INTELLIGENT RESOURCE DISCOVERY USING ONTOLOGY-BASED RESOURCE PROFILES

J. Steven Hughes [(1*)], ***Dan Crichton*** [(1)], ***Sean Kelly*** [(1)], ***Chris A. Mattmann*** [(1),] ***Jerry Crichton*** [(1)], ***Thuy Tran*** [(1)]

[(1)] *Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, California, 91109 USA*
Email:steve.hughes@jpl.nasa.gov, dan.crichton@jpl.nasa.gov, sean.kelly@jpl.nasa.gov, gerald.crichton@jpl.nasa.gov, thuy.tran@jpl.nasa.gov, chris.mattmann@jpl.nasa.gov

ABSTRACT

Successful resource discovery across heterogeneous repositories is highly dependent on the semantic and syntactic homogeneity of the associated resource descriptions in each repository. Ideally, consistent resource descriptions are easily extracted from each repository, expressed using standard syntactic and semantic structures, and managed and accessed within a distributed, flexible, and scalable software framework. In practice however, seldom do all three of these elements exist. To help address this situation, the Object Oriented Data Technology (OODT) project at the Jet Propulsion Laboratory has developed an extensible, standards-based resource description scheme that provides the necessary description and management facilities for the discovery of resources across heterogeneous repositories. The OODT resource description scheme can be used across scientific domains to describe any resource. It uses a small set of generally accepted, broadly-scoped descriptors while also providing a mechanism for the inclusion of domain-specific descriptors. In addition, the OODT scheme can be used to capture hierarchical, relational and recursive relationships between resources. In this paper we expand on prior work and describe an intelligent resource discovery framework that consists of separate software and data architectures focusing on the standard resource description scheme. We illustrate intelligent resource discovery using a case study that provides efficient search across distributed repositories using common interfaces and a hierarchy of resource descriptions derived from a complex, domain-specific ontology.

Keywords: Resource Discovery, Ontology, OODT, Semantic, Framework, Architecture

1 INTRODUCTION

The Object Oriented Data Technology (OODT) project was funded in 1998 by NASA's Office of Space Science (OSS). Its task was to develop a national software *framework* for sharing data across heterogeneous, distributed data repositories. The resulting framework today consists of separate but complementary data and software *architecture*s that enable the sharing of data and computational resources across multiple science and engineering disciplines (e.g., planetary science, cancer research, and earth science).

The *software architecture* provides reusable software components with homogeneous interfaces, allows new components to be easily integrated into the framework, and provides a mechanism to wrap (Sneed, 1997) legacy data system components with minimal impact. The complementary data architecture is domain-dependent, yet flexible: it can be instantiated in a single domain or even across multiple, different domains. Its effectiveness is primarily dependent on the maturity of the data model provided for the target environment. The framework also supports location independence in that the user describes *what* she wants, *not how* or *where* to get it. The framework can also be scaled to meet increases in the number of interconnected repositories.

As part of the *data architecture*, the design and development of a standards-based *resource description scheme* has enabled intelligent *resource discovery* across distributed heterogeneous repositories in support of the resource sharing task described above. In this paper we expand on prior work (Crichton, Downing, Hughes, Kincaid, & Srivistava, 2001, Crichton, Hughes, Kelly, & Hyon, 2000, Crichton, Hughes, Kelly, & Ramirez, 2002, Crichton, Hughes, & Kelly, 2003) and describe a architecturally compliant resource

discovery framework that consists of separate software and data components. We illustrate intelligent resource discovery across distributed repositories using a case study implementation that provides efficient search across distributed repositories using common interfaces and a hierarchy of resource descriptions derived from a complex, domain specific ontology. The rest of this paper is organized as follows. Section 2 presents background information and a survey of related work regarding resource description schemes, and software frameworks to support resource discovery and sharing. Section 3 describes the OODT resource description scheme's data architecture. Section 4 summarizes the supporting software architecture of the resource description framework. Section 5 provides an example case study using the resource description framework. Section 6 rounds out the paper.

2 BACKGROUND AND RELATED WORK

The intelligent resource discovery framework described in this paper is based on a foundation of related projects along with our own existing work in two key areas: (1) resource description schemes and (2) software frameworks that support resource discovery. In this section, we first expound on the related projects constructing resource description schemes and compare and contrast their foci and emphases to those of our own work. We follow by briefly summarizing some related software frameworks that support resource discovery.

Decker, Tangmunarunkit, & Kesselman (2003) define the *resource matching problem* for grid applications. The resource matching problem involves a user (or *agent*) selecting an appropriate set of resources that meet the requirements of an application that must execute in a grid environment. Decker et al. identified the inflexibility and ambiguity in current "flat" resource description schemes (e.g., selecting an operating system resource that is "UNIX compatible" when the resource descriptions only store flat named attributes, such as *OS*=Solaris, or *OS*=Linux). They propose a *Horn* and *F* logic based ontology resource matching approach (and accompanying resource description scheme) that separates the resource providers and the application requirements for a grid application. Furthermore, Decker et al. developed three initial resource description ontologies for their approach. The *Resource* ontology describes resources and their capabilities. The *Resource Request* ontology describes an application's request characteristics for a particular resource. The *Policy* ontology captures authorization and access privileges for resources based on user roles and authentication. Our work differs on several fronts from Decker et al.'s. First, our resource description scheme is grounded in query-based discovery and description of resources, as opposed to matching resources to a given job or application's requirements. In our framework, the requirements for a resource are provided via a DIS-style keyword query (which we describe further in Section 4.3), instead of a Horn or F logic based conjunction. Additionally, we have adapted and reused existing international metadata standards for resource description using Dublin Core (Dublin Core Metadata Initiative, 1999), and for data element description the ISO/IEC 11179 specifications (ISO/IEC-11179, 1999). Decker et al. have developed their own approach for description and have provided a brief example for identifying the appropriate (set of) high performance compute resources on which to execute a sample job.

