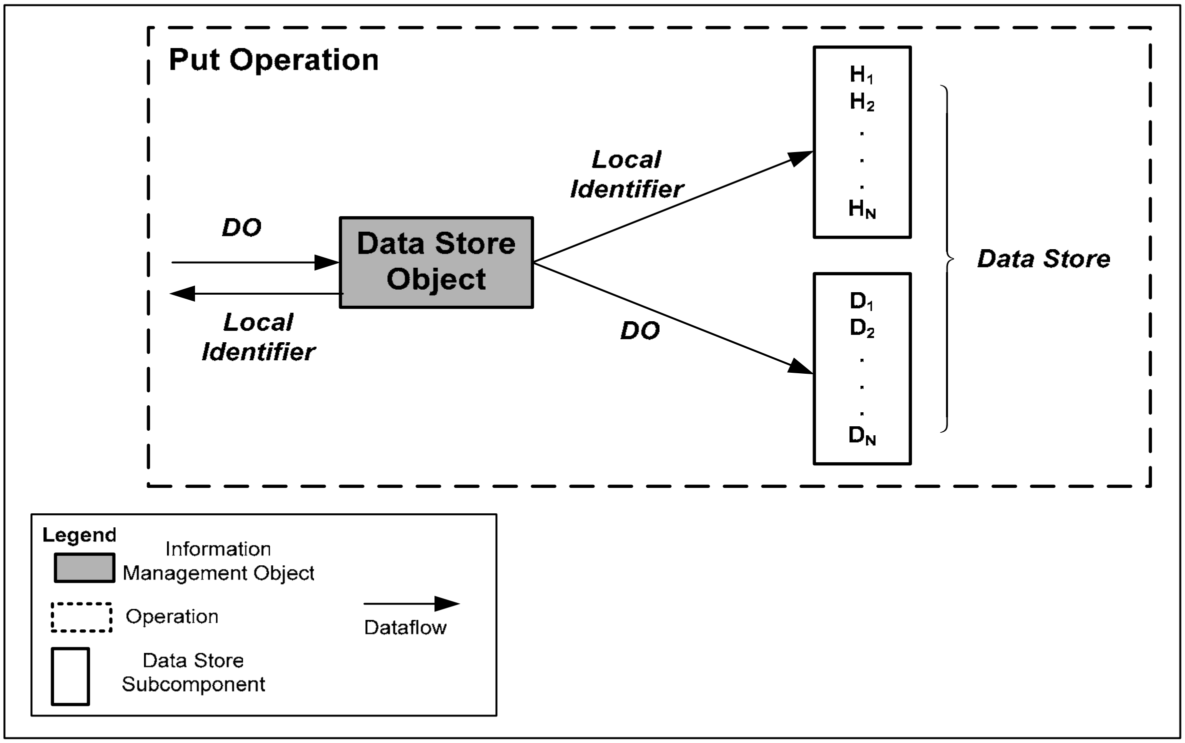

Figure 13. The *Put* Operation of the Data Store Object

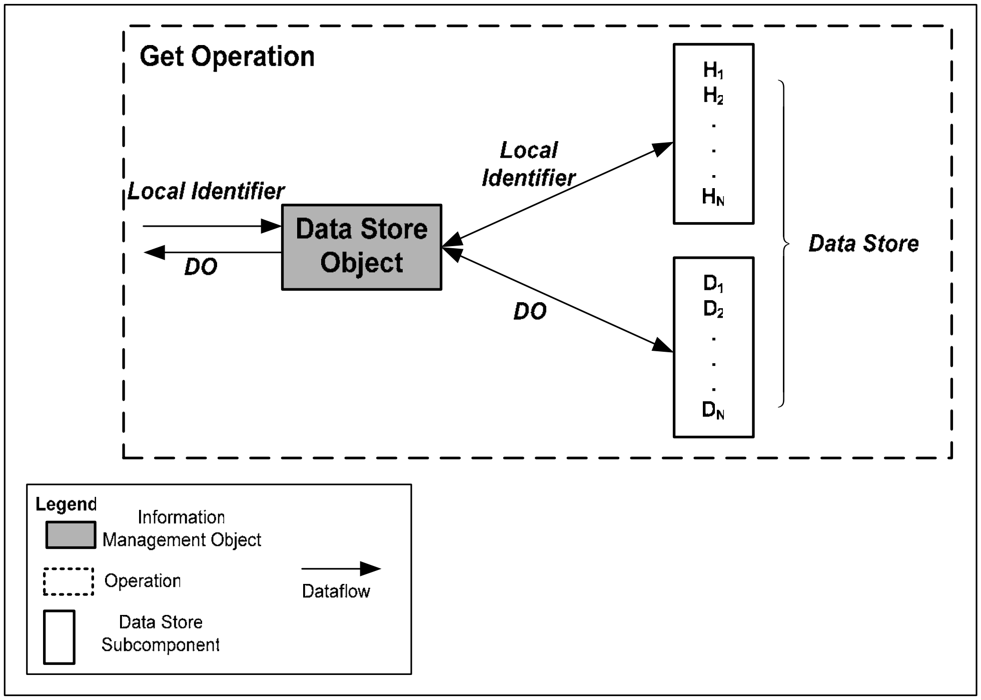

Figure 14. The *Get* Operation of the Data Store Object

3.1.2 QUERY OBJECT

The *query object* shown in Figure 15 enables retrieval of data objects. Data objects (shown as DOs in Figure 15) 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 16 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.

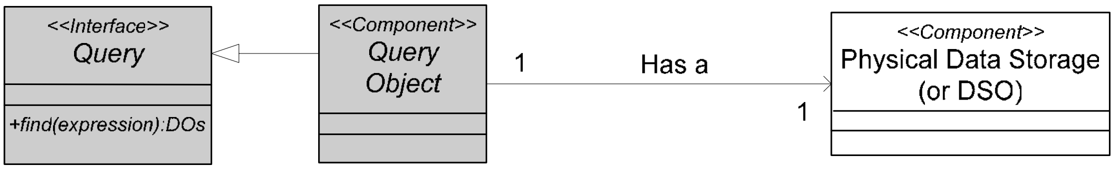

