# Consultative Committee for Space Data Systems

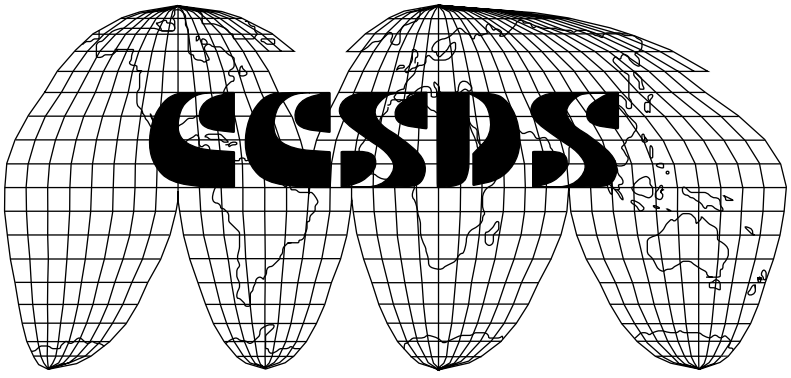
**DRAFT RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS**

# INFORMATION ARCHITECTURE FOR SPACE DATA SYSTEMS

**CCSDS *number***

**WHITE BOOK**

August 2004



# AUTHORITY

Issue:

Date:

Location:

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

## STATEMENT OF INTENT

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

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

– Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

– Whenever an Agency establishes a CCSDS-related standard, the Agency will provide other CCSDS member Agencies with the following information:

  • The **standard** itself.

  • The anticipated date of initial operational capability.

  • The anticipated duration of operational service.

– Specific service arrangements are made via memoranda of agreement. Neither this **Recommendation** nor any ensuing standard is a substitute for a memorandum of agreement.

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

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

# 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 |

# CONTENTS

# FIGURES

Figure Page

# 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 across ground and flight systems and across agency data systems. The focus of this document is to present a Standard Reference Space Information 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 standard functional components for information management, definition of standard interfaces for information management, standards in information representation (data structuring and packaging mechanisms), and standard definitions of information processes (interactions between users and systems).

Another goal of this document is to define how existing standards fit into an overall reference information architecture. The reference information architecture should encompass informatics aspects of space data systems and cover all 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 for the Reference Architecture for Space Data Systems (RASDS) [1] being developed by the CCSDS architecture working group.

This document also introduces a layered view of the information architecture. To achieve interoperability both within and across domains, and across applications built based on this standard, conformance at each layer must be achieved. Figure 1 depicts this view and represents possible means of achieving interoperability at each layer. Each level is critical

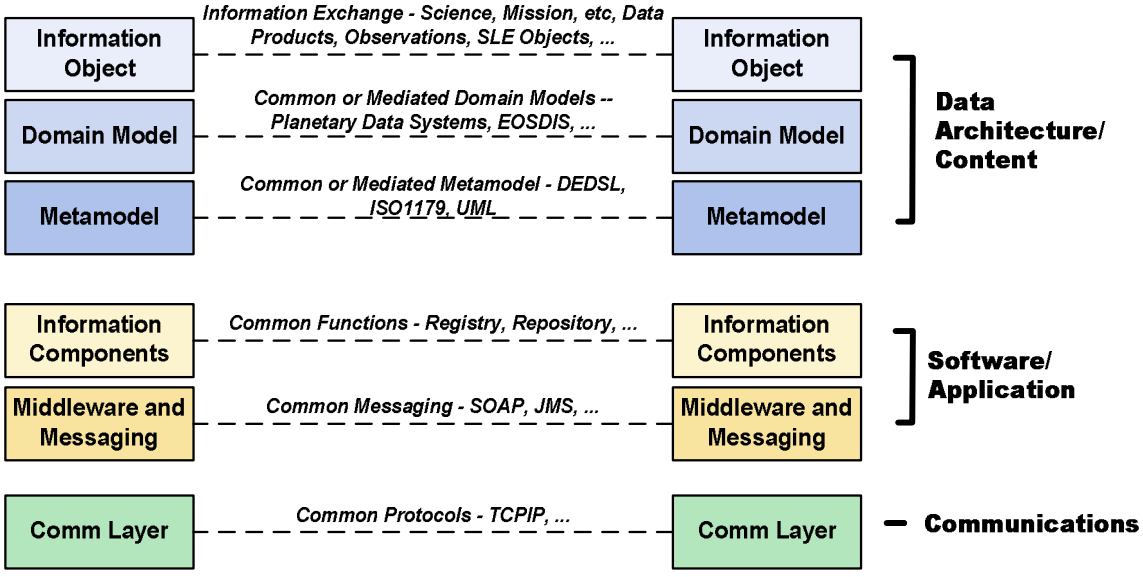



**Figure 1. High-level view of interoperable information architecture**

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

This document is intended for those interested in using and developing standard information architectural elements for building Space Data Systems. These elements include standard software components, such as registries, and repositories, and standard data components and descriptors such as profiles and resources. 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.

# 2 INFORMATION ARCHITECTURE

The driving force behind any standards document is the ability to uniformly prescribe a standard functional protocol, method, or model by which all those who conform will receive promised properties, benefits and consequences. In this standard, that model is the *Information Object*. It is the cornerstone of Information Architecture, through which data is described, transferred, and retrieved, in a standard fashion.

This section is driven by Figure 2. The information object is composed of: the *data object*, a sequence of bits responsible for physically representing data; and the *metadata object*, an additional sequence of bits which defines (a) the *type* or *classification* of data stored in the data object and (b) any *additional* data describing the sequence of bits represented by the data object. A small taxonomy of information object types commonly used in Information Architecture follows. A simple example in the space data systems domain to clarify the discussion on Information Objects is also presented. The section continues with a definition of *meta models, domain models, and data dictionaries,* which play a key role in the description of information objects. The section concludes with a discussion on related work in information representations including *Data Grids*, *CCSDS Standards,* and related projects.

## 2.1 DATA ARCHITECTURE

Architecture [2-5] is referred to as both the process and the outcome of reasoning and specifying the overall structure of a system, its logical components, and their logical interrelationships. *Data architecture* is the application of this concept to the data components of an Information Architecture.

### 2.1.1 DATA OBJECTS

*Data objects* constitute data as it is physically represented using a sequence of bits. Although data objects may exist without metadata objects, as mentioned above, without metadata objects, the utility of a data object significantly decreases, since not even the data structure is known.

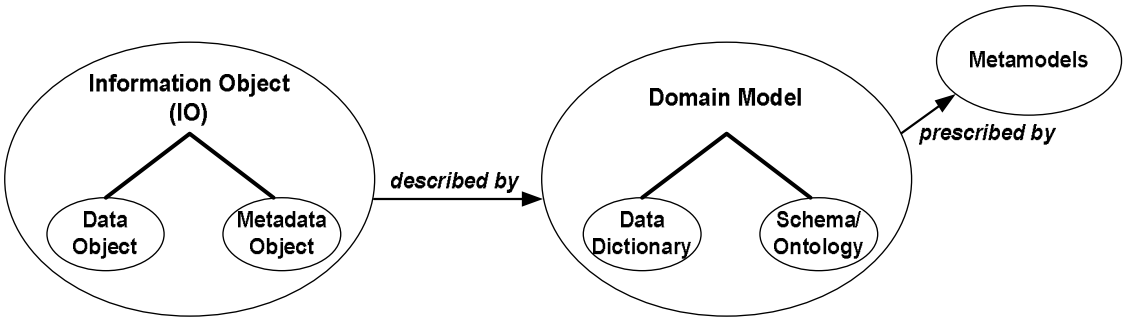



**Figure 2. Information Architecture - the Big Picture**

### 2.1.2 METADATA OBJECTS

*Metadata objects* are a special class of data objects (or bits) that describes the structure, (syntactic) validity, interrelationships, and (semantic) rules of a data object and its respective structural elements. Metadata objects may also contain *classification information* describing what type of data or resource is stored in the data object. A practical example of such a classification is the *Dublin Core* model [6] that may be used to describe any electronic resource.

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. This is seldom strictly implemented. However various mechanisms exist to address this issue, including the SFDU [7] 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.

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; users may not be able to view it. For example, consider that the data object is a zipped file, and consequently appears to the user when viewed (using a text editor) as a sequence of out of order ASCII characters rather than English language. On the other hand, 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 by the metadata object) will reveal the value of the pixel. Moreover, if the metadata object describes a spacecraft, the spacecraft Information Object can be related to the instrument Information Object that collected the image and used to support correlative science across spacecraft and instrument. Again, the actual spacecraft certainly does not exist as a data object, however a digitized image of the spacecraft could be referenced in the metadata object as part of the description of the spacecraft. This relationship is the basis of the information architecture described in this document.

## 2.2 INFORMATION OBJECTS

Information Objects are data 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 describes the data object's *structure*, such as what fields (e.g. in the space data domain, *Uplink Speed* or *Downlink Capacity* can be candidate 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

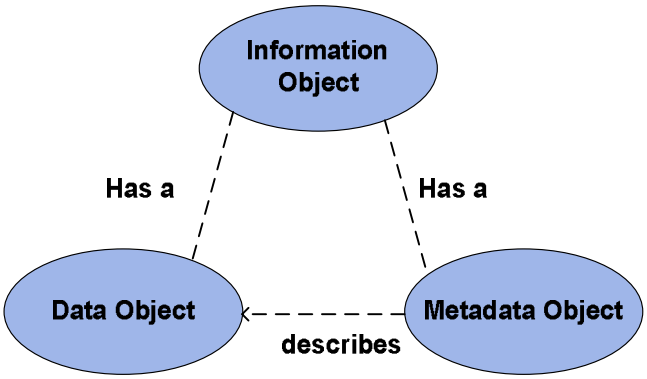



**Figure 3. The Model of a Simple Information Object**

such as 1MB or 2MB/sec), and the *semantic relationships* between the structural elements (such as *Uplink Speed* ***must always equal*** *Downlink Speed*).

The Information Object, including its subcomponents and relationships, is modeled in Figure 3.

### 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). Practically, however, an information object may be implemented in a number of ways. For example,

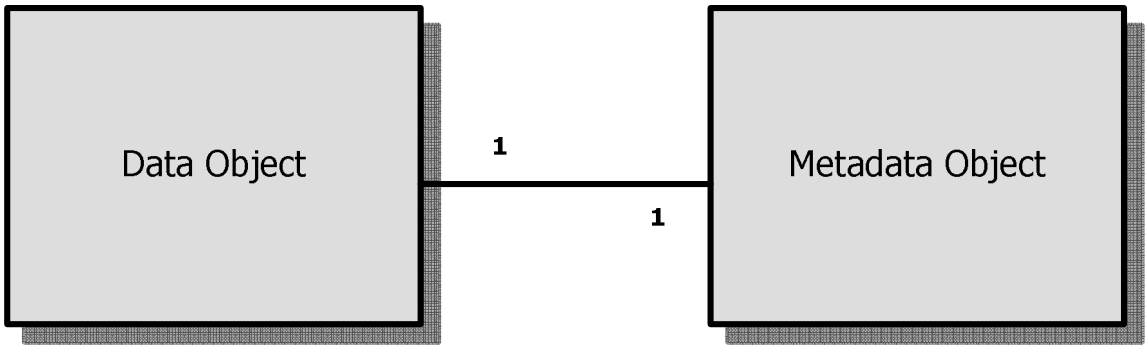



**Figure 4. NASA Planetary Data System cardinality restriction on information objects**

in NASA's Planetary Data System, each data object (such as a raw raster image or its histogram) must be accompanied by *exactly one* metadata object.