The *Resource Description Framework* (Lassila & Swick, 1999), or RDF, is a W3C Recommendation for describing resources for use in a myriad of processes. RDF is meant to be used in activities such as resource discovery, intelligent software agents, cataloging, and the semantic web. The RDF data model entails three main elements. *Resources* are anything that can be described by an RDF statement. *Properties* are named metadata elements for a particular resource. Properties also define their valid values, the types of resources they can describe, and any relationships they share with other properties. *Statements* consist of: (1) a particular resource, (2) a named property for that resource and (3) the property's value. The OODT resource description framework shares several goals with RDF. Both RDF and the OODT resource description framework are meant to be overly general, and applicable to many domains. Additionally, both frameworks are XML-based notations. However, our work and RDF also differ significantly. The OODT resource description framework is quite flexible: it prescribes a standards-based data model for resources; however, it also allows the user to define domain specific data elements (called *profile elements*). RDF on the other hand is meant to be overly generic, and does not prescribe any standard set of data elements for resources: users can define whichever data elements they like. Furthermore, instead of RDF statements, the

OODT framework relies on ISO-11179 to define relationships, valid values, and resource types between metadata elements.

There are several other resource description frameworks. Singh, Bharathi, Chrevenak, Deelman, Kesselman, Manohar, et al. (2003) describe the Globus *Metadata Catalog Service (MCS)* component for data-intensive environments. The MCS resource description framework classifies resources into 5 categories: *User metadata*, *Virtual Organization (VO) metadata*, *Domain-Specific metadata*, *Domain-Independent metadata*, and *Physical metadata*. User metadata stores information about user resources, such as their roles and permissions. Virtual Organization metadata captures information about shared datasets, users, and privileges. Domain-specific metadata is stored and created by application communities to describe their resources. Domain-Independent metadata are general, broad-scope descriptors (such as those provided by Dublin Core) that describe resources independent of domain (e.g., Creator, Author, Logical Name and the like). Physical metadata stores information about physical resources, such as file systems, tape drives, and mass storage systems. The IEEE *Learning Object Metadata* (LOM) draft standard (IEEE-LTSC, 2005) is focused on describing "learning object" resources. Learning objects are defined to be digital entities that can be used during technology-supported learning. IEEE LOM defines a base schema, and a set of vocabularies to describe learning resources. The schema includes elements such as *Creator*, *Language*, *Coverage* and *Structure*. The IEEE LOM document also provides a mapping to the related data elements of the Dublin Core standard. Pouchard, Cinquini & Strand (2003) describe the Earth System Grid (ESG) ontology and its associated resource description model. ESG resources are categorized into six broad classes: *Pedigree*, *Scientific Use*, *Datasets*, *Services*, *Access*, and *Other*. The resource description framework offered by Pouchard et al. is not overly general, and intended to be of use in the earth science domain. A related earth science resource description framework is the *Semantic Web for Earth and Environmental Terminology (SWEET)* taxonomy, developed by Raskin, Pan & Mattmann (2004). The goal of the SWEET taxonomy is to enable semantic web and agent technologies to discover earth science resources, and to provide more meaningful queries and results when users search for earth science resources. SWEET is developed in OWL and RDF, and has been tested and evaluated within a sample prototype query interface.

Several software frameworks are built with the intention to facilitate resource discovery. We highlight and discuss a representative cross-section of them below.

The *Globus Toolkit* (Chervenak, Deelman, Foster, Guy, Hoschek, Iamnitchi, et al., 2002, Foster, Kesselman, Nick & Tuecke, 2002, Kesselman, Foster & Tuecke, 2001, Singh, Bharathi, Chrevenak, Deelman, Kesselman, Manohar, et al., 2003) is the *de facto* standard technology for grid computing. It subsumes a web-service middleware substrate (Apache AXIS) that provides basic middleware capabilities such as programming language abstraction, data marshalling and unmarshalling, and message delivery. The Globus toolkit deals with the general problem of managing, locating, and distributing *resources*. In grid environments enabled by Globus, resources are any one of the following: (1) *computational* resources, which include computer cycles, experiment test beds, and instruments, and (2) *data* resources, which include files, mass storage systems, databases, and replicas. Our work on the OODT framework and the Globus toolkit are closely related. Both projects provide complementary mechanisms (data models) for resource description and discovery, and both projects provide core middleware services which enable resources to be shared and discovered in the domain of grid computing.

Sangpachatanaruk & Znati (2005) describe the *Personalized Web Architecture (PWA)*, which is a way for each user to create a *Personalized Web (PW)* of interest, allowing resource discovery, location and description of only those resources in which the user is interested in. The enabling technology of the PW is a *semlet*, a semantic agent built on top of an ontology-based overlay-network, using P2P communication technologies. In the PW, a user describes what she is interested in, and then it is the semlet's responsibility to discover what resources are available, how to get to them, and how to present them back to the user. Additionally, the semlet has the responsibility of advertising its user's own resources, so that other semlets can discover the resources. For resource discovery, the semlet uses a WordNet (Beckwidth, Fellbaum, Gross, Miller, Miller, Tengi, 1990) based ontology tree (or WOT) to match user filters (using a similarity metric) against available resources. Similar to our work, resources are described using *profile*s, which are created on a per-user basis.

The *Web Services Resource Framework* (WS-RF) specification (Czajkowski, Ferguson, Foster, Frey, Graham, Sedukhin, et al., 2004) is a work-in-progress and is being developed as the next generation mechanism for resource description in web-service enabled environments, such as grid-computing. The main motivation behind WS-RF was the lack of the ability to manage stateful resources in the earlier incarnations of the web services specification. WS-RF adds the ability to declare, destroy, construct and manage stateful resources, and associate them with a web service. Resources are described using XSD schema, and are then associated with a web service through its PortType specification.