Figure 15. A Query Object and its Corresponding UML View

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

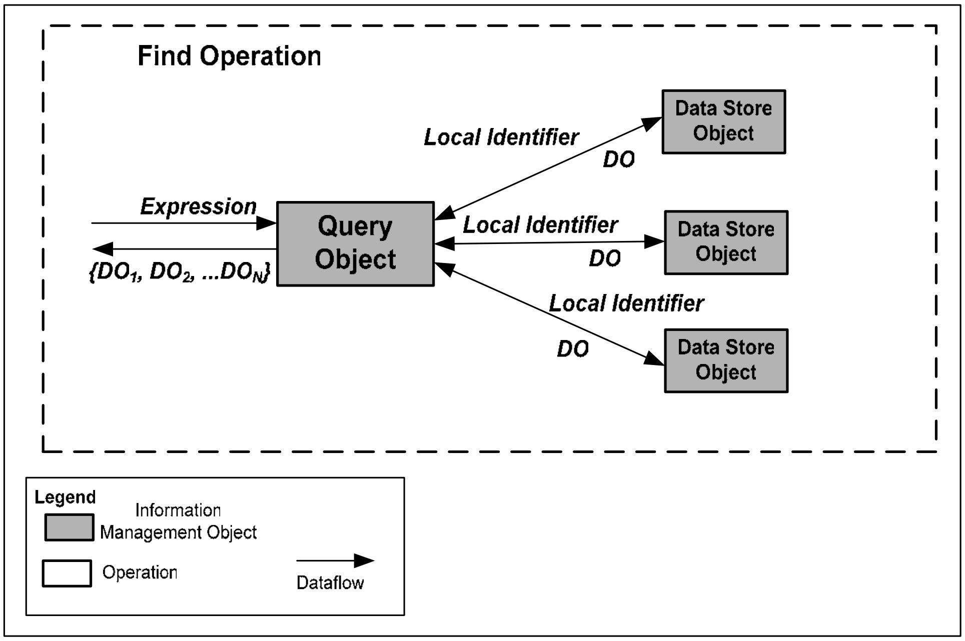

Figure 16. The *Find* Operation of the Query Object

advanced components that span the typical usage scenarios involved in space 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.*

3.2.1 REPOSITORY SERVICE OBJECT

The *repository service object* component is depicted in Figure 17. 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.