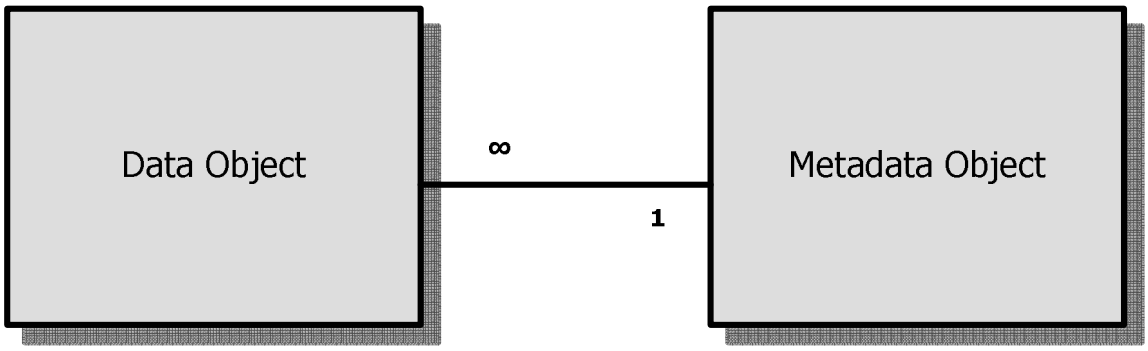



**Figure 5. Other Information Object cardinality relationships**

Sometimes in practice though, an information object may be implemented as a series of data objects, coupled with a single metadata object. In the view of the NASA Planetary Data System, the fact that there are a series of data object as opposed to just one is an implementation specific issue because the series of data objects, since they are just bits, could be considered as one single data object. Thus, the concept of one data object per one metadata object holds.

It is important to distinguish that these two scenarios are exactly the same. They differ only across the boundaries of *practical implementation* versus *conceptual views*. Even though both scenarios depict information objects with seemingly different cardinalities and relationships of data objects and metadata objects, conceptually, in the end, an information object can be viewed as a 1-to-1 relationship between a metadata object and a data object.

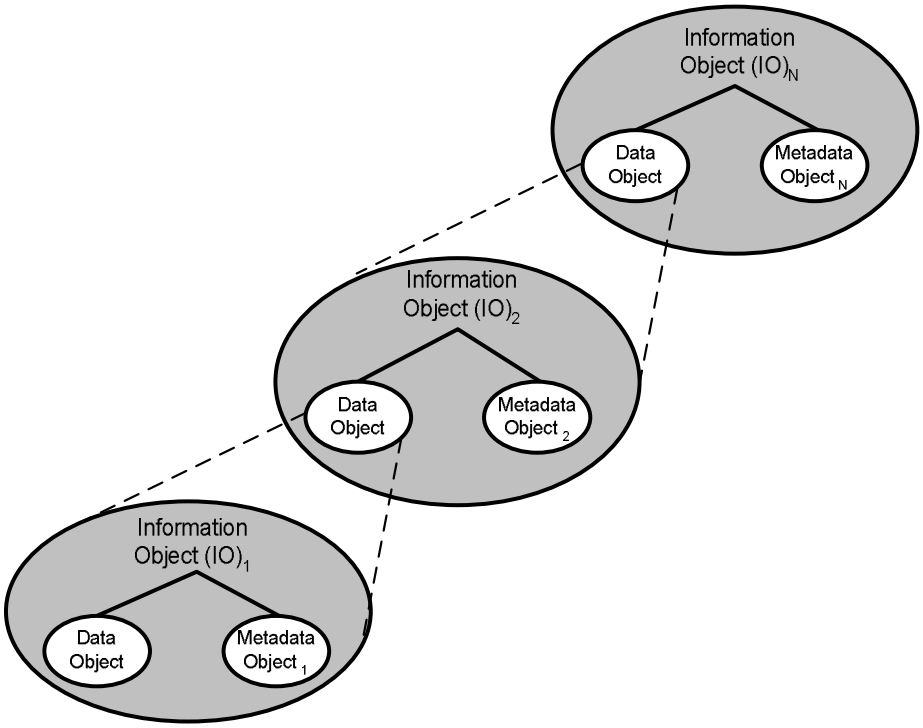



**Figure 6. Compositionality of an Information Object**

### 2.2.2 COMPOSITIONALITY

It is important to identify that data objects themselves may be information objects. A data object is a sequence of bits. An information object may also be viewed as a sequence of bits, and thus, information objects may be composed of other information objects.

The difference between this scenario and an *information package*, discussed in Section 2.2.3.3, is the metadata object. Also, it is true that a data object *may be an information object*, but an information object *is not a data object*.

In a compositional information object (i.e. an information object whose data object is actually an information object itself and so on recursively), each metadata object only describes its associated data object (e.g. the one-to-one cardinality still holds). In an information package, this also holds, but there is an additional metadata object which describes the aggregate of data objects, or the package's data contents as a whole.

### 2.2.3 TAXONOMY OF INFORMATION OBJECTS

So far, the discussion on Information Objects has been very broad in scope. Consequently, this standard is considered to be applicable to Information Objects of various *classes*. We differentiate among these classes along the following three dimensions:

- *metadata type* – defining the type of metadata (e.g. accounting, structure, inter-relationships, etc.) in the metadata object
- *composition* – defining whether the Information Object's data object is actually an Information Object in itself (which this standard does not preclude from happening)
- *class* – defining the type of data this information object contains and describes.

In this section, we present a preliminary classification of three different Information Objects along these three dimensions.

#### 2.2.3.1 Primitive Information Object

We define a *primitive information object* to be an information object with a *minimal* metadata type that is *not allowed* to be compositional in nature and that can contain *any* class of data. Minimal metadata indicates that the only metadata captured for a particular data object are primitive elements such as its size, format, etc, but it most likely will not identify other characteristics of the data object.

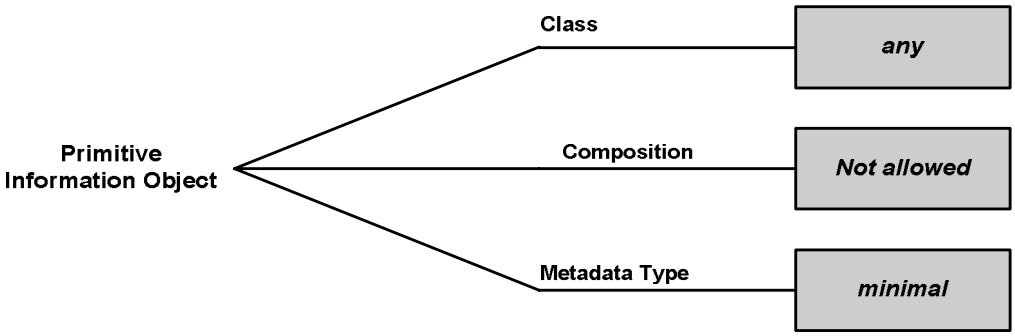



**Figure 7. Classification of Primitive Information Object with Information Object Taxonomy**

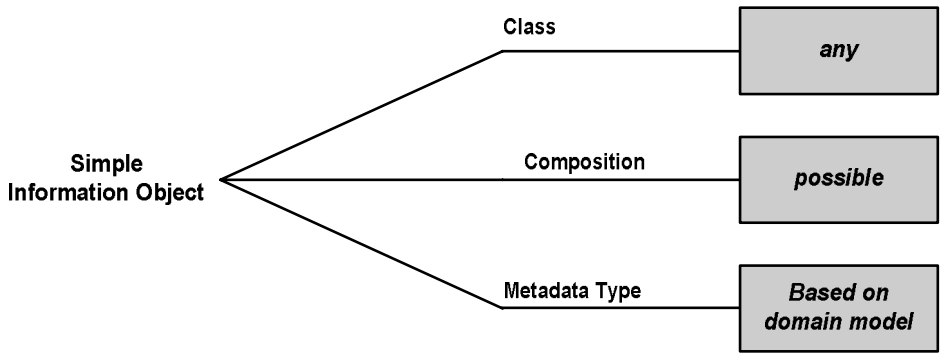



**Figure 8. Classification of Simple Information Object with Information Object Taxonomy**

An example of a primitive information object is a data file managed in a solid state recorder. Minimal metadata exists for it in the form of the address space that it occupies, and perhaps the name of the file.

Space data systems have typically focused on the management of primitive information objects, and have not made metadata objects first-class citizens. Figure 7 demonstrates the classification of primitive information objects.

#### 2.2.3.2 Simple Information Object

A *Simple Information Object* is defined as an Information Object that has a metadata type *defined by a domain model*. It could *possibly* be compositional in nature, and could contain *any* class of data. 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 the archive and science data systems. The metadata for these information objects are often stored in an online registry or database and made accessible 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. Figure 8 demonstrates the classification of primitive information objects.

#### 2.2.3.3 Information Package

*Information Packages* are collections of one or more Information Objects, coupled with a metadata object containing *Descriptive Information*, *Packaging Information*, and *Supporting Information* regarding the package itself. Defined in the OAIS reference

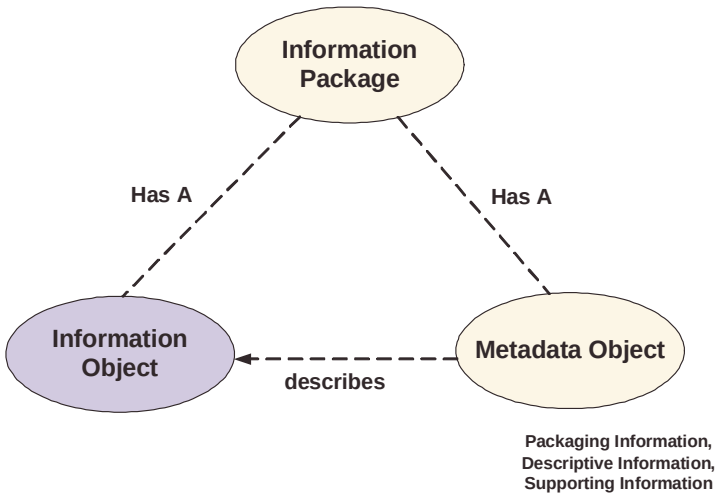



**Figure 9. An Information Package**

model [8], descriptive 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. Also defined by OAIS, 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 [9], TAR [9] , etc). Supporting information includes any representational information needed to understand the data. The information package is shown in Figure 9.