DSpace (Smith, Bass, McClellan, Tansley, Barton, Branschofsky, et al., 2003) jointly developed by MIT Libraries and Hewlett-Packard, is a distributed digital repository system that captures, stores, indexes, preserves, and redistributes the research material of a university in digital formats. The system is freely available as an open source system that can be customized and extended, and is built on top of other open-source tools, such as Apache Web server, the Tomcat Servlet engine, and the PostgreSQL relational database system. DSpace resources include anything an organization would like to manage in a digital library: papers, reports, standards documents, meeting minutes, and so on. Akin to our framework, resources in DSpace are described using the Dublin Core metadata. DSpace also supports transmission of its resource metadata using the METS standard (Library of Congress, 2005).

The *iGrid* information service (Aloisio, Cafaro, Epicoco, Fiore, Lezzi, Mirto, et, al., 2005) is a Grid-Security-Infrastructure (GSI) enabled software framework for publishing and discovering resources in a grid environment. iGrid resources are described using the relational model. Resource discovery is supported and relies on the availability of metadata including *CPU*, *Memory*, *File Systems* and *Network Interfaces* for particular compute resources iGrid supports publication of resource descriptions in a grid environment and allows subscription and notification, which the authors find to perform better than the typical *pull*-based model of discovering resource information, adopted in the Globus Metadata Catalog Service.

3 DATA ARCHITECTURE

Architecture is a term applied to both the process and the outcome of thinking out and specifying the overall structure, logical components, and the logical interrelationships of a computer system. We define *data architecture* to be the application of this concept to the data components involved in a computer system.

A key assumption in the development of a data architecture is that the data components to some degree model a specific problem domain. For example, in the space science domain, an image collected by a spacecraft instrument is modeled as a 2-dimensional structure of *lines* and *line samples*. The elements of this structure are *data numbers* that were collected within a context defined by the states of the instrument and spacecraft at the time the image was collected. In our work to capture data architecture, we have surveyed a representative set of approaches and decided to use *ontologies*. Our choice of ontologies is motivated and described below and throughout the paper using illustrative examples.

An ontology is a set of concepts—such as things, events, and relations—that are specified in some way in order to create an agreed-upon vocabulary for exchanging information within a domain. Historically there are many methodologies for defining and collecting various aspects of domain concepts into a domain model (for a survey, see Hull & King (1987)). For example the Entity-Relationship (E-R) model is a widely accepted data modeling technique that focuses on the definition of domain entities and their relationships. It defines an *entity* as something that exists either in concept or in actuality. Entities are defined using *properties* (often called attributes) and *relationship*s that relate two or more entities. Subsequently an E-R model is typically used to implement a database application using a specific record level model such as the Relational Model. Other methodologies involve the creation of taxonomies and controlled vocabularies. For this discussion we assert that *domain ontologies subsume the domain modeling information collected by any of these methodologies and specifically that which is necessary for intelligent resource discovery*. As such it becomes an appropriate choice of methodology for capturing a data architecture.

As examples, the image, instrument, and spacecraft mentioned earlier are all considered entities in the space science domain. The image entity is defined using attributes that describe its logical structure, namely the 2-D structure of *line*s and *line sample*s. Its relationship to the instrument and spacecraft entities help describe the context within which the image was collected. In addition, the instrument attributes *filter_name* and *exposure_duration* can be associated with the image through inference and the "`instrument` *produces* `image`" relationship. In a space science ontology, *image*, *instrument*, and *spacecraft* would each be defined as a class, and *lines* and *line samples* would be defined as properties of the image class. A level of intelligence is exhibited through the inference that the instrument class properties *filter_name* and *exposure_duration* can be associated with image.

3.1 Data Dictionary

A *data dictionary* (or controlled vocabulary) is a collection of terms and their definitions and is a basic component of a data architecture. It is a mechanism for defining entity attributes (also called *data element*s). Examples of data elements include *filter_name* and *exposure_duration* mentioned previously. Data elements also often carry additional *semantic* information, such as *value type* and *permissible values*. For example, the attribute *exposure_duration* is a data element in the planetary science data dictionary and is defined as the period of time over which data is collected by an instrument. It takes on a floating point value and is measured in units of milliseconds.

Special attributes, also called *meta-attributes*, are needed to collect data element definitional information. As part of our work on the OODT resource description framework, we have chosen the ISO/IEC 11179 standard (ISO/IEC-11179, 1999) to provide a base set of attributes to define domain specific data elements. As an international standard it also provides a basis for data element definition and classification that supports global data dictionary interoperability. The specification classifies the base set of attributes into four categories namely *identifying*, *definitional*, *representational*, and *administrative* as briefly summarized in Table 1.

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 organisation 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.

Table 1. ISO/IEC 11179 Attributes

The *identifying* category is used for the identification of a data element. For example, *exposure_duration* would be the value of the attribute "name". The *definitional* category is used to describe the semantic aspects of a data element and consists of a textual description that communicates knowledge about the data

element that typically is not captured by any of the basic attributes. The *relational* category describes associations among data elements and/or associations between data elements and classification schemes, data element concepts, objects, or entities. For example relating *exposure_duration* to an instrument entity provides critical information about how *exposure_duration* is to be interpreted. The *representational* category describes representational aspects of data element such the list of permissible data values and their type. For example, *exposure_duration* would be typed as floating point. Finally the *administrative* category provides management and control information.

	Dublin Core Data Elements
Title	A name given to the resource.
Creator	An entity primarily responsible for making the content of the resource.
Subject	The topic of the content of the resource.
Description	An account of the content of the resource.
Publisher	An entity responsible for making the resource available
Contributor	An entity responsible for making contributions to the content of the resource.
Date	A date associated with an event in the life cycle of the resource.
Type	The nature or genre of the content of the resource.
Format	The physical or digital manifestation of the resource.
Identifier	An unambiguous reference to the resource within a given context.
Source	A Reference to a resource from which the present resource is derived.
Language	A language of the intellectual content of the resource.
Relation	A reference to a related resource.
Coverage	The extent or scope of the content of the resource.
Rights	Information about rights held in and over the resource.