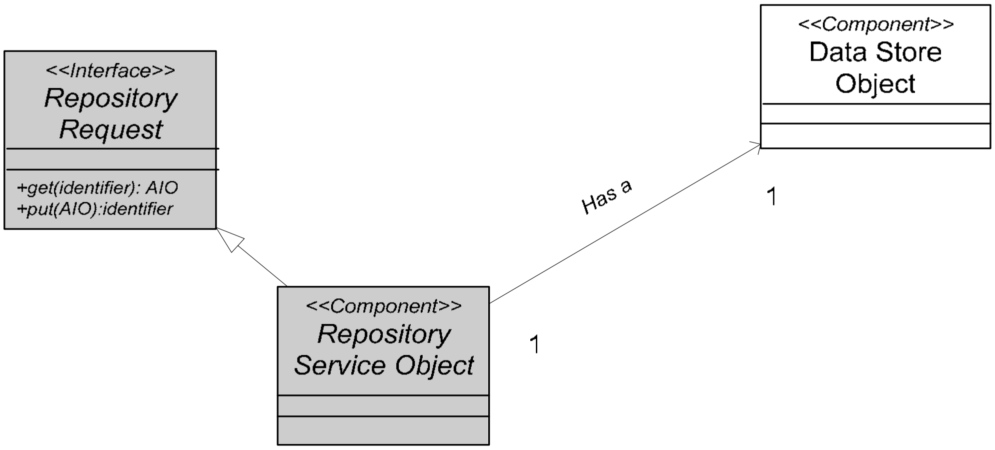

Figure 17. Repository Service Object and its corresponding UML View

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.

Table 2. 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 Section 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).

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 repositories ability to natively handle either primitive get/put

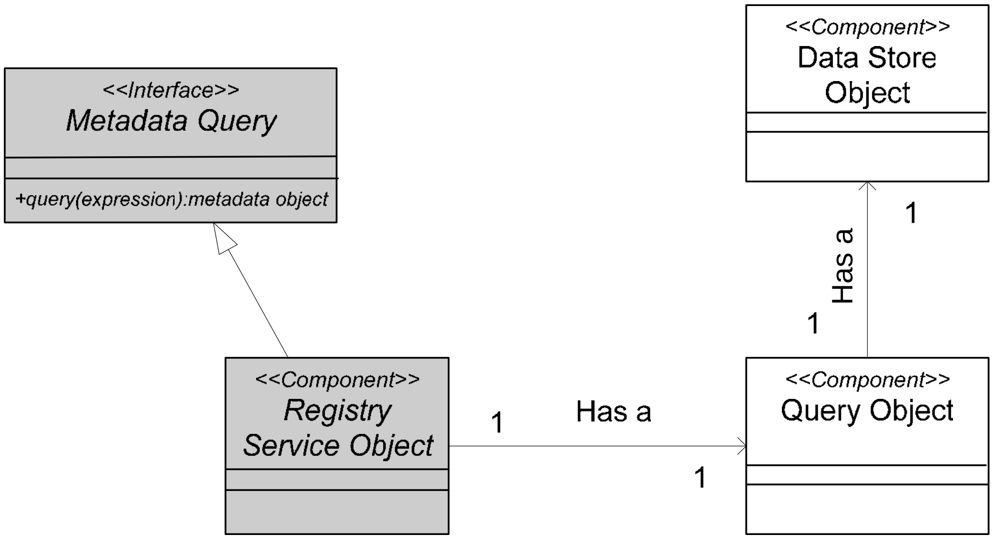

Figure 18. A Registry Service Object and its Corresponding UML View

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 2 lists the current taxonomy and classification of repositories.

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 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 18 depicts 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 are 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 [17] , or explicitly by specifying the data element's ID [17].

Table 3. 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

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 Web Services Description Language (WSDL) [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 [8, 9, 20]. 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 [1] 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 IA Working Group The taxonomy of registry service objects is summarized in Table 3.