Tying back to the taxonomy of information objects, information packages can be classified as Information Objects with *package* class. The metadata type could be specified as *supporting, descriptive* and *packaging,* with a *required* compositionality. Figure 10 depicts this classification.

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 (meta models are described in Section 2.3).

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

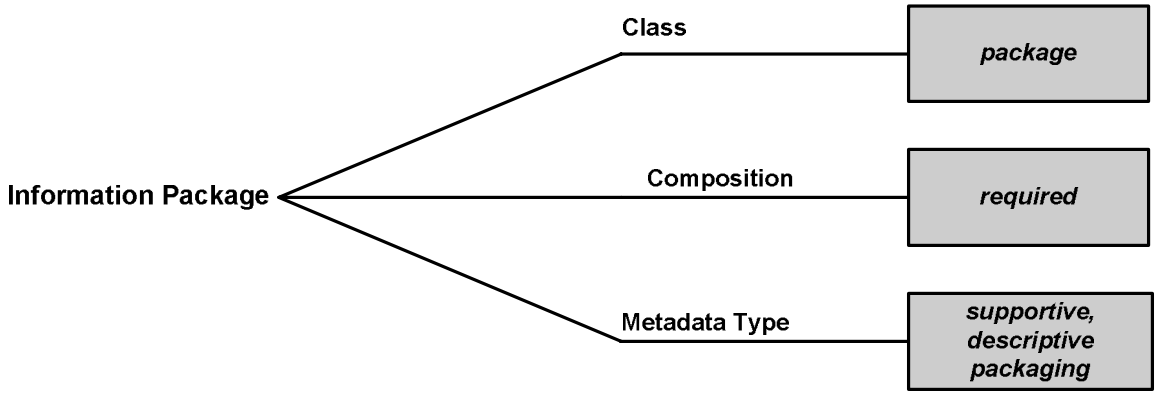



**Figure 10. Classification of Information Package with Information Object Taxonomy**

individual Information Objects of the package) can be used to deduce package properties.

**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 \| a ε { spectrometer, hi-resolution imager} |

Recent work involving packaging has resulted in the development of a CCSDS Working Group dedicated to studying packaging, and the development of the XFDU packaging standard [10] .

### 2.2.4 DISCUSSION

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 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 META MODELS

As a data model is used to describe a domain, a meta model is used to describe a data model. A meta model is important for generating consistent metadata objects. They comprise a set of data elements which are used to capture metadata in an information object. Data elements are granules of information which describe a particular facet of a data object. For instance, an example data element for a book data object would be *Title*.

In information architecture, meta models are important because they define the structure of metadata objects. Given this structural identification metadata objects can be

examined, compared, changed, and integrated if needed. These capabilities are critical because they facilitate the correlative use and exchange of data in space data systems. Due to the heterogeneous nature of data that must be exchanged between various space data system components, the ability to examine, compare and change metadata belonging to a particular information object is important. It serves as a direct enabler of services such as: (1) information object accountability, (2) information package generation and (3) information object delivery [11]. These services are critical because they allow scientists to discover the location and format of information objects at their precise stage in the space-to-ground pipeline. This has the potential to allow for earlier discoveries, earlier detection of faults and ultimately facilitates a larger understanding of where the data is in the space data system. Further meta models are one of the principle keys to data system interoperability because they facilitate the ability to actually compare data between two or more systems. Without a shared meta model, it is impossible to reuse and exchange information. This is not only true of information objects from space data systems, but true of engineering, software and other types of models and objects.

Practical examples of meta-models include the Dublin Core meta-model for describing electronic resources [6] and the XFDU [10] meta model for describing information packages.

In addition to meta-models, we also note the existence of *meta-meta* models for clarity. Meta-meta models describe the structure of the data elements that comprise data dictionaries. In information architecture, meta-meta models are important because they prescribe how data elements themselves can be compared and examined across dictionaries. If data elements did not conform to a particular meta-meta model, then it would be impossible to guarantee the ability to compare and examine the elements even within the same dictionary. Since the ability to compare elements is critical to enabling interoperability of data exchanged between systems, it is necessary that meta-meta models describe the underlying structure of the data elements themselves.

Practical examples of Meta-models include the ISO-11179 standard for the specification and standardization of data elements [12], along with the CCSDS Data Entity Dictionary Specification Language (DEDSL) [13].

## 2.4 DOMAIN MODELS

Domain models also help to facilitate correlative use and exchange of data. Domain models are defined structurally and semantically by meta models. A domain model is an instance of a meta model for a particular modeling domain. For instance, the NASA Planetary Data System domain model (shown in Figure 11) defines objects such as *instruments* and *data sets*. Further, it defines attributes of the objects, such as “an instrument has a spacecraft id, and an instrument name”. Lastly, the domain model defines the relationships between objects of a particular domain. Using the NASA Planetary Data System example, the relationship “instruments *produce* data sets” would be captured in a domain model.

With respect to domain models, *Ontologies* are examples of domain models that are used to define concepts and their relationships within a domain. An earth science ontology, for example, would be used as a mechanism for building the earth science data system

information architecture by identifying the major objects, their attributes and their relationships. This would then enable the construction of a data dictionary along with associated XML schemas that support the definition of information products useful in the capture and exchange of data within and between systems

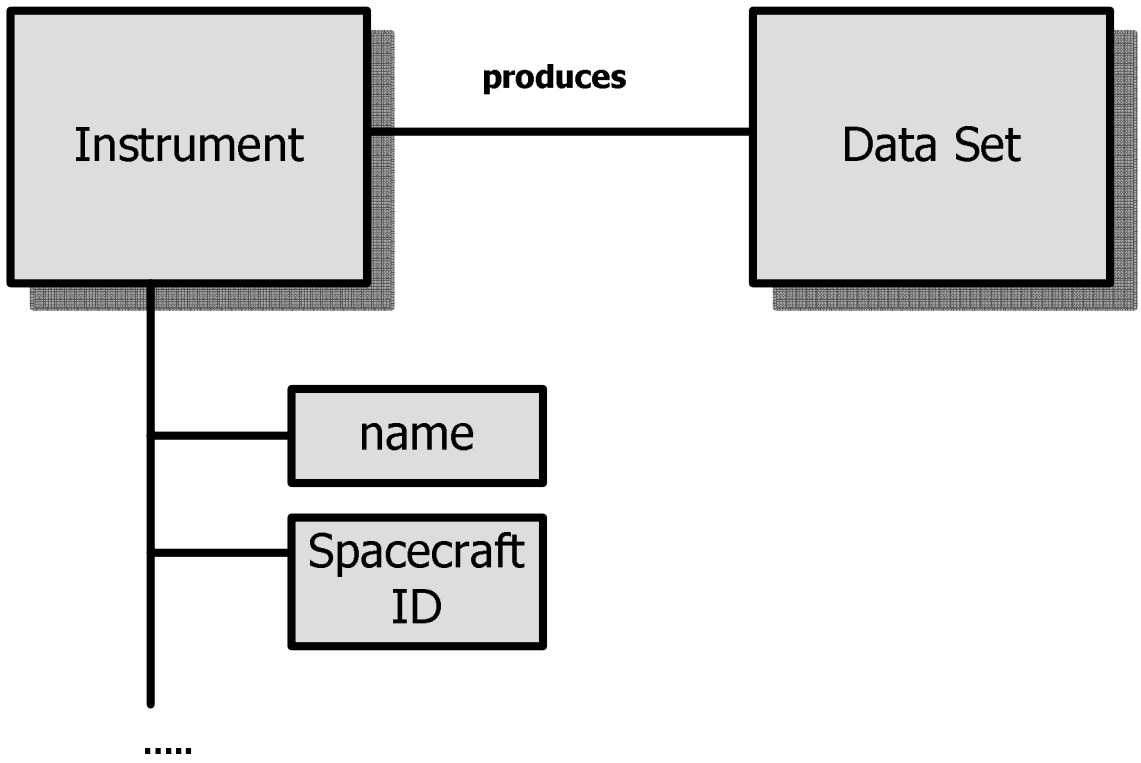



**Figure 11. Example Planetary Domain Model (Simplified)**

## 2.5 DATA DICTIONARY

A Data Dictionary captures the superset of de facto data elements for a particular domain, along with any semantic constraints (such as validation) and additional metadata per data element in the superset captured. Data dictionaries are useful for capturing definitions of data elements so that definitions and constraints can be re-used across disciplines. A practical example of this idea is the data element *Mass*. In space science, mass is defined as "the property of a body that causes it to have weight in a gravitational field". The planetary science domain shares this definition of mass as well, along with the earth science domain. If the definition and constraints (e.g. the information captured by the data dictionary) of mass is stored once, it should not be replicated unless its definition and/or constraints change. Seldom is this reuse principle seen in practice, however. In fact, it is not uncommon to see definitions, data elements, and constraints replicated across and within many different scientific domains, including space science. A consensus on the definition and constraints for a data element allow a software program to reason about the "real" meaning of a facet of data. This type of semantic reasoning is

critical in embedded software such as space data systems because it removes an element of the human-in-the-loop and allows software to perform a task that is typically delegated to humans. Further, it allows the reuse principle stated above and correlation among different types of information objects that need to be shared.

Practical examples of Data Dictionaries include the NASA Planetary Data System Data Dictionary [14] and the ESA BEAT Data Dictionary[15].

### 2.5.1 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.

Additionally, it is important to have a common meta model for a data dictionary so that they can be captured and exchanged in a common way. This is critical to building things like metadata registries (discussed in Section 3.2.2) for capturing and sharing data elements across projects. Further, it is important to recognize that data dictionaries cannot be constructed without a domain model. This relationship is depicted in .Figure 12.