Table 2 Dublin Core Elements

3.2 Common Data Elements

With the advent of the web and the resulting explosion of electronic resources available for online access, there was a compelling need for a set of standard attributes for describing electronic resources. The Dublin Core (DC) (Dublin Core Metadata Initiative, 1999) initiative addressed this issue and developed the 15 data elements briefly summarized in Table 2. (It should be noted that the DC data elements were defined using the ISO/IEC 11179 framework.)

The DC attributes are by definition very general in scope and when used as search constraints do not always produce precise results. They were developed as common attributes to describe Internet electronic resources across all possible domains and this generality limits their ability to partition the search space into easily managed subsets. For example a search for electronic resources that have format = “image/jpeg” will typically result in a large number of images from many repositories. The solution to this problem as suggested by the DC Subject attribute is the addition of concepts from the specific domain. This of course presupposes the existence of a domain model.

3.3 Domain Data Models and Complexity

A domain data model is developed to meet the requirements of one or more applications. For example, in a planetary science archive application, instrument and spacecraft models would be relatively general in order to span the set of instrument and spacecraft instances. However, models for image data products would proliferate since as the focus of the archive, the image models would require very specific image attributes and relationships.

Assuming this situation exists, the implementation of a search capability using traditional cataloging technology would conceptually require the design of a catalog for each image type. For example, the search for specific images from among the approximately 49,000 images of the planet Mars from NASA's Viking mission is easily accomplished by designing a catalog for the Viking image model and loading 49,000 records. However Mars missions such as Mars Global Surveyor, Mars Odyssey, and the Mars Exploration Rover missions are providing thousands more images that should also be available as search results. Since each of these missions has a different image model, separate image catalogs could be implemented but with increasing management complexity. Alternately a single catalog that spans all images results in a database that is sparsely populated. The solution to this problem requires the design of a single image model that is sufficient for describing all image types. In the following we will describe a general resource description and then present a software architecture suitable for implementing an efficient search capability using the resource description.

```
<!ELEMENT profiles
 (profile*)>

<!ELEMENT profile
 (profAttributes,
  resAttributes,
  profElement*)>

  <!ELEMENT profAttributes
   (profId, profVersion?, profType,
    profStatusId, profSecurityType?, profParentId?, profChildId*,
    profRegAuthority?, profRevisionNote*, profDataDictId?)>

  <!ELEMENT resAttributes
   (Identifier, Title?, Format*, Description?, Creator*, Subject*,
    Publisher*, Contributor*, Date*, Type*, Source*,
    Language*, Relation*, Coverage*, Rights*,
    resContext+, resAggregation?, resClass, resLocation*)>

  <!ELEMENT profElement
   (elemId?, elemName, elemDesc?, elemType?, elemUnit?,
    elemEnumFlag, (elemValue* | (elemMinValue, elemMaxValue)),
    elemSynonym*,
    elemObligation?, elemMaxOccurrence?, elemComment?)>
```

Figure 1. Profile Schema - DTD

3.4 General Resource Description – Resource Profile

In our work on the OODT resource description framework, an electronic resource is described by a *resource profile*, an XML document that uses both domain specific attributes and the Dublin Core attributes to concisely describe a resource. The domain specific attributes are obtained from a domain specific ontology and provide the specificity required to meet the resolution requirements for search results. In this context, intelligence is exhibited by inferring additional attributes from modeled relationships. In contrast the Dublin Core (DC) elements provide common resource attributes such as Title and Format, the values of which are often derived from domain specific attributes.

A profile has three sections: the *profile attribute*s, the *resource attribute*s, and domain specific attributes (called *profile element*s). Profile attributes simply describe the profile using information such as identifier, type, and status. The identifier attribute is typically implementation dependent and could be an Object

```
<profile>
 <profAttributes>
  <profId>1.3.6.1.4.1.1306.2.11480003140</profId>
  <profType>profile</profType>
  <profStatusId>active</profStatusId>
 </profAttributes>
 <resAttributes>
  <Identifier>GO-J/JSA-SSI-2-REDR-V1.0:24I0131</Identifier>
  <Title>GO-J/JSA-SSI-2-REDR-V1.0:24I0131</Title>
  <Format>image/pds</Format>
  <resContext>NASA.PDS</resContext>
  <resClass>data.product</resClass>
  <resLocation>http://product_server_query_to_return_actual_product...
 </resAttributes>
 <profElement>
  <elemName>TARGET_NAME</elemName>
  <elemType>CHARACTER</elemType>
  <elemValue>IO</elemValue>
 </profElement>
 <profElement>
  <elemName>INSTRUMENT_ID</elemName>
  <elemType>CHARACTER</elemType>
  <elemValue>SSI</elemValue>
 </profElement>
 <profElement>
  <elemName>FILTER_NAME</elemName>
  <elemType>CHARACTER</elemType>
  <elemValue>RED</elemValue>
 </profElement>
 <profElement>
  <elemName>EXPOSURE_DURATION</elemName>
  <elemType>REAL</elemType>
  <elemValue>62.50</elemValue>
 </profElement>
```

Figure 2. Data Product Profile - Example

Identifier (OID), Universal Resource Identifier (URI), or a sequence number. The type attribute (see the *profType* element in Figure 2) is typically set to the value 'profile' but the value 'data dictionary' can be used to indicate that the profile is being used to capture data element information.

Resource attributes generically describe the profiled resource using the Dublin Core (DC) attribute set. All DC attributes are allowed but only Identifier is required. As part of the work on the OODT framework, three additional resource attributes have been added to identify the following: (1) the resource's local domain (which we call *resContext*), (2) the resource's classification (which we call *resClass*) , and (3) the resource's location (which we call *resLocation*). The valid values assigned to the DC attributes are typically derived from selected domain specific attributes. For example, the DC attribute Title, a "label" for the resource, could take on values from a domain attribute providing a resource label of some type.