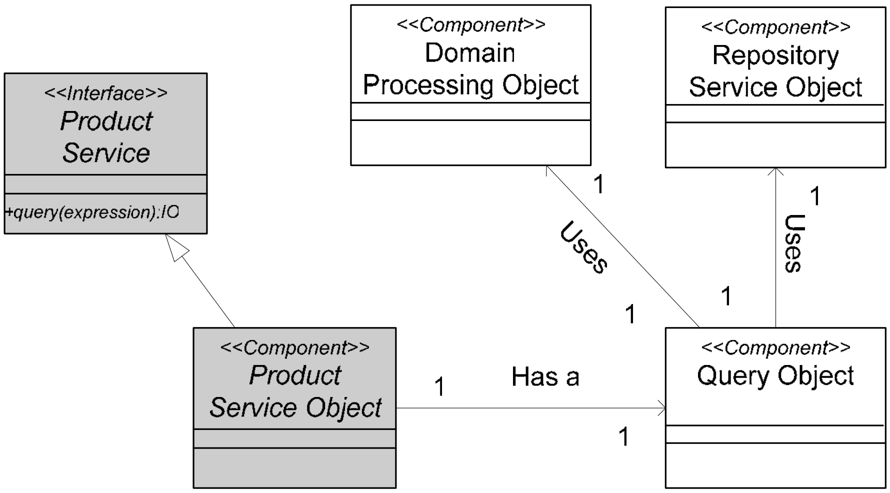

Figure 19. A Product Service Object and its Corresponding UML View

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 domain processing or transformation object. The domain processing object is a functional component that translates data objects from the underlying format used by the data store, to the required format requested by the user. This type of transformation can involve elements such as compression, decompression, scaling (in the case of an image), data type 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 19) 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. A product service object is shown in Figure 19.

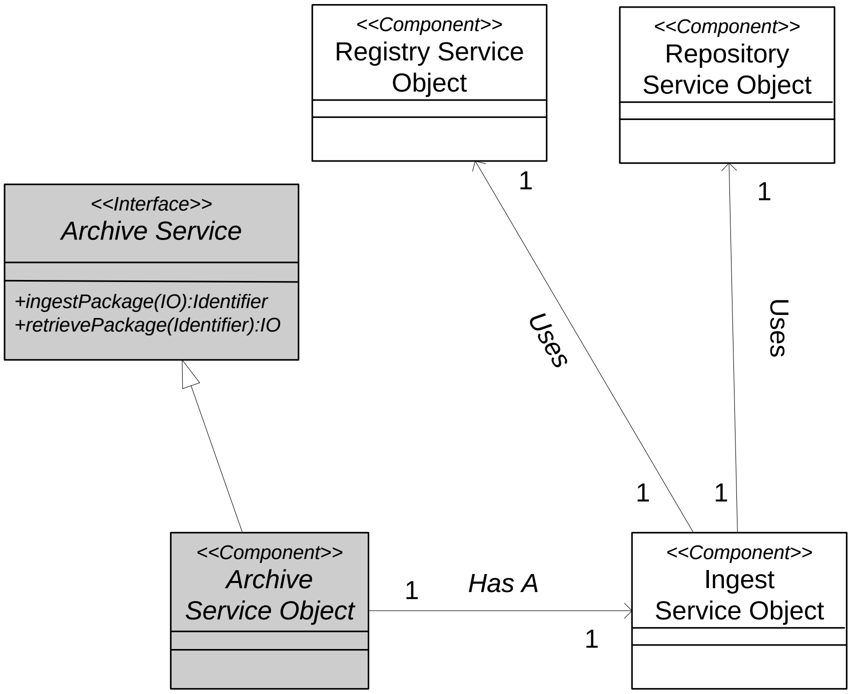

Figure 20. An Archive Service Object and its Corresponding UML View

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 20). 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* [2, 10, 11]. This type of ingestion process is shown as the *ingest service object* component in Figure 20. 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 [3]. This type of transaction capability would be provided by the ingest service object in Figure 20. An archive service object is shown in Figure 20.

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 21) 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 21. Figure 21 depicts a query service object.