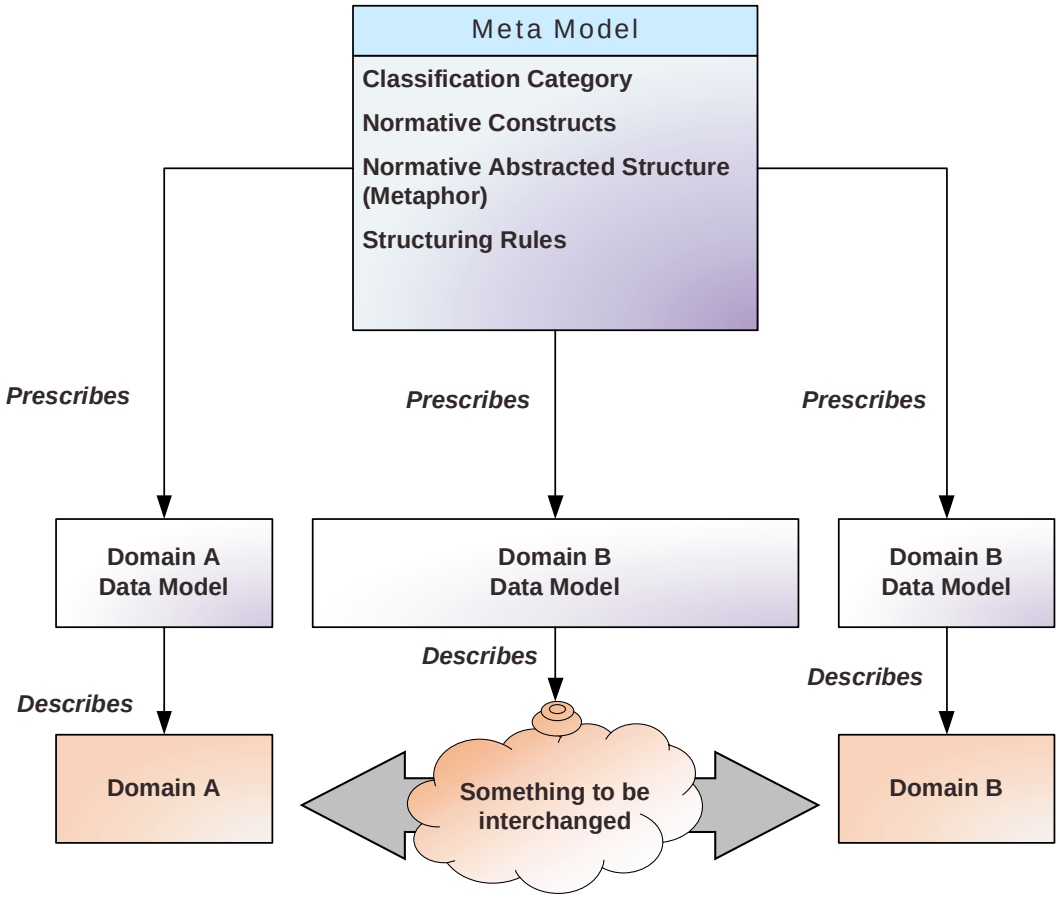



**Figure 12. Data Models, Meta Models and Domains**

## 2.6 RELATED WORK

### 2.6.1 METADATA

The need for metadata and its use has been rigorously studied in the Digital Library community and in particular has been a major focus of the Joint Conference on Digital Libraries (JCDL) [16]. The OAI protocol [17] has defined metadata as a key factor in the Digital Library community, along with harvesting and archiving processes which may be mapped to this standard's definition of data correlation and product federation respectively. Metadata elements and methods of element integration have been shown by [6, 17-27] to be important in integration of heterogeneous data and data interoperability. Views of information in Object-Oriented (OO) terms have been studied by [28-30], which are similar to the OO-like approach taken by this standard. The Unified Modeling Language (UML) [28] has been used to represent information and (meta) models in OO terms as well. A single-schema generalized resource description that supports correlative search across heterogeneous models has been defined and implemented as part of the OODT [21, 31, 32] infrastructure.

### 2.6.2 META-MODELS

In the realm of meta models, the ISO/IEC 11179 [12] 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*.

**Table 2. ISO/IEC 11179 Attributes**

| Attribute | Value |
|---|---|
| ***Identifying Attributes*** | |
| Name | Single or multi word designation assigned to a data element. |
| Identifier | A language independent unique identifier of a data element within a Registration Authority. |
| Version | Identification of an issue of a data element specification in a series of evolving data element specifications within a Registration Authority. |
| Registration Authority | Any organization authorized to register data elements. |

| | |
|---|---|
| Synonymous name | Single word or multi word designation that differs from the given name, but represents the same data element concept. |
| Context | A designation or description of the application environment or discipline in which a name and/or synonymous name is applied or originates from. |
| Definition | Statement that expresses the essential nature of a data element and permits its differentiation from all other data elements. |
| ***Relational Attributes*** | |
| Classification scheme | A reference to (a) class(es) of a scheme for the arrangement or division of objects into groups based on characteristics which the objects have in common, e.g. origin, composition, structure, application, function etc. |
| Keyword | One or more significant words used for retrieval of data elements. |
| Related data reference | A reference between the data element and any related data. |
| Type of relationship | An expression that characterizes the relationship between the data element and related data. |
| ***Representational Attributes*** | |
| Representation category | Type of symbol, character or other designation used to represent a data element. |
| Form of representation | Name or description of the form of representation for the data element, e.g. 'quantitative value', 'code', 'text', 'icon'. |
| Datatype of data element values | A set of distinct values for representing the data element value. |
| Maximum size of data element values | The maximum number of storage units (of the corresponding datatype) to represent the data element value. |
| Minimum size of data element values | The minimum number of storage units (of the corresponding datatype) to represent the data element value. |
| Layout of representation | The layout of characters in data element values expressed by a character string representation. |
| Permissible data element values | The set of representations of permissible instances of the data element, according to the representation form, layout, datatype and maximum and minimum size specified in the corresponding attributes. The set can be specified by name, by reference to a source, by enumeration of the representation of the instances or by rules for generating the instances. |

| ***Administrative Attributes*** | |
|---|---|
| Responsible organization | The organization or unit within an organization that is responsible for the contents of the mandatory attributes by which the data element is specified. |
| Registration status | A designation of the position in the registration life-cycle of a data element. |
| Submitting organization | The organization or unit within an organization that has submitted the data element for addition, change or cancellation/withdrawal in the data element dictionary. |
| Comments | Remarks on the data element. |

On the subject of meta model integration, it is note worthy that several approaches address the problem of integration of metadata objects [19, 20, 33, 34], yet all have focused more on mediating schemas rather than ensuring that different information objects conform to a standard meta model. In this standard, we focus on prescribing that metadata objects conform to a standard model so that the comparison between metadata objects is straightforward.

### 2.6.3 DATA DICTIONARY

With respect to practical implementations of the data dictionary structures described in this standard, 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 [13].

### 2.6.4 DATA MODELS

On the subject of data models for resource description (described in this standard), the Dublin Core Elements for Electronic Resources [6] addressed the compelling need for standard attributes for describing electronic resources on the web. The 15 data elements where defined using the ISO/IEC 11179 framework.

### 2.6.5 OAIS

Similar to this standard's focus on metadata, the CCSDS OAIS reference model [8] 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

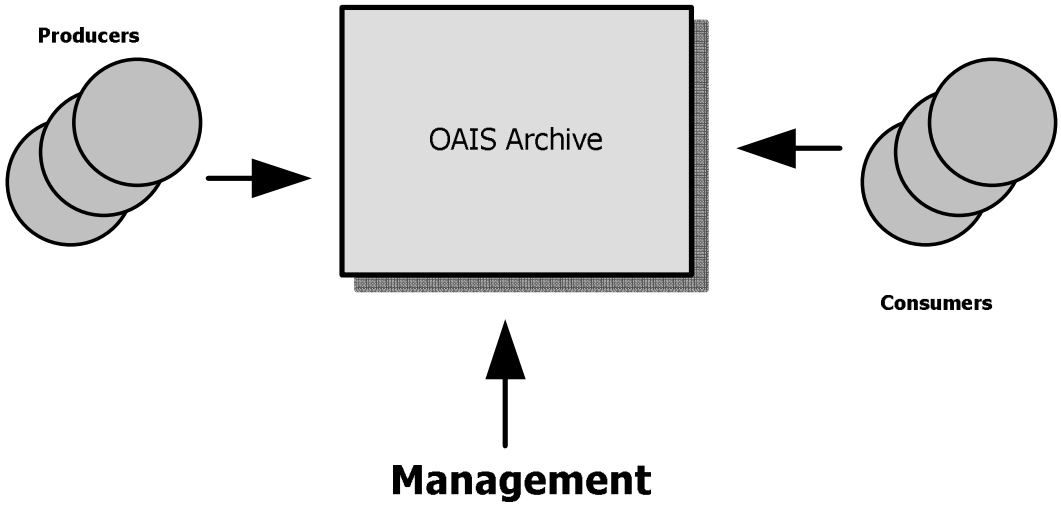



**Figure 13. The Open Archival Information System Reference Model**

information architecture, the OAIS DIPs, SIPs, and AIPs could all be considered information objects conforming to each respective package format specified in [8].

OAIS compliant archives are in the business of preserving, providing, managing and collecting information. Inherently they are most related to the archive software component 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 are compliant with the information objects described in this standard.

### 2.6.6 GRIDS

Recent work in the Grid Community [35] has characterized a class of distributed data interoperable systems as *Data Grids* [18, 26, 27, 36]. Data grids involve the identification of metadata, and different classes of metadata [18] which is required to make heterogeneous software systems interoperable. In the next paragraphs, some overviews of grid projects at various space agencies.

#### 2.6.6.1 SpaceGRID

ESA's Space Grid Study [37] 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

