PaN-data ODI

Deliverable D4.1

Requirements analysis for common data catalogue

Grant Agreement Number	RI-283556
Project Title	PaN-data Open Data Infrastructure
Title of Deliverable	Requirements analysis for common data catalogue
Deliverable Number	D4.1
Lead Beneficiary	STFC
Deliverable Dissemination Level	Public
Deliverable Nature	Report
Contractual Delivery Date	01 July 2012 (Month 9)
Actual Delivery Date	20 July 2012

The PaN-data ODI project is partly funded by the European Commission under the 7th Framework Programme, Information Society Technologies, Research Infrastructures.

Abstract

This document i) defines a set of criteria for evaluating data catalogue systems, ii) classifies the PaNdata-ODI requirements as set in D5.1 according to these criteria, iii) evaluates ICAT based on the requirements, iv) surveys a number of systems that provide data catalogue functionality, and v) discusses metadata standardisation issues.

Keyword list

Data catalogue, metadata, data management

Document approval

Approved for submission to EC by all partners on...

Revision history

Issue	Author(s)	Date	Description
0.6	Milan Prica, George Kourousias	26 June 2012	First version
	Idrissou Chado		Description of Soleil tools
0.7	Alistair Mills	2 July 2012	ICAT description and corrections
0.8	Thorsten Kracht	9 July 2012	Corrections
0.9	Brian Matthews	17 July 2012	Corrections and additional contribution
1.0		18 July 2012	Final version

Table of contents

Page

1 INTRODUCTION 4
2 EVALUATION CRITERIA FOR SCIENTIFIC DATA CATALOGUES 5
2.1 AUTHENTICATION SYSTEM 5
2.2 METADATA MODEL 5
2.3 QUERYING/SEARCHING METHODS 5
2.4 SOFTWARE REQUIREMENTS 5
2.5 USER INTERACTION 5
2.6 SERVICE API 6
2.7 HARDWARE REQUIREMENTS 6
2.8 DOCUMENTATION 6
2.9 SUPPORT 6
2.10 LICENCE 6
2.11 DATA POLICIES 6
2.12 TOTAL COST OF OWNERSHIP (TCO) 7
2.13 SCALABILITY AND PERFORMANCE 7
2.14 DATA INGESTION 7
2.15 ADDITIONAL SERVICES AND PLUG-INS 7
2.16 DEPLOYMENT ARCHITECTURE 7
2.17 MAINTENANCE & SUSTAINABILITY 8
2.18 SPECIALISATION AND SYSTEMS SCOPE 8
3 REQUIREMENTS FOR A DATA CATALOGUE SERVICE 9
4 SURVEY OF DATA CATALOGUE SYSTEMS 11
4.1 ICAT 11
4.2 OTHER SYSTEMS WITH DATA CATALOGUING FUNCTIONALITY 13
4.2.1 DSpace 13
4.2.2 Fedora Commons Repository Software 13
4.2.3 iRODS 14
4.2.4 SRB / MCAT 15
4.2.5 Tardis 16
4.2.6 AMGA 17
4.2.7 Artemis 17
4.2.8 GCube Framework 18
4.2.9 ISPyB (ESRF) 18
4.2.10 Twist (Synchrotron Soleil) 19
4.2.11 Zentity (MS Research) 19
5 ICAT AGAINST THE DCS REQUIREMENTS 20
6 METADATA ELEMENTS, KEYWORDS, AND DATA FORMATS 22
7 SUMMARY 27

1 Introduction

The PaNdata consortium brings together thirteen major European research infrastructures to create an integrated information infrastructure supporting the scientific process. PaNdata-ODI will develop, deploy and operate an Open Data Infrastructure across the participating facilities with user and data services which support the tracing of provenance of data, preservation, and scalability through parallel access. It will be instantiated through three virtual laboratories supporting powder diffraction, small angle scattering and tomography.

The project aims at the standardisation and integration of the consortium's research infrastructures in order to establish a common and traceable pipeline for the scientific process from scientists, through facilities to publications. At the core there is a series of federated data catalogues which allow scientists to perform cross-facility, cross-discipline interaction with experimental and derived data at high speeds. This will also deliver a common data management experience for scientists using the participating infrastructures particularly fostering the multi-disciplinary exploitation of the complementary experiments provided by neutron and photon sources.