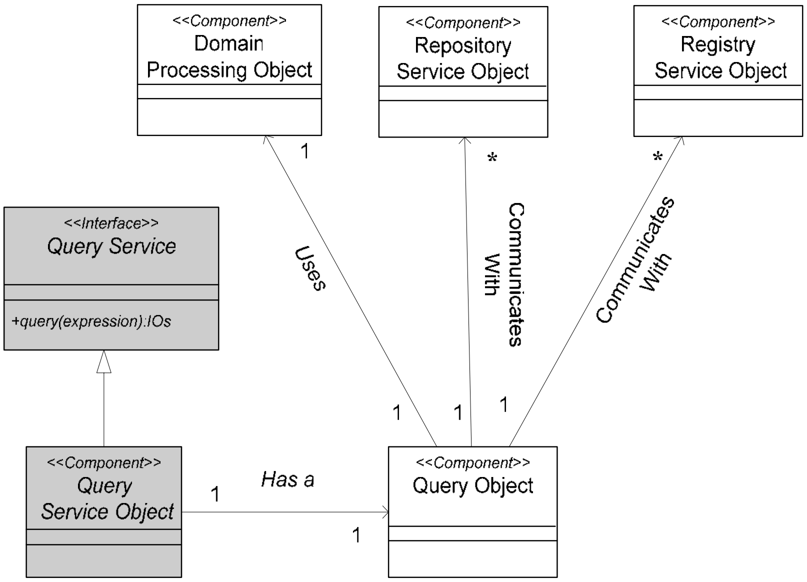

Figure 21. A Query Service Object and its Corresponding UML View

4 VIEWPOINTS FOR INFORMATION ARCHITECTURE

Information Architecture can be grouped into different categories, particularly in the context of space data systems. These categories are often referred to as *views* and carry much of the notion that views carry within the realm of software architecture. Views help to disseminate the same information to different stakeholders, who have different perspectives of the system. The two views of information architecture of particular interest in space data systems are the *information view*, which is concerned with data and its structure, and the *functional view*, which is concerned with supporting the locating, searching, and retrieval of data.

Section 4.1 discusses the information view with respect to the topics introduced in Section 2. Section 4.2 discusses the functional view with respect to the topics introduced in Section 3.

4.1 INFORMATION VIEW

Figure 22 shows the information view with respect to the other views involved in space data systems. This stack of views is organized from top to bottom, with the bottom view being the most related to implementation issues of space data systems, and the top view being the most abstract, and concerned with issues of the space organization (such as NASA, ESA, etc.). The information view is more abstract than the communications view, but is more related to implementation level issues than the functional view.

The concerns associated with the information view in information architecture are that of data, metadata (in the form of structure, semantics, relationships and security) and the representation of data (in forms such as data objects). These concerns are discussed extensively in Section 2.

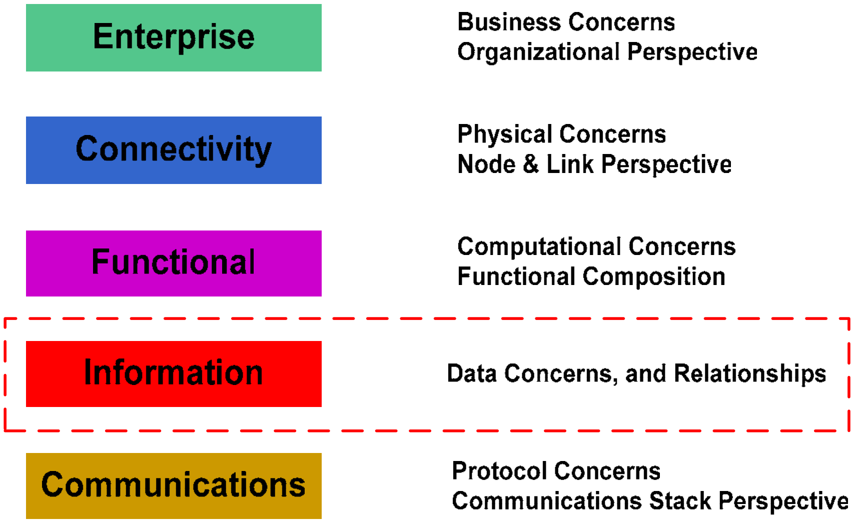

Figure 22. Information View in Perspective

4.2 FUNCTIONAL VIEW

The functional view of information architecture is concerned with supporting the capture, discovery, search and retrieval of information via functional components which implement the aforementioned capabilities. Section 3 discusses these functional components with respect to their software implementations and can be consulted for further detail. This work is an elaboration of the information view and a treatment of the information management objects described in the RMODP and RASDS modeling notations.