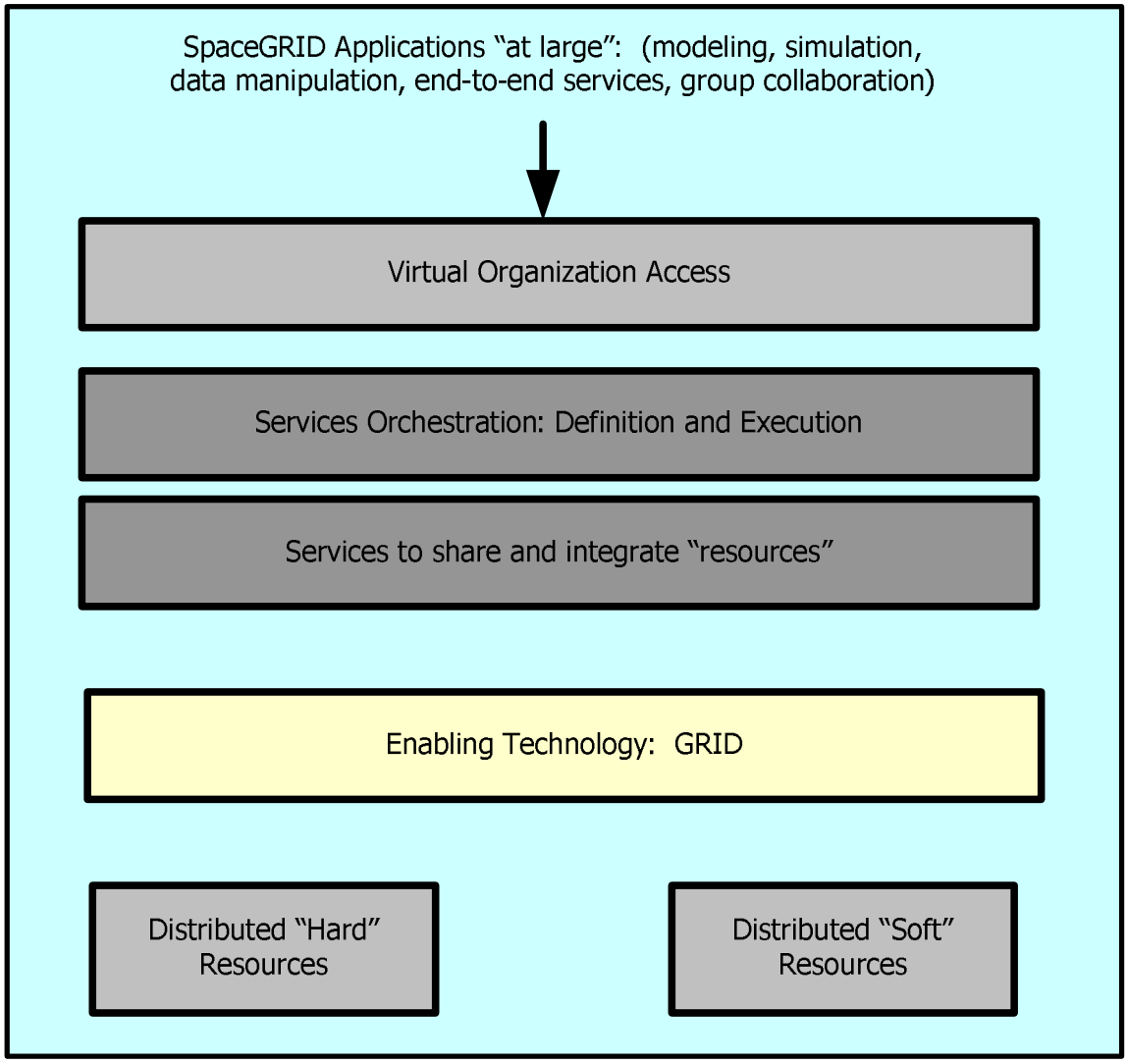



**Figure 14. SpaceGRID proposed infrastructure**

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:

1. *Flexibility*

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

Figure 14 depicts the proposed SpaceGRID infrastructure, which is very similar architectural model to that proposed in this standard. 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.

#### 2.6.6.2 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*, which our standard is aligned with. The relation between EOSDIS and this standard lies in the fact that EOSDIS is a domain-specific example of (1) earth science specific information objects, and packages, (2) earth science meta models and data dictionaries, (3) earth science metadata objects and (4) earth science domain models and ontologies.

#### 2.6.6.3 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 [38]. 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 information objects and information packages made available by the EDG system.

#### 2.6.6.4 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 [22, 35] grid middleware infrastructure to distribute, process, retrieve and search for astrophysical science data. The components of NVO are essentially the components of this standard: (1) a well defined information architecture, including standard information objects (or astrophysical data products), (2) standard metadata to describe the information objects, and (3) standard software components (in the form of grid services) to exchange data and cooperate for science.

# 3 SOFTWARE COMPONENTS FOR INFORMATION ARCHITECTURE

In addition to data standards, this document also describes the specification of standard information management objects (IMOs)[1] used for the access, distribution, capture and management of information objects. Two types of IMOs have been 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. On the other hand, 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.

In Section 3.1 and 3.2 we describe pIMO and aIMO components respectively. We then offer a small survey of related work in Section 3.3.

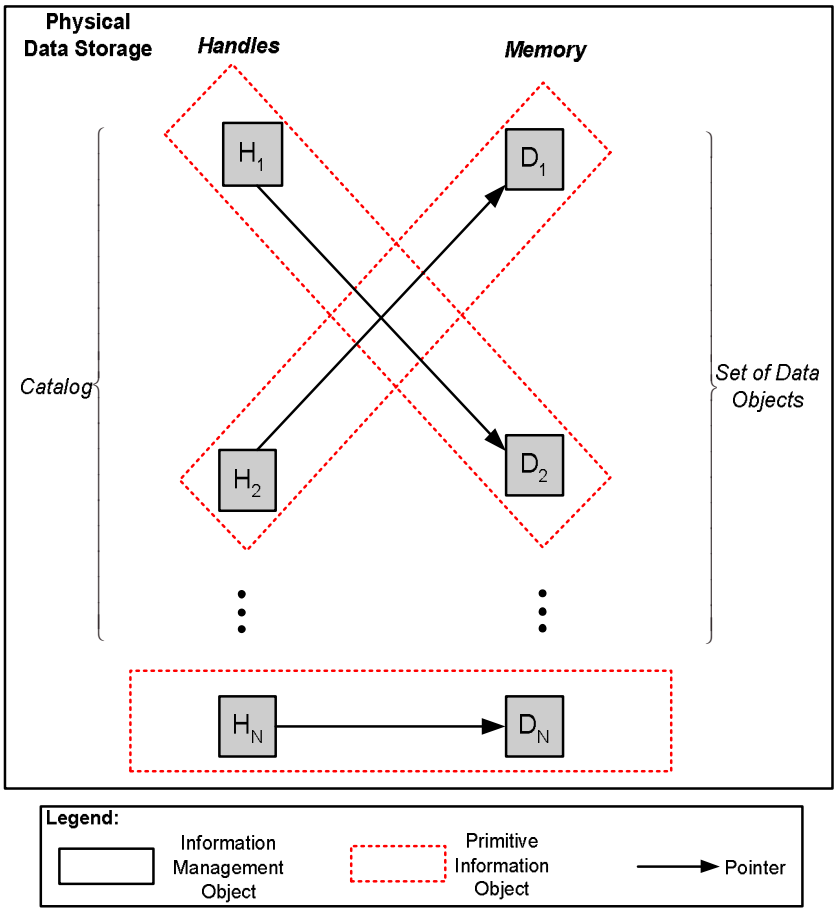



**Figure 15. The Internal Structure of a Physical Data Storage**

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

## 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. We identify two types of pIMO: *Data Store Object (DSO)*, and *Query Object (QO)*. These objects (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. In this document, we identify two constituents of physical data storages:

1. **Memory.** the physical location of the data in the data storage
2. **Handles.** the index catalog of pointers to memory containing data objects

These are key constituents of physical data storages because they 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 15. We now describe the two types of primitive information object, data store objects and query objects, in more detail.

### 3.1.1 DATA STORE OBJECT

The DSO component (shown in both the UML and RASDS notations[2] in Figure 19) is attached to an underlying physical data storage and supports *putting* and *getting* information. Figure 16 and Figure 17 depict the *put* and *get* operations of the DSO, respectively. The *get* operation takes an *identifying handle* as input (ranging from a simple memory address to a string identifier) and returns the *data object* (DO) 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 determined by the catalog and ingestion process of the underlying physical data storage.

[2] The software components in this document are shown using both UML and RASDS notations to ensure conformance to the ongoing work in the System Engineering Working Group on the RASDS architecture standard, as well as ensure conformance to such a broadly accepted modeling notation such as UML