By a **Data Catalogue** we mean a systematic record of the data generated within facilities including information on the context in which the data was collected, and information on how the data itself is stored. The contextual information would typically contain information on the experiment or other process (e.g. simulation, data analysis) which was undertaken to generate the data, such as information on the experimental team, the instrument, the sample and the parameters set. The data storage information would contain information on where the data set are located and organizes, together with information and controls on how to access the data.

In the next sections a set of criteria for the evaluation of data catalogue systems is defined. The focus is on cataloguing scientific data and especially for those produced at the research institutes of the PaNdata consortium members. A separate PaNdata-ODI activity related to virtual laboratories (WP5) has documented a number of requirements (D5.1) that in this document are classified against the evaluation criteria. After critically examining ICAT, the data catalogue system of choice in PaNdata, we present a survey of systems that provide similar functionalities. Finally, the issue of common metadata is presented in contrast to a standard set of elements (Dublin Core$^{®}$) and that of a scientific data format (NeXus).

2 Evaluation Criteria for Scientific Data Catalogues

In order to analyse, examine and evaluate a data catalogue system we identify a set of criteria suitable for accessing the value of a data cataloguing system within facilities. Many of these criteria have been explicitly requested from the end-user scientific communities as presented in D5.1. Additional criteria will also be mentioned. The weight and necessity of each varies. A final subset will form the requirements for the Data Catalogue service of this project.

2.1 Authentication System

The Data Catalogue Systems (DCS) are mostly multi-user thus there is the need for user authentication or even full AAA services. In distributed deployments, that permit cross-facility operations, support for a federated identity solution may be necessary.

2.2 Metadata Model

A metadata model is appropriate to support the publication and sharing of the categories of information collected within the catalogue, and made available for searching and access. Generic and standards models (i.e. Dublin Core) tend to be brief thus cannot contain the information required from specialised scientific applications. On the other hand specialised models are suitable for very small class of problems/applications. A data catalogue system may permit multiple models or expandable ones.

2.3 Querying/Searching Methods

One of the main reasons for storing and indexing metadata and associating them with the data, is to provide the end-user with advanced search/query services. Such a service may permit: 1) free-text searches, 2) standard hierarchical keywords based on common metadata models, 3) Tags and Tag clouds for non-hierarchical keywords/terms, 4) numerical ones through knowledge representation, 5) wildcards and 6) query logic.

2.4 Software Requirements

In a complete data catalogue solution there will be software requirements on both the server and DB system side and also for the client side/end-user terminal. Such requirements are mostly on the Operating System (i.e. Windows/Linux), dependent services and technologies (i.e. Oracle/PostgreSQL/MySQL, Java, .Net), and web-browser plug-ins (i.e. Java, Adobe Flash, MS Silverlight). A suitable data cataloguing system should clearly state its dependencies on such systems and it would be preferable if they could support commonly used platforms.

2.5 User Interaction

The core of a data catalogue solution is a database system provided usually as a server. Nevertheless the end-user interaction is of high importance thus the data catalogue may provide the means for end-user interaction. This could be achieved through standalone GUI applications, command line/console programs, web portals, and integration with existing systems (i.e. integration with the institute's ERP). The data catalogue should provide options for user interaction.

2.6 Service API

The presence of an application programming interface (API) is considered to be an advantage and of high importance. A flexible API permits programmatic interaction with the data catalogue and the integration of the system to existing ones. This requires stable API bindings for popular programming languages (e.g. Python, C++, Java, C#) and potentially with domain specific technologies (i.e. high level numerical computing environments like Matlab or IDL and distributed control systems such as Tango and EPICS).

2.7 Hardware Requirements

A standalone data catalogue software may require just a personal computer for the use of individual or small teams. However, within a facilities context, our focus is on enterprise-level data catalogues which can be deployed in server-class hardware with large memory, multi-CPUs, and high bandwidth data networks, supported within a dedicated systems team. This should allow automation, high-availability and high-throughput, as well as accessibility by a large number of users.