The profile element section encodes domain specific attributes associated with the resource. This set of attributes includes the resource attributes specifically identified in the model as belonging to the resource as well as attributes that can be inferred from modeled relationships. For example, the image attribute *lines* and the instrument attribute *filter name* would both be included in an image profile. Each attribute is encoded into the profile with at least the *name* and *value* of the domain attribute. Other meta-attributes such as *unit* are allowed but optional. A profile to describe a planetary science image is illustrated in Figure 2. The profile element section encodes the inferred attributes *instrument_id*, *filter_name*, and *exposure_duration* for a precise characterization of the image. For referential purposes, the XML DTD for

the profile is provided in Figure 1. An XML Schema and XML/RDFS have also been developed, although we have elided these from the paper was they would be mostly redundant given in the DTD in Figure 1.

4 SOFTWARE ARCHITECTURE

The Object Oriented Data Technology (OODT) software framework that supports the OODT resource description framework described above consists of a set of distributed, cooperating software components. The major components of the OODT framework implement a metadata (profile) and data (product) model reified in the form of the *Profile* and *Product* server components. In addition, a *Query Service* component directs queries by traversing a network of connected profile and product servers, providing the veneer of a peer-to-peer network. The distributed services allow for the location and description of resources (profile queries) and retrieval of resources (product queries). Given that the focus of this paper is the resource discovery framework, we briefly sketch the capabilities and architecture of the OODT software components. A more detailed description of the software framework is provided in Crichton, Downing, et al., (2001), Crichton, Hughes, et al., (2000), Crichton, Hughes, et al., (2002), and Crichton, Hughes, et al., (2003).

4.1 Distributed Framework - Communications

OODT is a distributed system, wherein components may be dispersed geographically across a standard TCP/IP network, such as the Internet. Connectivity between components is provided by a standards based distributed systems implementation such as Java Remote Method Invocation (RMI) or the Internet Inter-ORB Protocol (IIOP) for CORBA-based communication. Latest developments include HTTP-based access.

The OODT software components support plug-ins that extend their base implementations by performing the work of querying both the metadata catalogs and the data repositories themselves. In this way, the OODT software is a framework in that application programmers extend and implement prescribed software objects and interfaces that directly integrate into the framework. This is in contrast to other software implementation efforts that may not specify ubiquitous interfaces. More work necessarily falls upon the users of the framework to support the prescriptive interfaces.

OODT's software framework defines three major components:

- *Profile Servers* that serve scientific metadata and can tell whether a particular resource can provide an answer to a query.
- *Product Servers* that serve data products in a system-independent format.
- *Query Servers* that accept profile and product queries and traverse the network of profile and product servers, collecting results. It is possible to access the query service through direct interface with the distributed computing interfaces (such as RMI and CORBA invocations), or through an HTTP interface.

As described earlier, profiles are metadata descriptions of resources; that is, they “profile” a resource by describing its inception and composition using the common data elements of the OODT resource description framework. Profile servers enable discovery of resources by providing the ability to search resource collections. In short, profile servers answer the question, “Where can I go to find out about X?”

A profile server’s primary responsibility is to provide a way to evaluate a query against the server’s set of profiles. Although users may access a profile server directly via its remote interface, it is far more common for queries to enter the system through the query server, which directs them transparently to and along directed graphs of appropriate profile servers (in the case when a profile’s *resLocation* field points to another profile server).

- IDENTIFIER=GO-J/JSA-SSI-2-REDR-V1.0:10I0012
- TARGET_NAME=IO AND EXPOSURE_DURATION > 60.0
- TARGET_NAME=IO AND NOT FILTER_NAME=RED
- DATA_SET_ID=ODY-M-THM-2-IREDR-V1.0 AND CENTER_LONGITUDE > 359.25
- DATA_SET_ID=MGN-V-RDRS-5-DIM-V1.0 AND IMAGE_ID=FO43S181

Figure 3. Example DIS Query Expressions

Upon receiving a query, the profile server's backend interprets the query passed in a way appropriate to the implementation. For example, a backend that stores information in a relational database may convert parts of the query into a database SQL query. For each matching profile, the backend constructs a list of matching profiles and returns them.

4.2 Product Servers

Product servers exist to provide a way to retrieve specific data products. Product servers accept the same query structure as profile servers, but instead of returning a list of matching profiles, they return matching products. Data products in this sense can be individual *data granules*, *datasets*, or *collections of datasets*, depending on the backend implementation in the product server and the way it handles queries and results.

When constructing a query, the user may indicate preferred MIME types. For example, a user wanting PNG images may list *image/png* as the only acceptable MIME type. A user preferring PNG images but willing to have JPEG images would list *image/png*, and *image/jpeg* in that order. A user preferring PNG images but willing to accept any image type would list *image/png*, *image/**. If the user doesn't specify a MIME type when creating the query, the software generates a default list of acceptable MIME types, namely */*, meaning that any type is acceptable. Sophisticated product servers can convert between data types. One mechanism for handling interoperability of legacy data systems is to deploy product servers that convert between file formats that are native to the local data system and the common data formats supported by the larger data system.

4.3 Query Servers

Query servers manage queries across distributed resources and are the point of entry into an OODT software framework installation. Query servers contain the algorithms necessary to traverse the network of profile and product servers, executing queries at appropriate servers and gathering results. In this manner query servers can simplify the interaction with the user, who is freed from the knowledge of accessing the remote interfaces of profile servers and product servers. Users instead call upon a query server for all profile and product interaction. The implementation of search algorithms for a query server is flexible allowing a broad range of possibilities including using a simplistic algorithm in the query server while implementing a more complex algorithm in the client application for testing. This is the case for this paper and the domain search algorithm is explained in section 5.1.

The OODT software framework supports several different interfaces to the query service to ensure that it is cross platform and supports cross language interoperability. This includes not only interfaces for programming languages such as Java, but interfaces using the web standard HTTP.