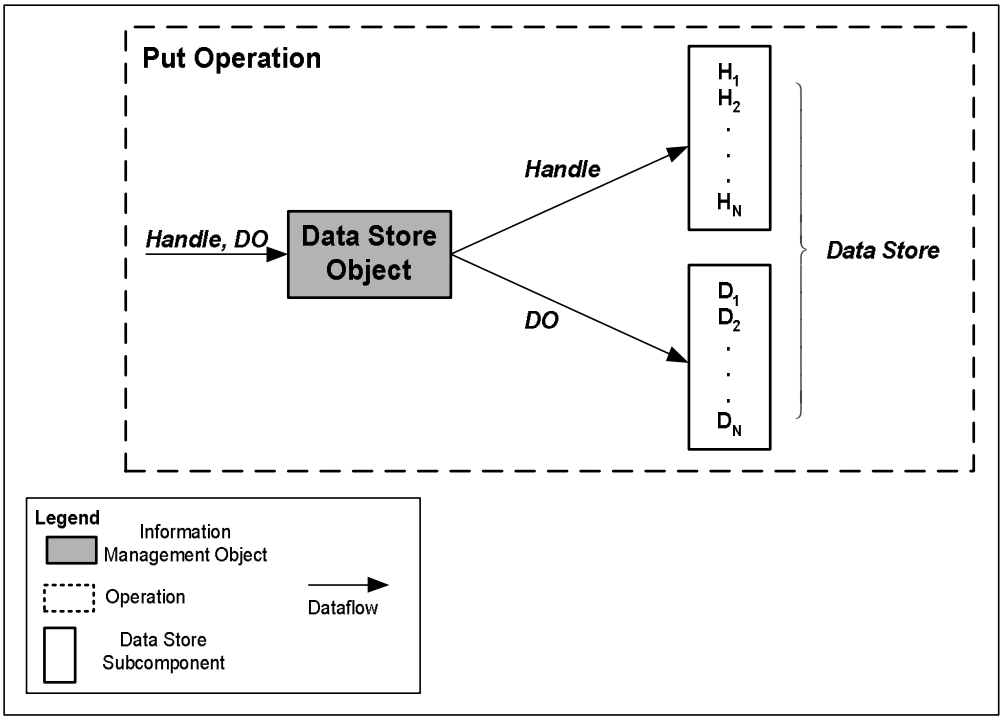



**Figure 16. The *Put* Operation of the Data Store Object**

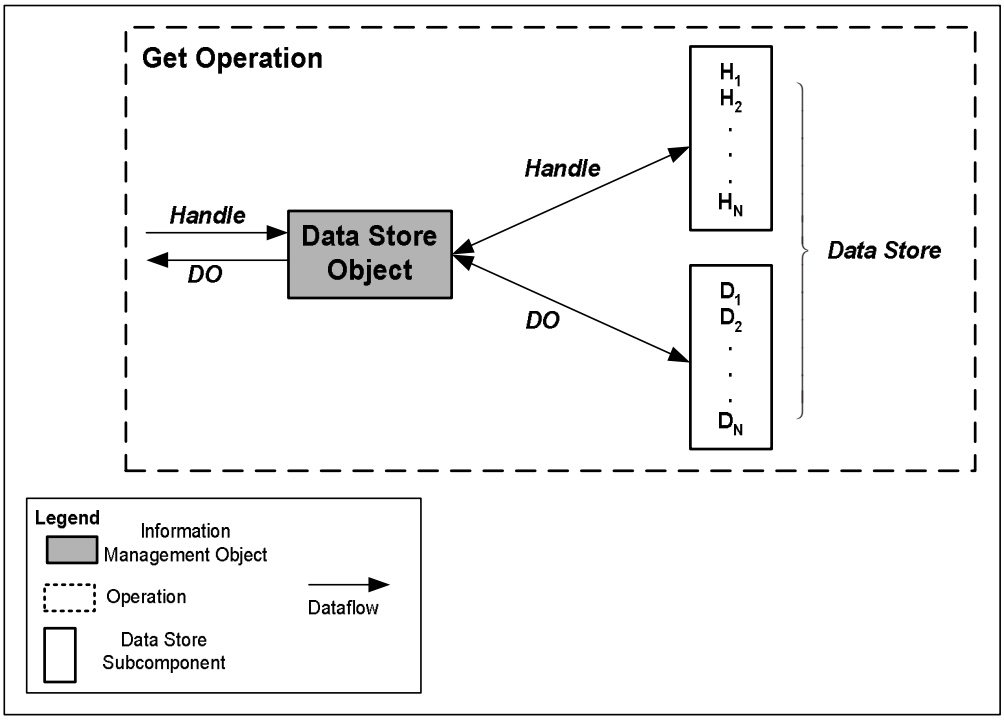



**Figure 17. The *Get* Operation of the Data Store Object**

### 3.1.2 QUERY OBJECT

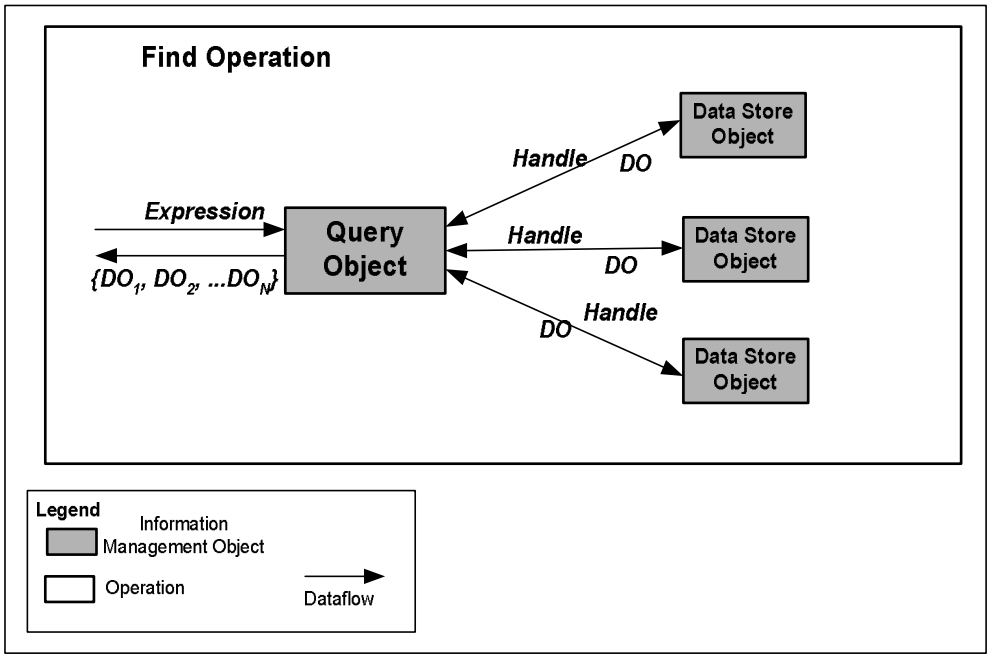



**Figure 18. The *Find* Operation of the Query Object**

The Query Object (shown in Figure 20 in UML and RASDS notation) enables retrieval of data in the form of Information Objects. Information Objects 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 information object is then returned to the caller of the find operation. A find invocation may return zero or more Information Objects. Figure 18 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.

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

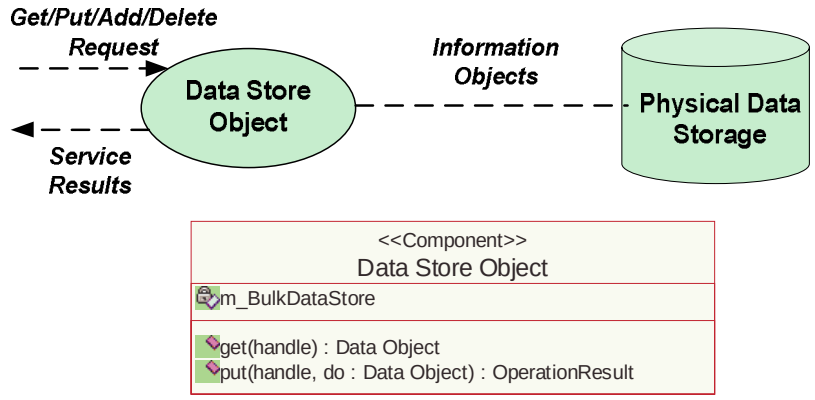



**Figure 19. A Data Store Object and its Corresponding UML View**

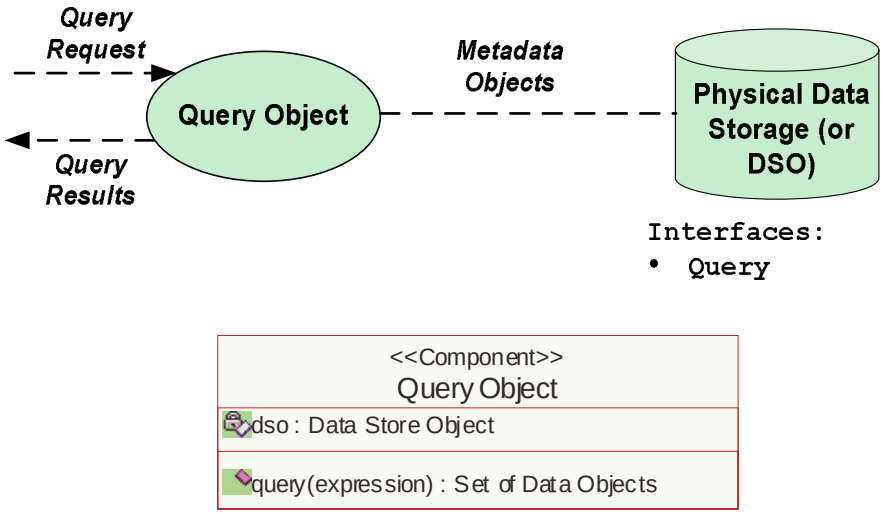



**Figure 20. A Query Object and its Corresponding UML View**

### 3.2.1 REPOSITORY SERVICE OBJECT

The *Repository Service Object* component is depicted in Figure 21. 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 DSO 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

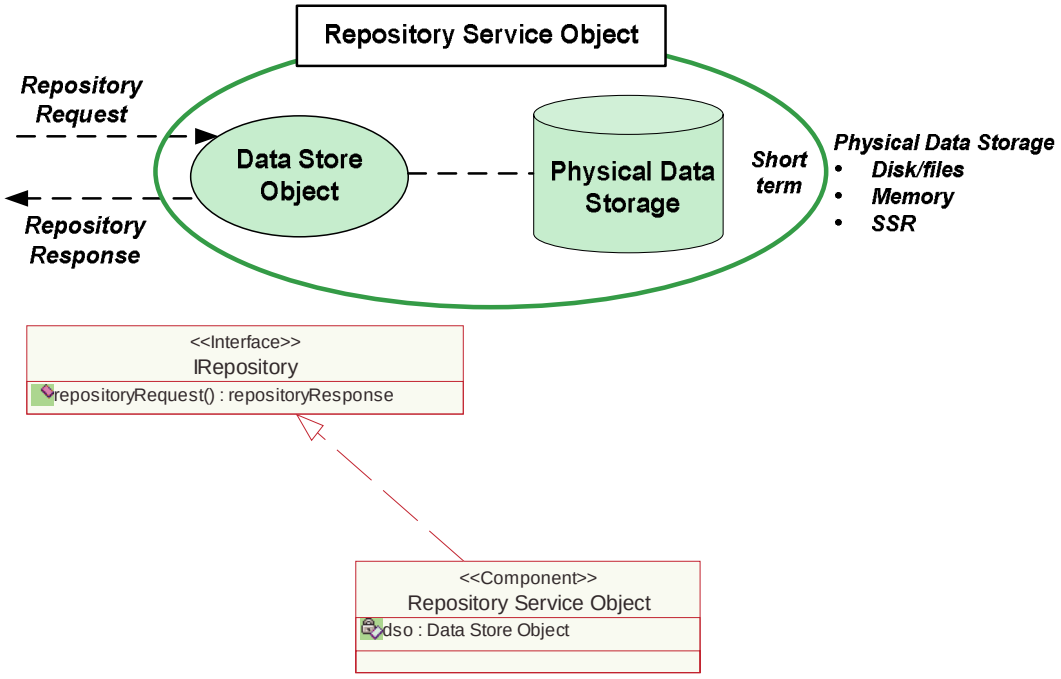



**Figure 21. Repository Service Object and its corresponding UML diagram**

objects. Information objects can be retrieved from the repository via the request interface, and a response from the repository is provided. The Repository Service Object also provides basic *put* capabilities of information objects using the capabilities of its associated DSO.

#### 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. We have identified three main dimensions in our preliminary taxonomy: *repository object type*, *object properties*, and *object description* each of which are further explained in this section.

First, we identify repository objects via their *type*. Type provides a quantifiable grouping for a family of repositories with similar functional and non-functional properties. This document identifies four key repository types: *Data Store*, *Product Repository*, *Short-term Archive*, and *Long-term Archive*. Secondly, we identify *object properties* dimension that 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, we plan to further categorize these properties 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

**Table 3. 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) |
| Product Repository | Component that stores data products and higher level products, possibly including metadata. Supports retrieval of data products through possibly complex methods, and processing. | Advanced Component supporting retrieval of possibly complex data products, including their metadata. |
| Short-term Archive | No support for permanence. Stores products for short term (e.g. less than 10 years), and allows retrieval of products. | Archive for short-term preservation of data products, get, put, and query retrieval methods. |
| 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). |

natively handle either primitive get/put operations, or higher level operations possibly requiring both querying and processing of data being returned. Finally we identify the *object description* dimension for a repository. The description dimension identifies key services and responsibilities of the repository when deployed together with a set of other software components. In Table 3, we summarize our brief taxonomy and classification of repositories.

### 3.2.2 REGISTRY SERVICE OBJECT

The *Registry Service Object* component provides an interface to retrieve the class of Information Objects referred to as *Metadata Objects*. Metadata Objects (recall Section 2) are Information Objects only containing representational information. The registry service object returns metadata objects which satisfy a particular query expression provided by the user of the *metadataQuery* interface. Figure 22 depicts a Registry Service Object.