2.8 Documentation

Documentation should be available for the conceptual part of the system (i.e. architecture and metadata model), the API (including code samples), and the front-end system (i.e. web-portal). In case the software is open source, properly commented code may assist further developments. Further, there should be help and tutorial material available.

2.9 Support

A data catalogue system provider may provide support to the entities (i.e. an institute) that will adopt it. The end-users on the other hand may receive support from local specialists/helpdesk familiar with any customisations that the local deployment may have. In practice the end user needs technical support by the experts of the facility while these experts may need to rely on the support from the data catalogue system developer.

2.10 Licence

The data catalogue should have a clear licence. Either a proprietary licence, or a free and open source licence. A popular copyleft licence is the GNU General Public Licence (GPL). Other notable permissive free software licences are the MIT licence and BSD licences. Proprietary systems are often licenced per single user (named user, client, node). In general open source licences permit easier adoption of the data catalogue to specialised requirements through heavy customisation or code branching, and would be preferred.

2.11 Data Policies

The catalogued data may need to comply with a specific data policy required by the facility. An enforced data policy often defines explicitly or implicitly the constraints that regard ownership, access and time-periods for restricted access. A data catalogue may need to integrate the policy and even be policy-aware, enforcing policies (e.g. When the policy is the PaNdata Europe, change ownership to Public after 3 years without user intervention).

2.12 Total Cost of Ownership (TCO)

A formal determination of the Total Cost of Ownership (TCO) for a data catalogue may include: expertise cost, maintenance, support, and licences. In addition to those, factors as Competitive Advantage (i.e. a better catalogue than that of the competition) or Compliance to Standards (i.e. using a common one among similar institutes) are very important and affect the TCO for a facility.

2.13 Scalability and Performance

The amount of produced data in the experimental science field, both raw and processed, is growing exponentially over time. This growth makes the scalability and performance characteristics of a data catalogue system of crucial importance. Often the design decisions (i.e. reliance on a powerful enterprise-level database) for a data catalogue system take into account the forecasted data flow.

2.14 Data Ingestion

Depending on the way that data/metadata are ingested, the system may be suitable or not for certain use cases. For single-user standalone software, a manual method for ingesting data/metadata to the system may be adequate but for enterprise-level multi-user catalogues other methods are necessary. A suitable API should permit automated ingestion as part of the data workflow. A typical scenario is that the scientific data are stored in a format (e.g. NeXus) and automatically parsed and ingested into the data catalogue using a common set of metadata.

2.15 Additional Services and Plug-ins

A criterion of evaluation to consider is the expandability of the data catalogue system. The system can be modular and permit plug-ins or add-ons in such a way that would allow the accommodation of additional services. Such services may include workflow systems and data provenance functionality. The cataloguing phase of ingestion, querying and retrieval can be inserted in any stage of the workflow. Data Provenance procedure could interact at any stage of the cataloguing adding extra metadata. The frond-end could be enhanced if required to allow data downloads and uploads.

2.16 Deployment architecture

The deployment architecture and the technologies involved within a DCS can vary. Different DCS deployments can involve: 1) standalone for single-user, 2) multi-user as typical server-client, 3) utilisation of Cloud technologies, 4) reliance on distributed DBs, 5) multiple DCS instances, and 6) being part of greater Grid computing platform.

2.17 Maintenance & Sustainability

Maintenance and sustainability should be examined in the case of large scale DCS deployment. This should indicate whether the system has stable version, debugging/troubleshooting processes, future developments roadmap, quality assurance procedures, and follows software sustainability practices. An active open-source development community is a good indicator of a sustainable model.

2.18 Specialisation and Systems Scope

A data catalogue may be a component of a larger system (i.e. a Grid computing platform) or a specialisation of a more exhaustive system. Its scope may be a generic catalogue of any information type, partially specialised for a domain (i.e. scientific data) or built-up around a focused application i.e. (data for the Protein Data Bank)

These eighteen evaluation criteria are clearly related within each other, with the extant of the support for some factors influencing others. We give a view on the relationships between the criteria in Figure 1.