OODT queries (NASA Jet Propulsion Laboratory, 2005) are transported in an XML structure that provides pertinent system information, a results buffer, and the query expression in a parsed form (prefix notation). Since the parsed form of the query is transported, different query expression languages can be used as long as the parsed results can be expressed in the query structure. The currently implemented language, the Distributed Inventory System (DIS) expression language, is parameter/value based and allows the relational operators, logical connectors, and the grouping of expression using parentheses. The DIS query expression language meets the profile server requirement of being able to match on any attribute in the resource profile. Some simple example query expressions are shown in Figure 3.

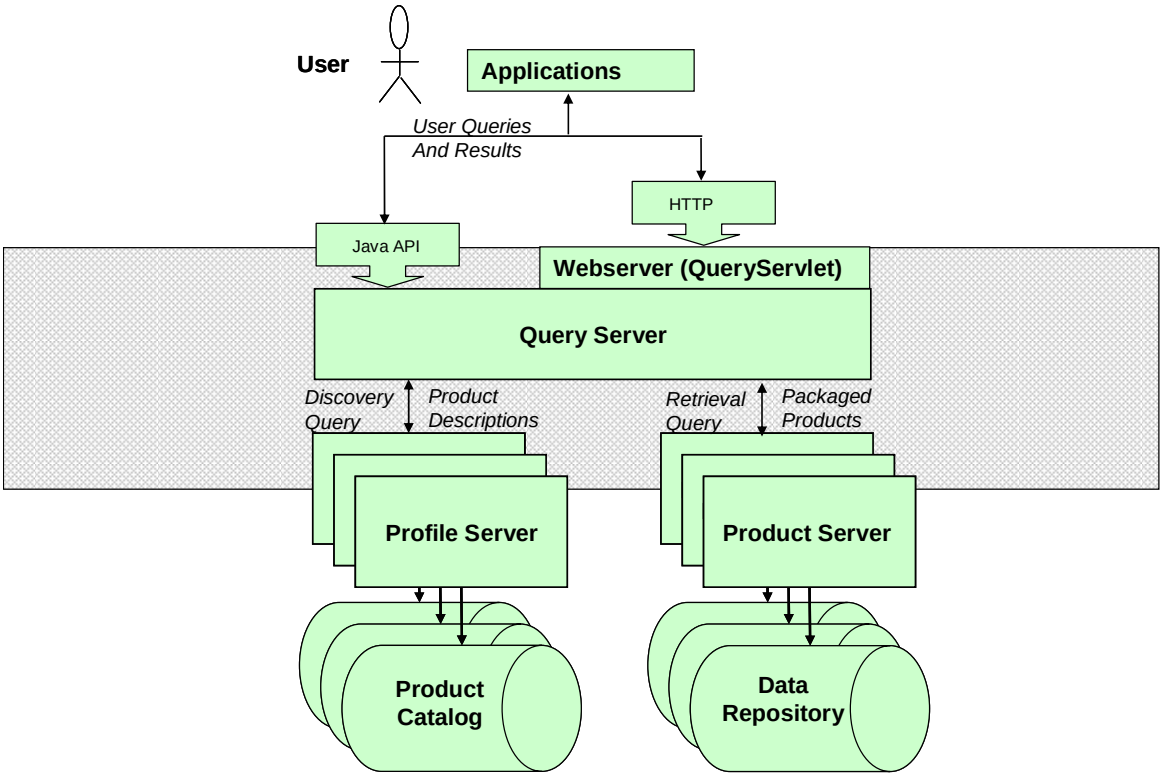

Figure 4. Software Framework Component Architecture

The OODT software framework components are illustrated in Figure 4.

5 CASE STUDY – NASA'S PLANETARY DATA SYSTEM

The Planetary Data System (PDS) is the official science data archive for NASA's planetary science community. As such, it contains tens of terabytes of data collected from over thirty years of solar system exploration and is growing exponentially. At its inception in the late 1980's, the PDS developed a data model that guides the capture of the information necessary to describe the data and ensure that the data remain scientifically useful for future scientists. Collected and validated using the data model, this information and the science data was submitted to peer review and then distributed to funded scientists in the planetary science community on CD and DVD media. The data model and much of the content of the catalog has recently been imported into an ontology tool[1] to provide easier management, better documentation, and to support the development of semantic web applications.

The combination of several factors including the advent of the Internet, requests to support correlative search across instruments, and huge increases in the volume of data returned from missions necessitated the development of an online system that supports search and retrieval of data products from across the

[1] Protégé, http://protege.stanford.edu

distributed heterogeneous data repositories of the PDS. In October of 2002, the PDS released the first version of its online distribution system to support the 2001 Mars Odyssey mission. Using the OODT framework and product servers at each distributed data repository, data products were available to planetary scientists as soon as they were released from the mission. Since then the distribution system has been augmented to include the majority of all data products in the PDS archive. The current system provides a two level search capability, first at the data set level (collections of data products) and then a data product search. Development is now focused on a single level search of data products across the archive using a hierarchical configuration of OODT profile servers.

The key to the success of the PDS search capability is the fact that each data product in the archive can be associated with identification, descriptive, and contextual information obtained either directly from class attributes or inferred through relationships. This information whether it is detailed information packaged with the data product or more general information maintained in a common repository is extracted and represented in a single data product profile.