Similar to the Repository Service Object, there also exist different *classes* of Registry

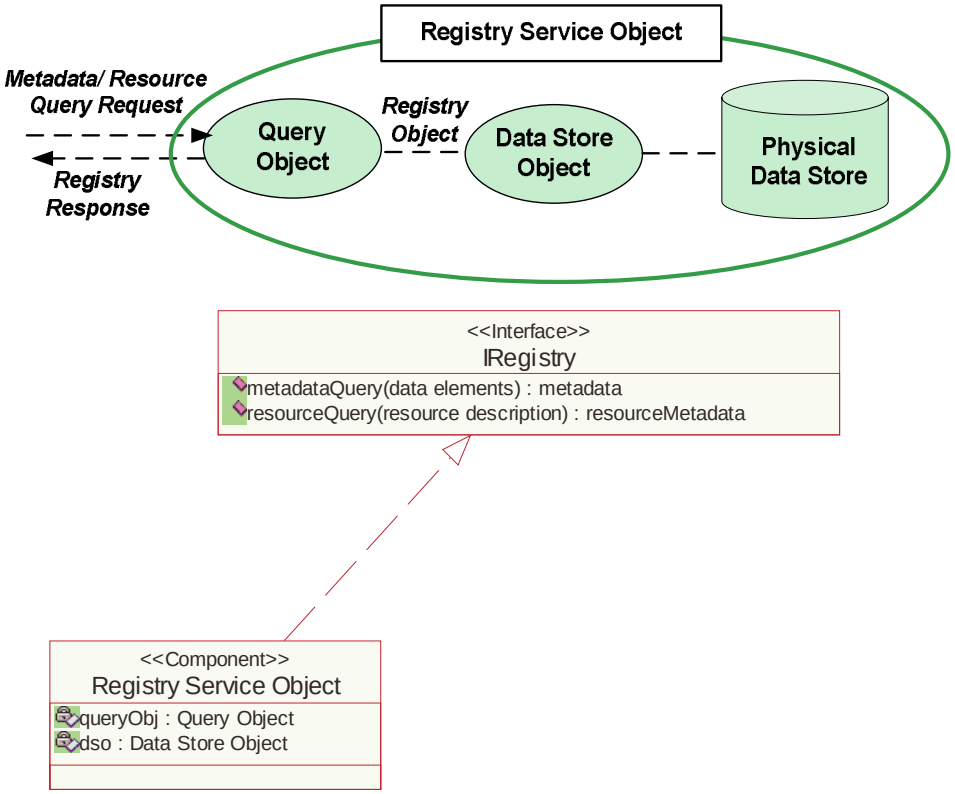



**Figure 22. A Registry Service Object and its Corresponding UML View**

Service Objects. We attempt to identify a representative subset of these classes below.

### 3.2.2.1 A Taxonomy of Registry Service Objects

In this section, we identify three main classes of registries and classify them along a particular set of dimensions. These dimensions entail 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 class of data 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 [12] , or explicitly by specifying the data element's ID [12].

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) [39] 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 [21,

**Table 4. 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 |

31, 32]. 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.

Finally, 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. The resource description is described using the notion of profile presented in Section 2. This enables description of an information object using the representational information defined with a profile. Science catalogs such as the Simbad Astrophysics Catalog [40] are examples of resource registries that serve information about data products. Resource registries can also point to other resource registries to enable discovery of information objects across distributed registries.

We acknowledge that 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 we identify this contribution as an element of on-going work within this document and within the IA-WG[3]. The taxonomy of registry service objects are summarized in Table 4.

### 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 an element of domain processing or transformation. The domain processing element essentially translates data objects from the underlying format used by the data store, to the required format specified by the user. The product service object serves as a standard interface to

[3] CCSDS Information Architecture Working Group

heterogeneous data sources, and allows for the querying the information objects via a

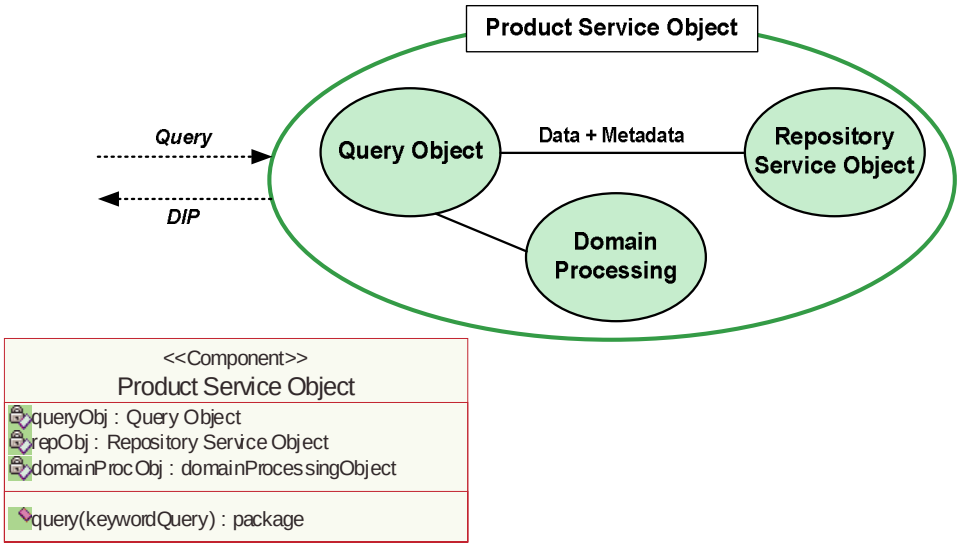



**Figure 23. A Product Service Object and its Corresponding UML View**

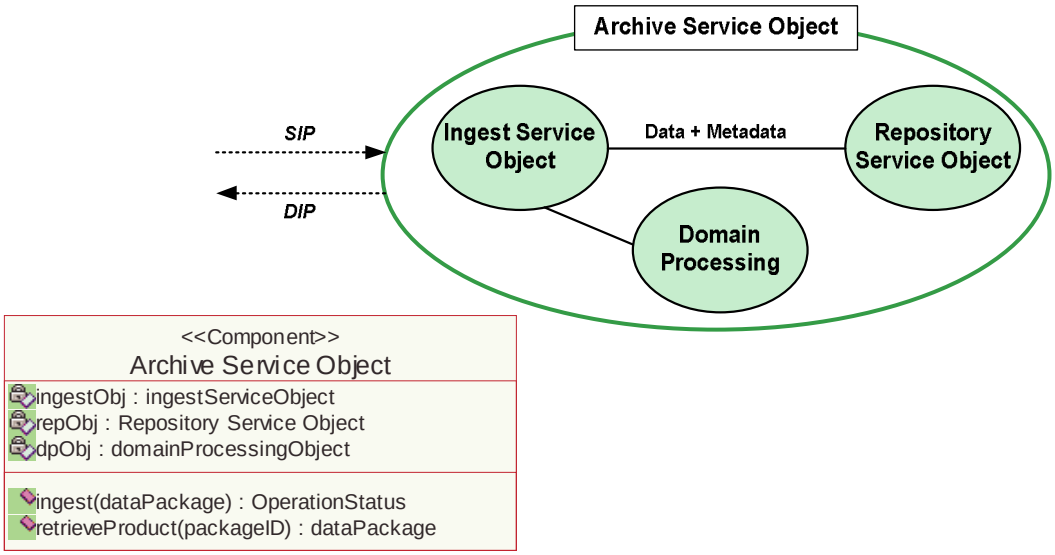



**Figure 24. An Archive Service Object and its Corresponding UML View**

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 23

### 3.2.4 ARCHIVE SERVICE OBJECT

*Archive Service Objects* provide a standardized architectural component 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 both data and metadata objects. These tasks can then be managed via

a rule-based policy which given a set of criteria such as time, task type, ingestion type,

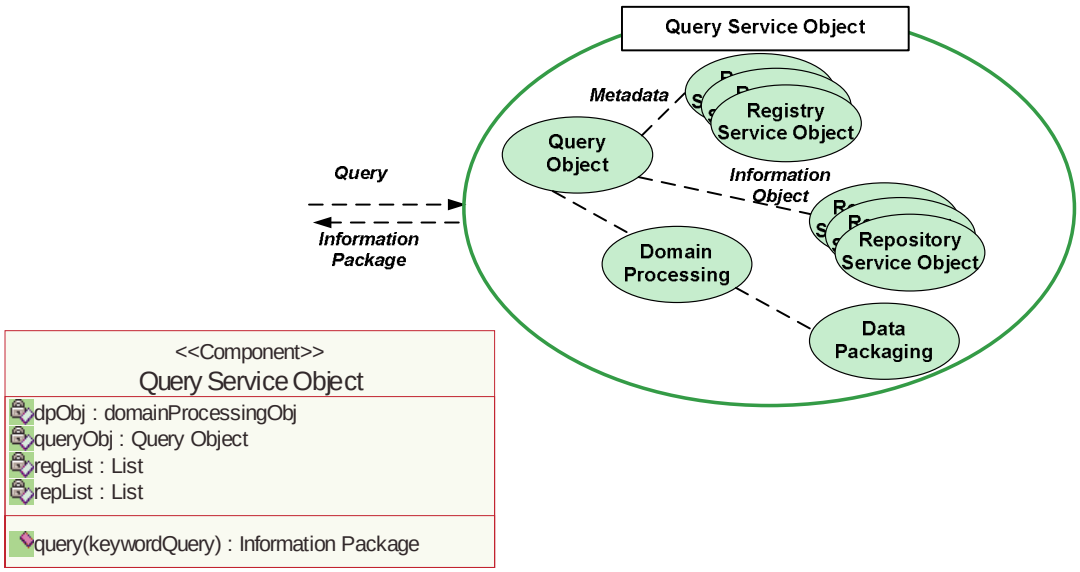



**Figure 25. A Query Service Object and its Corresponding UML Views**

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* [41-43]. Further, 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 [8]. An Archive Service Object is shown in Figure 24.

### 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 the appropriate product service objects, as well as repository service objects. This is accomplished by querying registry service objects in order to discover the location of the appropriate repository, product service or information objects by which to obtain data from. 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 of Information Objects, translations, and other types of advanced processing. Typically the federation of returned Information Objects includes a set of metadata objects (described the federated Information Object package returned) and thus in turn is an information object. We coin this federation of Information Objects along with its respective metadata objects an information package (recall Section 2). Figure 25 depicts a Query Service Object.

## 3.3 RELATED WORK

The work described in this document is based on a foundation of related work in the areas of data grids, CCSDS archiving standards and architectural styles for network-based software systems. We highlight each of these prior works below as they are all related to

the area of supporting architecture-based, information management via software components.

Chervenak et al. [18, 26, 27] define a *layered services architecture* [2] of software components that can be used to federate heterogeneous data resources, which is related to the federation of space data system resources described in this standard. Further, we distinguish our standard from data grids through our definition of a Domain-Specific Software Architecture (DSSA) [3]and Domain Reference Architecture for the Space Data Systems Domain. Chervenak et al. are focused on systems with super-computing resources in terms of processing, memory, and network resources, whereas Space Data Systems are typically embedded systems which clearly have very different memory resources, bandwidth availability and latency.

The OAI protocol [17] and OAIS reference model [8] seek to define functional components for digital library style systems, and archival systems in general, but their focus on the overall problem, independent of domain, is not practical for consideration in Space Data Systems. Further, the OAIS model does not specify components at the level at which a system can be decomposed and implemented. Our goal; however, is to provide architecture and a set of components which can be used to compose OAIS compliant systems.

Fielding [5] defines a set of *software architectural styles* for network based software systems which help to motivate the interaction mechanisms of certain software components in this standard. Software architectural styles provide a set of standard component types, a set of interaction mechanism types (or *connector types*) and heuristics which guide the composition and deployment of components and connectors in the specified style [2-4]. Long term goals of this standard include defining a standard architectural style (e.g. set of components, connectors and valid configurations) for information architecture in space data systems and beyond.