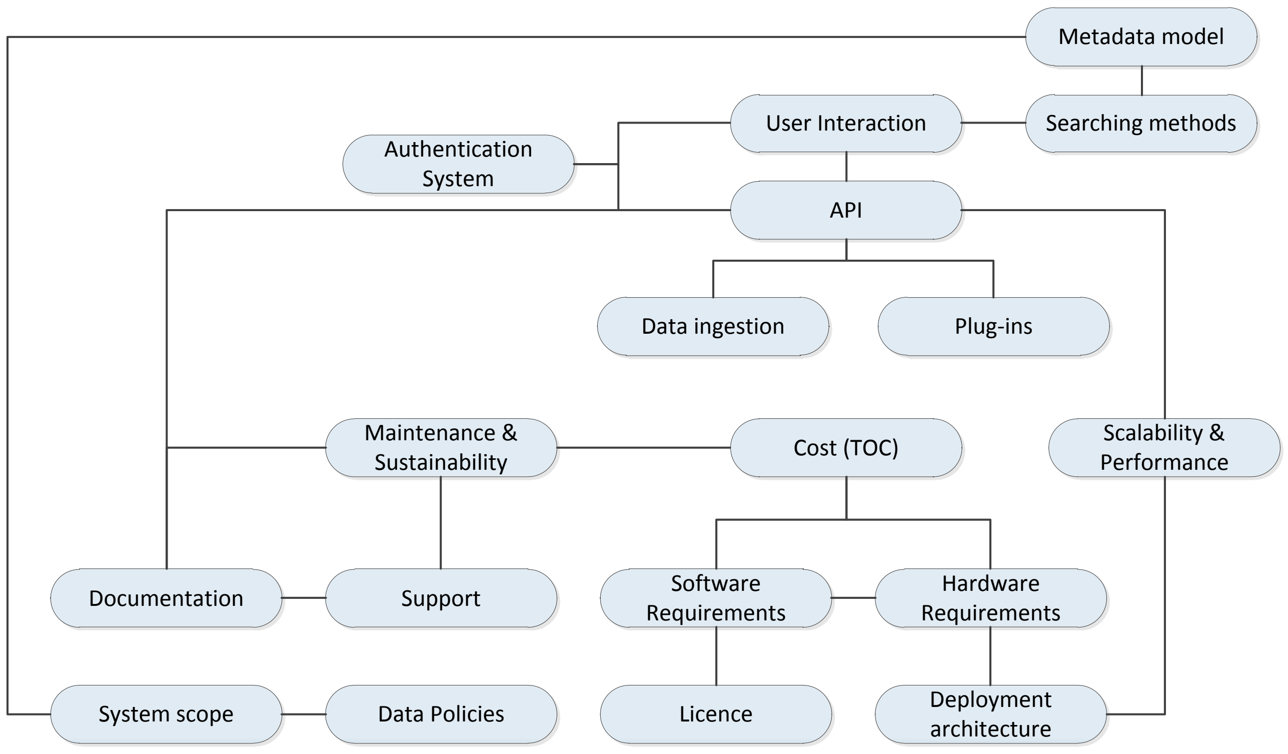

Figure 1 Criteria relationships for Data Catalogue System

3 Requirements for a Data Catalogue Service

The PaNdata consortium has defined a main list of requirements for the data catalogue service. This is focused on satisfying specific needs of the scientific institutions they represent and has been formally proposed in deliverable D5.1 after detailed analysis of the requirements of the Virtual labs. Additionally, we have analysed the wider context of facilities within the PaNdata consortium to determine some additional requirements.

We formulate the D5.1 requirements (Table 1) plus additional ones (Table 2) based on the previous list of criteria.