5 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 the following table:

Table 4. Information Architecture Usage in Space Data Systems

Project	Information Architecture Concepts Used
OAIS	information objects, information packages, archive service object
SPACEGRID	information objects, registry service object
EOSDIS	Meta models, domain models, metadata objects, information objects
European Data Grid	information objects, information packages, archive service object, registry service object
National Virtual Observatory	information objects, information packages, archive service object, registry service object

5.1 OAIS

The CCSDS OAIS reference model [3] 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,

1. producers produce submission information packages (or SIPs) to send to the OAIS compliant archive.
2. consumers consume dissemination information packages (or DIPs) that they retrieve from the OAIS compliant archive
3. 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 (or 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 [3].

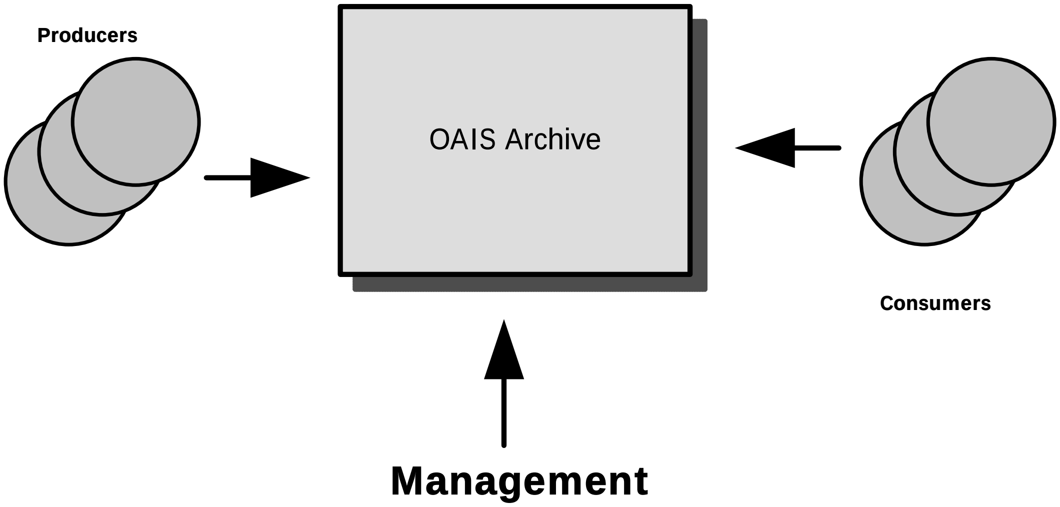

Figure 23. The Open Archival Information System Reference Model

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 Section 3.2.4; 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.

5.2 GRIDS

Recent work in the *grid* community [14] has characterized a class of distributed data interoperable systems as *data grids* [6, 7, 21, 26]. Data grids involve the identification of metadata, and different classes of metadata [26] which is required to make heterogeneous software systems interoperable. In the next paragraphs, some overviews of grid projects at various space agencies are listed. Each section details how each grid project uses the components of information architecture.

5.2.1 SPACEGRID

ESA's Space Grid Study [19] 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 3 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:

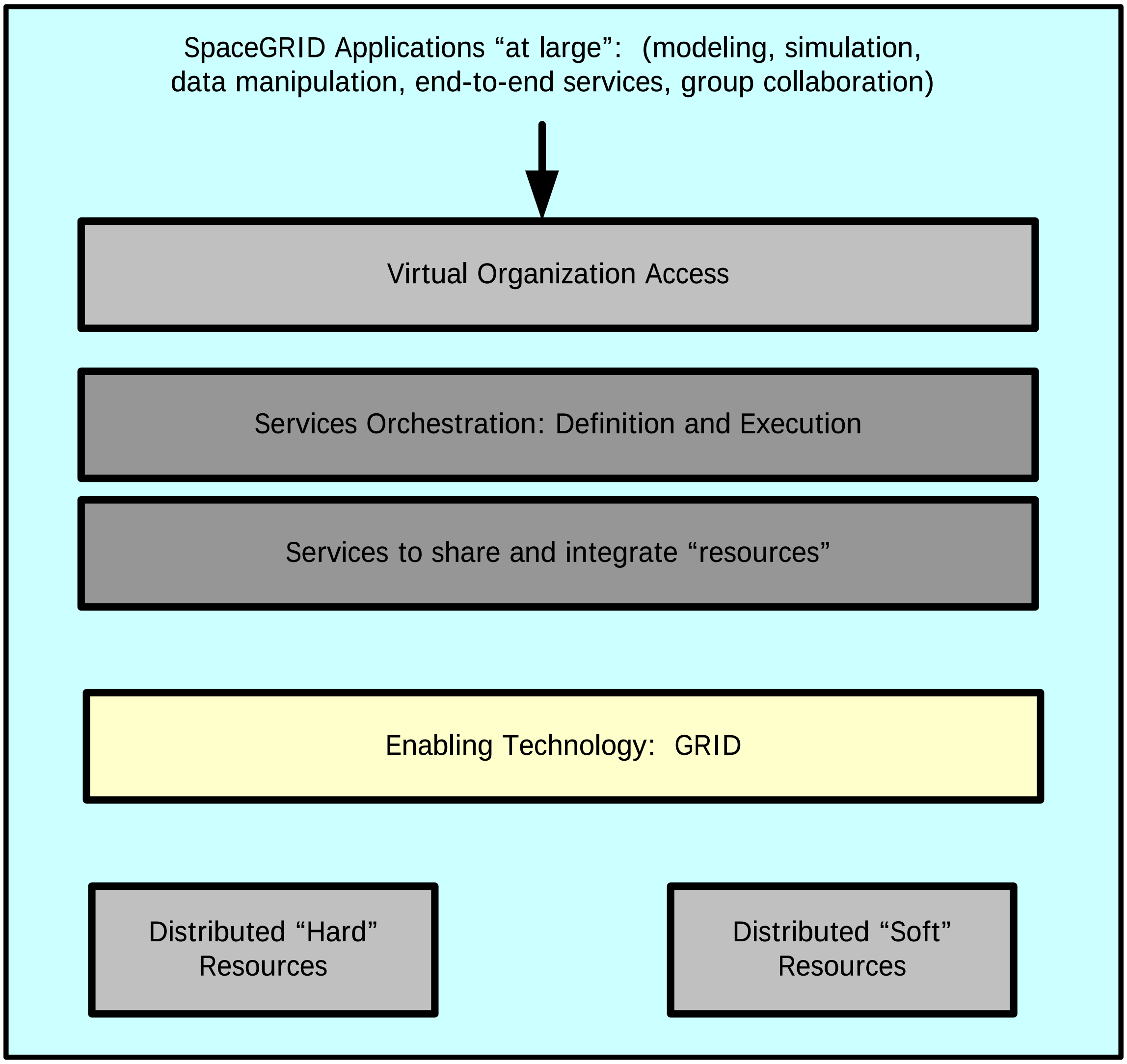
SpaceGRID Applications "at large": (modeling, simulation, data manipulation, end-to-end services, group collaboration)
Virtual Organization Access
Services Orchestration: Definition and Execution
Services to share and integrate "resources"
Enabling Technology: GRID
Distributed "Hard" Resources
Distributed "Soft" Resources

Figure 24. SpaceGRID proposed infrastructure

1. *Flexibility*
2. *Portal*
3. *Security*
4. *Distributed Access*
5. *Human Computer Interface*
6. *Virtual Organization*
7. *Collaborative Environment*
8. *Reliability*

Figure 24 depicts the proposed SpaceGRID infrastructure, which is very similar 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.

5.2.2 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 save sending data on removable media and taking large amounts of time 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.

5.2.3 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 [13]. 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.

5.2.4 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 [14, 18] 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.

6 APPENDIX I

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

Table 5. 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