Of particular interest are the *client-server* and *peer-to-peer* styles because of their practical ramifications within this standard. For instance, the functional software objects defined in this standard typically communicate use one or both of these interaction mechanisms or styles. A practical example is the query service object, defined in this standard, which interfaces with the underlying data stores (or data store objects) in a client-server fashion, where the query object is acting as the client and the data stores are acting as the servers. On the other hand, the query service object may operate with other query service objects in a peer-to-peer style of interaction, during its discovery search for resources that satisfy a particular query.

# 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 of Information Architecture, with respect to the topics introduced in Section 2. Section 4.2 discusses the Functional View of Information Architecture, with respect to the topics introduced in Section 3.

## 4.1 INFORMATION VIEW

Figure 26 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

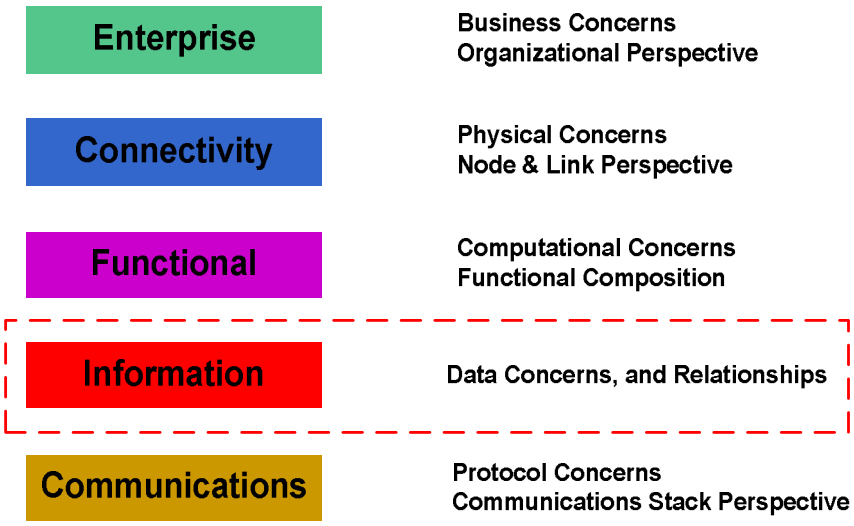



**Figure 26. Information View in Perspective**

extensively in **Section 2**.

## 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 Reference Architecture for Space Data Systems (RASDS) [1] work that is being conducted in parallel to this effort. Furthermore, this work is an elaboration of the Information View and a treatment of the Information Management Objects (IMOs) from the Functional View of RASDS.

# 5 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 |
|---|---|
| Data Dictionary 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 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 |
| Data Object Format Specification | *Standard Formatted Data Units — Control Authority Data Structures.* Blue Book. Issue 1. November 1994. http://www.ccsds.org/documents/632x0b1.pdf |

# 6 GLOSSARY

| Term | Definition |
|---|---|
| Metadata | Structured data which describes other data (not including metadata itself) |
| Meta Models | Set of data elements describing the metadata object, used to capture metadata in an information object |
| Schema | A set of semantic units, along with their attributes. A set of Metadata Elements. |
| Data Element | An OO class-like representation of metadata. The Data Element class itself contains structural information regarding its own structure, and the Data Element instance serves as a descriptor for Data Value. An example of a Data Element instance would be the Data Element "Author", used in Dublin Core. |
| Information Object | A compositional object containing an internal Data Object and Representational Information which describes the structure of the internal Data Object with structure information and which prescribes the nature of the internal Data Object with semantic information. |
| 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 in information, or data-intensive systems. |
| 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. |
| Data-Intensive System | Any system which is IO-bound. |
| Metadata Catalog Service | A Service providing the storage and retrieval of descriptive metadata in Grid-based systems. |
| Grid Computing | A new paradigm focusing on supporting Virtual Organizations, and the sharing, distribution, and |

| | |
|---|---|
| | retrieval of heterogeneous information, stored in heterogeneous data sources, possibly across many organizations. |
| Grid-based systems | Any software system which is modeled upon Grid Computing, either the Data aspect of Grid Computing (i.e. Data Grids), or the Computational Aspects of Grid Computing (i.e. Computational Grids). |
| MCAT Catalog | The San Diego Supercomputing Center's Metadata Catalog Service. |
| | |

# 7 REFERENCES


[1] P. Shames and T. Yamada, "Reference Architecture for Space Data Systems," presented at 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems (RCSGO), Pasadena, CA, 2003.

[2] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*: Prentice Hall, 1996.

[3] M. Shaw and P. Clements, "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems," presented at 21st IEEE International Computer Software and Applications Conference (COMPSAC), 1997.

[4] D. E. Perry and A. L. Wolf, "Foundations for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes (SEN)*, vol. 17, 1992.

[5] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," in *Information and Computer Science*. Irvine, CA: University of California, Irvine, 2000.

[6] DCMI, "Dublin Core Metadata Element Set," 1999.

[7] PDS, "SFDU Usage (http://pds.jpl.nasa.gov/documents/sr/Chapter16.pdf)," 2003.

[8] CCSDS, "CCSDS OAIS Reference Model," 2002.

[9] Hyperdictionary.com, "Definition of Compression," web, 2004.

[10] L. Reich, "XML Packaging for the Archiving and exchange of Binary Data and Metadata," presented at Open Forum on Metadata Registries, 2003.

[11] C. A. Mattmann, P. M. Ramirez, D. J. Crichton, and J. S. Hughes, "Packagaging Data Products using Data Grid Middleware for Deep Space Mission Systems," presented at 8th International Conference on Space Operations (Spaceops-2004), Montreal, Canada, 2004.

[12] ISO/IEC, "Framework for the Specification and Standardization of Data Elements," ISO/IEC, Ed. Geneva, 1999.

[13] CCSDS, "Data Entity Dictionary Specification Language (DEDSL)," 2000.

[14] Planetary-Data-System, "Planetary Data System Data Dictionary Lookup (http://pds.jpl.nasa.gov/tools/data_dictionary_lookup.cfm)," PDS, 2004.

[15] ESA, "Envisat/ERS Atmosphere Data Processing using the Basic Envisat Atmospheric Toolbox (BEAT) (http://envisat.esa.int/envschool_2003/practicals/Beat/beat-presentation.pdf)."

[16] JCDL, "Joint Conference on Digital Libraries," vol. 2004, 2004.

[17] O. A. I. (OAI), "OAI Protocol for Metadata Harvesting," January 2004 2004.

[18] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman, "A Metadata Catalog Service for Data-Intensive Applications," presented at IEEE International Conference on Supercomputing, 2003.

[19] S. Spaccapietra, "View Integration: A Step Forward in Solving Structural Conflicts," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, pp. 258-274, 1994.

[20] J. Najjar, E. Duval, S. Ternier, and F. Neven, "Towards Interoperable Learning Repositories: the ARIADNE Experience," presented at IADIS International Conference WWW/Internet, 2003.

[21] C. Mattmann, D. Crichton, J. S. Hughes, S. Kelly, and P. Ramirez, "Software Architecture for Large-scale, Distributed, Data-Intensive Systems," presented at 4th IEEE/IFIP Working Conference on Software Architecture (WICSA-4), Oslo, Norway, 2004.

[22] C. Kesselman, I. Foster, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputing Applications*, pp. 1-25, 2001.

[23] IEEE, "Learning Object Metadata (LOM)," standard 1484.12.1, 2002 2002.

[24] Globus, "Grid Metadata Catalog Service," 2004.

[25] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," *IEEE Computer*, pp. 37-46, 2002.

[26] A. L. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, pp. 1-12, 2000.

[27] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services," presented at IEEE Supercomputing Conference, Baltimore, MD, 2002.

[28] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Languge User Guide*: Addison Wesley, 1998.

[29] J. M. Bruel and Z. Bellahsene, "Advances in Object-Oriented Information Systems," in *Lecture Notes in Computer Science*. Montpellier, France, 2002.

[30] B. Stroustrup, "What is Object Oriented Programming?," ATT 1991 1991.

[31] D. J. Crichton, J. S. Hughes, and S. Kelly, "A Science Data System Architecture for Information Retrieval," in *Clustering and Information Retrieval*, W. Wu, H. Xiong, and S. Shekhar, Eds.: Kluwer Academic Publishers, 2003, pp. 261-298.

[32] D. Crichton, S. Hughes, S. Kelly, and P. Ramirez, "A Component Framework Supporting Peer Services for Space Data Management," presented at IEEE Aerospace Conference, Big Sky, Montana, 2002.

[33] G. Yan, W. K. Ng, and E. Lim, "Product Schema Integration for Electronic Commerce - A Synonym Comparison Approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, pp. 583-598, 2002.

[34] L. Haas and E. Lin, "IBM Federated Database Technology," *IBM's DB2 Developer Domain*, 2000.

[35] Globus, "The Globus Alliance," vol. 2004, 2004.

[36] R. W. Moore, C. Baru, R. Marciano, A. Rajasekar, and M. Wan, "Data-Intensive Computing," in *The Grid: Blueprint for a New Computing Infrastructure*, vol. 1, C. Kesselman and I. Foster, Eds.: Morgan-Kaufman Publishers, 1999.

[37] P. G. Marchetti, S. Ansari, H.-P. d. Koning, L. Fusco, G. Kreiner, H. Evans, E. Daly, S. Beco, V. Perrin, C. Perry, P. Vielcanet, D. Rodgers, F. Lei, and C. Morris, "SpaceGRID Study Final Report," ESA SGD-SYS-DAT-TN-100-1.2, 2002.

[38] EU-DataGrid, "The DataGrid Project (http://web.datagrid.cnr.it/LearnMore/index.jsp)," 2004.

[39] W3C, "Web Services Description Language (WSDL)," vol. 2004, 2004.
[40] "The SIMBAD astronomical database (http://cdsweb.u-strasbg.fr/Simbad.html)," 2004.
[41] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda, "Mapping Complex Workflows onto Grid Environments," *Journal of Grid Computing*, vol. 1, pp. 25-39, 2004.
[42] E. Deelman, R. Plante, C. Kesselman, G. Singh, M. Su, G. Greene, R. Hanisch, N. Gaffney, A. Volpicelli, J. Annis, V. Sekhri, T. Budavari, M. Nieto-Santisteban, W. O'Mullane, D. Bohlender, T. McGlynn, A. Rots, and O. Pevunova, "Grid-Based Galaxy Morphology Analysis for the National Virtual Observatory," presented at IEEE Conference on Supercomputing, Phoenix, AZ, 2003.
[43] J. Blythe, E. Deelman, and Y. Gil, "Automatically Composed Workflows for Grid Environments," *IEEE Intelligent Systems*, vol. 19, pp. 16-23, 2004.