Authentication System	Should enable DCS operations across PaNdata facilities. Should be easily integrated with the existing local authorisation and file systems.
Metadata model	Should be extendible and support discipline specific annotations. Unique identifiers to allow for citation (DOIs).
Querying/Searching methods	Should permit keyword based searching.
Software Requirements	The DCS should be based on a widely-used DB.
User Interaction	Should provide the end-user with a web-portal.
Service API	An API that permits at least all the operations that can be done through the portal.
Hardware Requirements	N/A
Documentation	Documentation and user-guide for the API and the web-portal.
Support	N/A
Licence	N/A
Data Policies	N/A
Total Cost of Ownership (TCO)	N/A
Scalability and Performance	Fast enough to permit interactive work (web-portal)
Data ingestion	N/A
Additional Services and Plug-ins	Upload/download/delete of files, change permissions (through the web portal). Support for a Workflow system.
Deployment architecture	N/A
Maintenance & Sustainability	N/A
Specialisation and Systems scope	Virtual labs as described in D5.1 and scientific applications related to the participating institutes.

Table 1 WP5 requirements for data catalogue service based on the evaluation criteria.

Authentication System	Easy integration with the Umbrella system as described in D3.1.
Metadata model	A model that covers the metadata present in an advanced scientific data file format (NeXus).
Querying/Searching methods	Querying value ranges. Use of Tags.
Software Requirements	Server-class Linux OS and reliance on open source solutions.
User Interaction	As in D5.1.
Service API	API bindings for languages as C++, Java, and Python. Ideally, bindings for high level numerical computing environments (Matlab/IDL) so that the users can query the system and index post-processed data too.
Hardware Requirements	Server-class systems.
Documentation	As in D5.1.
Support	Support for both usage and development from the system's development team.
Licence	Open source.
Data Policies	It should not conflict with the PaNdata Europe Data Policy. Ideally should permit automated policy enforcement actions like changing the access of certain data to Public after a predefined time.
Total Cost of Ownership (TCO)	Follow the Compliance to Standards (i.e. using a common system among similar institutes). Pay no fees for support and proprietary software. No high maintenance costs after the end of the project.
Scalability and Performance	Should scale well when additional hardware is added and cope with the data flow of each member facility.
Data ingestion	Sample code for data ingestion through the API.
Additional Services and Plug-ins	Possible interaction with workflow systems (e.g. Taverna, Kepler) and data provenance frameworks.
Deployment architecture	Should permit multiple instances (distributed across the institutions) able to return cross-facility search results.
Maintenance & Sustainability	The system development team should have high maintenance and software sustainability standards.
Specialisation and Systems scope	Software specialised on scientific data without additional technologies like Grid frameworks and complicated certificate based systems.

Table 2 Additional requirements for data catalogue service based on the evaluation criteria.

4 Survey of Data Catalogue Systems

In this section we provide an overview of ICAT; the data catalogue system already in use by a number of PaNdata partners and that is critically evaluated against the above criteria. For purposes of comparison, we also present a number of systems that offer data catalogue services, and discuss them in regard to the above criteria.

4.1 ICAT

ICAT is a metadata cataloguing system which is has been deployed in several of the laboratories of the partners in PaNdata. The initial development was carried out in the years of 2002-2008 and production deployments started in 2008 at the ISIS Neutron facility in the UK. The number of facilities deploying ICAT is increasing at the rate of one per annum and some of the facilities are using ICAT for more than one infrastructure. ICAT has active support and development.

ICAT supports three authentication systems. It supports a simple username and password pair which are checked against a local database. It also supports the use of an LDAP authentication server using username and encrypted password. ICAT also supports the use of digital certificates.

ICAT supports the use of standard metadata models such as Dublin core. In addition it supports a flexible metadata model which does not have to be fully defined when the system is installed. This allows ICAT to expand the data types and names during the life of data catalogue.

ICAT provides a rich set of search methods in the data catalogue including free text search, numerical and string search, query logic. There is a SQL like syntax for expressing search operations and the representation of the results of the search.

The ICAT software provides an API against which applications can provide user clients. Several clients have been written to provide particular behaviour such as data ingestion, catalogue browsing, data retrieval and catalogue searching. The ICAT software is written in Java and runs in an application server such as Glassfish. This environment is supported on Microsoft Windows (all versions), Linux (all versions), Apple Macintosh, and other common systems. ICAT supports any database supporting the Java persistence architecture and this includes Oracle, PostgreSQL and MySQL. The web clients for ICAT support many web browser environments by using the Google Web Toolkit to interface the application to the web browser.