5.1 Search Algorithm

The global product search capability is designed to have a single point-of-entry interface and a search scope that encompasses the entire geographically distributed PDS archive and its thousands of product types and millions of data products. To implement this search, a hierarchical configuration of profile servers and resource profiles has been implemented. At the leaf nodes of the hierarchy, each data product is represented by a single resource profile. For example, Figure 5 above illustrated a portion of a profile for a Galileo, Solid State Imaging System (SSI) image product where the domain attributes, *target_name*, *filter_name*, and *exposure_duration* have been encoded into the profile elements section. The resource attributes *Identifier* and *Title* have been set to the value of the concatenated domain attributes *data_set_id* and *image_id*, and the profile *Identifier* has been set to a unique object id, generated by the system.

```
<profile>
  <profAttributes>
    <profId>1.3.6.1.4.1.1306.2.1148</profId>
    <profType>profile</profType>
    <profStatusId>active</profStatusId>
  </profAttributes>
  <resAttributes>
    <Identifier>GO-J/JSA-SSI-2-REDR-V1.0</Identifier>
    <Title>GO-J/JSA-SSI-2-REDR-V1.0</Title>
    <Format>image/pds</Format>
    <resContext>NASA.PDS</resContext>
    <resClass>system.profileServer.dataSet</resClass>
    <resLocation>http://starbrite.jpl.nasa.gov/servlet/
  </resAttributes>
  <profElement>
    <elemName>TARGET_NAME</elemName>
    <elemType>CHARACTER</elemType>
    <elemValue>GANYMEDE</elemValue>
    <elemValue>IO</elemValue>
    <elemValue>JUPITER</elemValue>
    <elemValue>...</elemValue>
  </profElement>
  <profElement>
    <elemName>INSTRUMENT_ID</elemName>
    <elemType>CHARACTER</elemType>
    <elemValue>SSI</elemValue>
  </profElement>
  <profElement>
    <elemName>FILTER_NAME</elemName>
    <elemType>CHARACTER</elemType>
    <elemValue>CLEAR</elemValue>
    <elemValue>GREEN</elemValue>
    <elemValue>RED</elemValue>
    <elemValue>...</elemValue>
  </profElement>
  <profElement>
    <elemName>EXPOSURE_DURATION</elemName>
    <elemType>REAL</elemType>
    <elemMinValue>0.0</elemMinValue>
    <elemMaxValue>51195.8</elemMaxValue>
  </profElement>
</profile>
```

Figure 5. Data Set Profile Example

```
<profile>
 <profAttributes>
  <profId>1.3.6.1.4.1.1306.2.9999</profId>
  <profType>profile</profType>
  <profStatusId>active</profStatusId>
 </profAttributes>
 <resAttributes>
  <Identifier>ALL-ALL-ALL-N-ALL-V1.0</Identifier>
  <Title>ALL-ALL-ALL-N-ALL-V1.0</Title>
  <Format>text/xml</Format>
  <resContext>NASA.PDS</resContext>
  <resClass>system.profileServer.all</resClass>
  <resLocation>http://starbrite.jpl.nasa.gov/servlet/profileprofile...
 </resAttributes>
 <profElement>
  <elemName>TARGET_NAME</elemName>
  <elemType>CHARACTER</elemType>
  <elemValue>GANYMEDE</elemValue>
  <elemValue>IO</elemValue>
  <elemValue>JUPITER</elemValue>
  <elemValue>VENUS</elemValue>
  <elemValue>...</elemValue>
 </profElement>
 <profElement>
  <elemName>INSTRUMENT_ID</elemName>
  <elemType>CHARACTER</elemType>
  <elemValue>SSI</elemValue>
  <elemValue>NIMS</elemValue>
  <elemValue>SAR</elemValue>
  <elemValue>...</elemValue>
 </profElement>
</profile>
```

Figure 6. Root Profile Example

Aggregating the data product profiles of a single product type produces a product type profile. This product type profile describes the collection of products using the same set of attributes but with aggregated attribute values. In general attributes with discrete values such as *target_name* result in value lists. Numeric attributes such as *exposure_duration* result in value ranges. Performing this aggregation for each product type in the archive produces the second level of the hierarchy. For brevity we assume that each data set only contains one data product making the product type profiles equivalent to the data set profiles. For the PDS this results in several thousand data sets or level two profiles. Figure 5 illustrates a portion of the Galileo image data set profile. This data set contains the imaging data product mentioned above.

Data set profiles are in turn aggregated to create the *root* profile. The root profile describes all data sets and so also describes the entire archive. The root profile as a single, all inclusive description of the archive is suitable as the starting point for searches initiated from a single-point of entry user interface. The root

profile in Figure 6 contains the values for *target_name* and *instrument_id*, the result of aggregating the attribute values in the data set profiles, after aggregating the attribute values in the data product profiles.

Each resource profile includes a resource location that provides a link to the resource. A resource location in a *data product profile* will provide a URI for a data product, such as a URL to a data product server that will retrieve the product when invoked. A resource location for a *data set profile* provides a link to the profile server that serves the associated data product profiles. Similarly all higher level aggregated profiles provide links to profile servers that serve their associated lower level profiles. For the initial PDS prototype, the root profile and all data set profiles are served by a single profile server. All product level profiles are served by a two or three product level profile servers. These were configured based on a set of performance and maintenance requirements (the description of which is outside the scope of this paper).

The hierarchy of profiles is served by a combination of root and leaf profile servers (described earlier in this section). For example, a search application initially queries the profile server containing the root profile and builds a query interface using the attributes and their values in the root profile as query constraint choices. After the user has selected his query constraints, the location of the data set profile server provided in the root profile is used to query for all data set profiles matching the query constraints. The search application aggregates the matching data set profiles and builds a second query interface using the aggregated attributes and attribute values as query constraint choices. The cycle continues down the hierarchy of profile servers until product profiles are displayed to the user. The user is able to read the product description and subsequently accesses the data product through the location provided in the data product profile.

As a PDS example, the root profile contains “Galileo” as a value of the attribute *Mission_Name*. The selection of “Galileo” in the root interface would match on all Galileo data set profiles in the data set profile server. The second user interface produced by the search application would include the Galileo image attribute *Filter_Name* and its permissible values, including RED. A subsequent selection of RED for *Filter_Name* and a request to return all profiles would return all Galileo imaging data product profiles that were taken through the RED filter.