User interaction with ICAT is supported by the ICAT clients. There are clients for command line scripting, GUI and web browsers. It is possible to write a client to interface ICAT to an institute's enterprise resource planning (ERP) system.

The API provided by ICAT supports Java and Python applications. There are restrictions on the connection of the ICAT API directly to C++ and C#. However, in principle a small facade can provide a suitable interface which would allow a C++ and C# program to interact with ICAT in a manner similar to a Java program. The ICAT API provides a programmatic interaction with ICAT. In addition to interfacing ICAT to client applications, it allows the high level computing environment Mantid to use ICAT as a data source and as a destination for derived data.

The hardware requirements for ICAT depend on the deployment rather than on ICAT. It is possible to run ICAT on a single PC to manage a small personal catalogue. It is also possible to run ICAT in a clustered environment where a large number of transactions can be sustained concurrently.

There is a large collection of documents on ICAT and example programs showing how to program client applications. The documentation includes release and installation notes as well as descriptions of the architecture and metadata model.

Active support is provided to users of ICAT by the development team. At each of the institutions using ICAT, there is a local support person who is familiar with the local deployment.

The licence for ICAT is Free BSD.

Data policy is not normally enforced by ICAT. However data access is controlled by ICAT to restrict access to data to authorised users. This functionality is required to implement data policy.

An institution deploying ICAT can make an assessment of the total cost of ownership.

The Topcat software allows the federation of multiple ICAT deployments. This is very useful when the catalogues are distributed across many institutions. ICAT has deployments where the total volume of data is close to a Petabyte and others where the total volume is less than a Gigabyte. The experience is that the deployment can be matched to the size of the requirements and that scalability and performance are generally satisfactory.

Data ingestion into an ICAT is supported by a suitable ingestion client. A number of ingestion clients have been developed to meet the needs of the deployments. At present there is no client which provides easy ingestion of Nexus data, however it is not a difficult problem to create one. There are a number of additional plug-ins for ICAT. These provide authentication, data transport and catalogue browsing facilities. At present there is no plug-in for provenance or workflow but it is possible to create this.

The ICAT development project has been sustained for almost ten years and follows industry standard practices for software engineering. There is a program of software releases supported by quality assurance procedures. There are five institutions actively participating in the development and testing of releases. Continuous build and test procedures are used and the results are published of the internet.

ICAT can be deployed in a wide range of situations from a single user, isolated system to a widely distributed system. This is possible as it is deployed on the Java Enterprise Edition and this supports a wide range of deployment conditions.

While an independent component, the ICAT data catalogue is integrated by some facilities as part of the facility management system. It has been integrated with user and investigation management systems.

http://www.icatproject.org/

4.2 Other systems with data cataloguing functionality

4.2.1 DSpace

DSpace is free, open source software for building digital repositories that targets academic, non-profit, and commercial organizations with more than 1,000 installations worldwide. DSpace preserves and enables easy and open access to all types of digital content including text, images, moving images, mpegs and data sets. DSpace is an out-of-the-box repository software package for creating repositories focused on delivering digital content to end users, and providing a full set of tools for managing and preserving content within the application.

DSpace is a set of cooperating Java web applications and utility programs that maintain an asset store and an associated metadata store. The web applications provide interfaces for administration, deposit, ingest, search, and access. The asset store is maintained on a file system, or similar storage system. The metadata, including access and configuration information, is stored in a relational database. DSpace offers a full application, not just a framework with components (i.e. database, data model, workflows, browse/search, storage manager, front end web interface) built into the architecture. Components may be swapped or added, but there is no need to build new ones. DSpace comes packaged with Apache Lucene, an OS indexing engine that allows for enabling full-text searching for end users. In addition, you can optionally enable a faceted search/browse interface via Apache Solr, an OS enterprise search platform. DSpace auto-recognizes files of any common format (e.g. TXT, DOC, PDF, JPEG, MPEG, TIFF). But, DSpace will accept files of any format.

Qualified Dublin Core is the default metadata schema, but DSpace can accept custom metadata schemas similar to Qualified Dublin Core. DSpace can also translate metadata from other metadata schemas such as MARC/MODS. DSpace comes with management tools including batch import/ export, batch metadata editing, curation, and object backup & restoration tools. The platform comes with an authorization stack or organizations may use an existing LDAP, Shibboleth, or similar protocols to link their internal systems. DSpace allows you to control permissions as granular as item level, or you can set global permissions based on communities and collections. OAI-PMH/SWORD/WebDAV: DSpace complies with standard protocols for access, ingest and export. Organizations can choose either PostgreSQL or Oracle for the database in which DSpace manages items and metadata.

http://www.dspace.org

4.2.2 Fedora Commons Repository Software

The Flexible Extensible Digital Object Repository Architecture (Fedora) is a conceptual framework that uses a set of abstractions about digital information to provide the basis for software systems that can manage digital information. It provides the basis for ensuring long-term durability of the information, while making it directly available to be used in a variety of ways. Fedora provides a foundation upon which to build a variety of information management schemes for different use cases, not a full solution for a specific use case.

In a Fedora repository, all content is managed as data objects, each of which is composed of components ("datastreams") that contain either the content or metadata about it. Each datastream can be either managed directly by the repository or left in an external, web-accessible location to be delivered through the repository as needed. A data object can have any number of data and metadata components, mixing the managed and external datastreams in any pattern desired. Each object can assert relationships to any number of other objects, providing a way to represent complex information as a web of significant meaningful entities without restricting the parts to a single context.

Each data object is represented by an XML file that is managed in the file system, which contains information about how to find all of the components of the object, as well as important information needed to ensure its long-term durability. The system keeps an audit trail of actions that have affected the object; any formal policies may be asserted about the object and its use, and things like checksums, all within that XML file. As long as both the XML files and the content files that are managed by the repository are backed up properly, the entire running instance of the repository can be reconstructed from the XML files. There is no dependence upon any software to do so, no relational database that cannot be completely reconstructed from the files.

While Fedora can easily be used to model digital collections of surrogates of traditional, catalog-based collections, it has been designed to be able to support durable web-like information architectures. Because each object completely contains all of the content, metadata and attributes of a unit of content, and can assert any number of relationships to any other object, it is easy to support schemes in which objects have multiple contexts with no dependencies upon each other. A Fedora repository does not have a particular catalog. Any number of indices, designed for specific purposes can be applied to any pattern of components of objects desired.

http://www.fedora-commons.org/

4.2.3 iRODS

iRODS™, the Integrated Rule-Oriented Data System, is a data grid software system developed by the Data Intensive Cyber Environments research group (developers of the SRB, the Storage Resource Broker), and collaborators. The iRODS management policies - sets of assertions these communities make about their digital collections - are characterized in iRODS Rules and state information. At the iRODS core, a Rule Engine interprets the Rules to decide how the system is to respond to various requests and conditions. iRODS is open source under a BSD licence.

iRODS iCAT [1]Metadata Catalog stores state information and descriptive metadata in a database, enabling search, management, controlling and tracking of all data access and manipulation.
Rule Engine applies user-defined Policies and Rules to data to automate administrative tasks, enforce management policies, and evaluate assessment criteria, enabling large-scale collections.

iRODS interfaces (GUI, Web, WebDAV, command line) let users search for, access and view, add/extract metadata, annotate, analyze and process, manage, replicate, copy, share, repurpose,

[1] Not to be confused with STFC's ICAT.

control and track access, subscribe, and more. iRODS Server software and Rule Engine run on each data server. The iRODS iCAT Metadata Catalog uses a database to track metadata describing data and everything that happens to it. Logical namespace spans administrative domains, institutions, and international collaborations. Federation of multiple iRODS Data Grids enables flexible architectures: central archives, master-slave data grids, chained data grids, and deep archives, through independent but collaborating iCAT Metadata Catalogs. Policies can be added without modifying the core code. iRODS has a number of API Interfaces: C, Unix shell, Java, Python, Kepler, Taverna, Web. iRODS is interoperable with other data management systems e.g. Fedora and DSpace.