6 CONCLUSION

The Object Oriented Data Technology (OODT) task has developed a standard resource description scheme that can be used across domains to describe any resource. It uses a small set of generally accepted, broad scope descriptors while also providing a mechanism for the inclusion of domain specific descriptors. The use of this resource description scheme in an OODT software framework of profile and product servers provides a powerful, flexible, and scalable resource discovery capability across distributed resource repositories. Intelligent resource discovery is provided by the use of domain ontologies, relationship inference, and semantically homogeneous resource descriptions. In this paper, we described the resource description scheme in detail, and illustrated its utility on a case study: NASA’s Planetary Data System (PDS) project. PDS is implementing intelligent resource discovery across its distributed data repositories, thousands of data product types, and millions of individual data products using the OODT framework and a hierarchy of resource profiles created from a planetary science ontology and both explicit and inferred information.

7 REFERENCES

Aloisio, G., Cafaro, M., Epicoco, I., Fiore, S., Lezzi, D., Mirto, M. & Mocavero, S., (2005) Resource and Service Discovery in the iGrid Information Service. *International Conference on Computational Science and its Applications, Singapore.*

Beckwidth, R., Fellbaum, C., Gross, D., Miller, K., Miller, G. A., Tengi, R. (1990) Five Papers on WordNet. *Special Issue of the International Journal of Lexicography*, *3*(4), 235-312

Chervenak, A., Deelman, E., Foster, I., Guy, L., Hoschek, W., Iamnitchi, A., Kesselman, C., Kunszt, P., Ripeanu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K. & Tierney, B., (2002) Giggle: A Framework for Constructing Scalable Replica Location Services. *IEEE Supercomputing Conference*, Baltimore, MD, USA..

Crichton, D., Downing, G., Hughes, J.S., Kincaid, H. & Srivistava, S., (2001) An Interoperable Data Architecture for Data Exchange in a Biomedical Research Network. *14th IEEE Symposium on Computer-Based Medical Systems*, Bethesda, MD, USA..

Crichton, D., Hughes, J.S., Kelly, S. & Hyon, J., (2000) Science Search and Retrieval using XML. *Second National Conference on Scientific and Technical Data*, Washington, D.C., USA.

Crichton, D., Hughes, S., Kelly, S. & Ramirez, P., (2002) A Component Framework Supporting Peer Services for Space Data Management. *IEEE Aerospace Conference*, Big Sky, MT, USA..

Crichton, D.J., Hughes, J.S. & Kelly, S., (2003), Wu, Xiong, & Shekhar, (Eds), A Science Data System Architecture for Information Retrieval. *Clustering and Information Retrieval*, Kluwer Academic Publishers, Norwell, MA, USA. 261-298.

Czajkowski, K., Ferguson, D.F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S. & Vambenepe, W., (2004) The WS-Resource Framework. Retrieved June 11, 2005 from the Globus Alliance website: http://www.globus.org/wsrf/specs/ws-wsrf.pdf.

Dublin Core Metadata Initiative (1999) DCMI Metadata Terms, Retrieved April 15, 2005 from the Dublin Core Metadata Initiative website: http://dublincore.org/documents/dcmi-terms/.

Decker, S., Tangmunarunkit, H. & Kesselman, C., (2003) Ontology-based Resource Matching. *2nd International Semantic Web Conference*, Sanibel Island, FL, USA..

Foster, I., Kesselman, C., Nick, J.M. & Tuecke, S., (2003) The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, *Grid Computing*, 217-249.

Hull, R. & King, R., (1987) Semantic Database Modeling: Survey, Application and Research Issues. *ACM Computing Surveys*, *19* (3), 201-260.

IEEE-LTSC (2005) WG12: Learning Object Metadata (LOM). Retrieved June 15, 2005 from the IEEE - LTSC website: http://ltsc.ieee.org/wg12/.

ISO/IEC-11179 (1999) Information Technology, Metadata Registries (MDR). Retrieved April 15, 2005 from the ISO/IEC JTC1 SC32 WG2 Development/Maintenance website: http://metadata-standards.org/11179/.

Kesselman, C., Foster, I. & Tuecke, S., (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputing Applications*, 15 (3). 200-222.

Lassila, O. & Swick, R.R., (1999) Resource description framework (rdf) model and syntax specification, W3C.

Library of Congress (2005) METS Metadata Encoding and Transmission Standard. Retrieved June 15, 2005 from the Library of Congress website: http://www.loc.gov/standards/mets/.

Miller, G., Beckwith, R., Felbaum, C., Gross, D. & Miller, K., (1990) Five papers on WordNet, Cognitive Science Laboratory, Princeton University, Princeton, USA.

NASA Jet Propulsion Laboratory (2005) PDS Client v.1.0.0 for the Planetary Data System. Retrieved June 11, 2005 from the NASA Jet Propulsion Laboratory website: http://oodt.jpl.nasa.gov/pds-client/pds-client.pdf.

Pouchard, L., Cinquini, L. & Strand, G., (2003) The Earth System Grid Discovery and Semantic Web Technologies. *ISWC 2003 Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, Sanibel Island, FL, USA..

Raskin, R., Pan, M.J. & Mattmann, C.A., (2004) Enabling Semantic Interoperability for Earth Science Data. *4th NASA Earth Science Technology Conference*, Palo Alto, CA, USA..

Sangpachatanaruk, C. & Znati, T., (2005) A P2P Overlay Architecture for Personalized Resource Discovery, Access and Sharing over the Internet. *IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, USA..

Singh, G., Bharathi, S., Chrevenak, A., Deelman, E., Kesselman, C., Manohar, M., Patil, S. & Pearlman, L., (2003) A Metadata Catalog Service for Data-Intensive Applications. *IEEE International Conference on Supercomputing*, Phoenix, AZ, USA..

Smith, M., Bass, M., McClellan, G., Tansley, R., Barton, M., Branschofsky, M., Stuve, D. & Walker, J.H., (2003) DSpace: An Open Source Dynamic Digital Repository. *D-Lib Magazine*, *9* (1).

Sneed, H.M., (1997) The Rationale for Software Wrapping. in *International Conference on Software Maintenance*, Bari, Italy.