https://www.irods.org

4.2.4 SRB / MCAT

SRB – The DICE Storage Resource Broker – supports shared collections that can be distributed across multiple organizations and heterogeneous storage systems. The SRB can be used as a Data Grid Management System (DGMS) that provides a hierarchical logical namespace to manage the organization of data (usually files). The SRB software infrastructure can be used to enable Distributed Logical File Systems, Distributed Digital Libraries, Distributed Persistent Archives, and Virtual Object Ring Buffers. The most common usage of SRB is as a Distributed Logical File System (a synergy of database system concepts and file systems concepts) that provides a powerful solution to manage multi-organizational file system namespaces.

SRB presents the user with a single file hierarchy for data distributed across multiple storage systems. It has features to support the management, collaboration, controlled sharing, publication, replication, transfer, and preservation of distributed data. The SRB system is middleware in the sense that it is built on top of other major software packages (file systems, archives, real-time data sources, relational database management systems, etc). The SRB has callable library functions that can be utilized by higher level software. However, it is more complete than many middleware software systems as it implements a comprehensive distributed data management environment, including end-user client applications ranging from Web browsers to Java class libraries to Perl and Python load libraries.

The SRB software can be provided to educational and U.S. government agencies as per the copyright. iRODS, which is open source and freely available, provides much of the same functionality as SRB.

MCAT is a meta information catalog system implemented at SDSC as part of the Data Intensive Computing Environment (DICE) with requirements mainly based on the Storage Resource Broker system. MCAT catalog is designed to serve both a core-level of meta information and domain-dependent meta information. Data objects are first class objects in MCAT. Meta information or metadata about data objects can be internal to the object (i.e., it can be part or even whole of it) or external and formed separately from the data object. Internal metadata are mostly extracted, derived or mined where as external metadata are annotated or logged from external sources. Metadata about data objects describes the properties and attributes of the objects. Every data object is associated with a (data) collection. A collection, in MCAT, can be viewed as a set of data objects and other collections which are called sub-collections to the collection. The collection lattice is strictly hierarchical, and notions such as navigation through a collection, searching recursively through a collection hierarchy, etc are possible within MCAT. Every replica of a data object resides in a physical storage in a location in that storage. Each data object has an access control mechanism imposed on it through an access control list (ACL) which connects it to the user entity. Each user is given exactly one permission per data object. Each PermissionId has an associated list of actions that is permitted.

There is also a functionality of auditing in MCAT/SRB. Each action on a data object can be audited and the actions success or failure noted in an audit trail. The owner of a data object (or anyone with control permission on the object) can impose an audit on actions on a data object for specific users. Then whenever that user accesses the dataset the action is noted in the audit trail. The user can impose a restriction on the audit, by stating that he/she doesn't want to be audited. In such a case, an anonymous user audit trail is written.

Apart from obtaining a read/write permission on a data object a person can get a "ticket" on a data object. A ticket (which is a 10-character string internally generated) provides a holder with an action (currently only read) permit on the data object. A ticket giver can impose some restrictions on the ticket such as who can use it (can be a registered user, a registered user group, or any one in the world!), when (time-period) it can be used and how many times it can be used. A ticket can be issued at collections-level but allows a holder access only to data objects that are controlled by the ticketGiver. A collection-level ticket can be given with a recursive option allowing the holder to drill down the collection.

http://www.sdsc.edu/srb/index.php/Main_Page
http://www.sdsc.edu/srb/index.php/MCAT

4.2.5 Tardis

TARDIS (formerly an acronym for 'The Australian Repositories for Diffraction ImageS' was initially created for the storage and dissemination of public datasets containing raw crystallography data (known as 'diffraction images'), along with rich metadata for search and persistent handles for citation in publications. Storage was and remains federated, meaning the public index, TARDIS.edu.au contains no data itself and merely points to data stored in external labs and institutions. At the beginning of 2010, the Australian Synchrotron and VeRSI collaborated with Monash University to bring two new TARDIS-related products to life - Synchrotron-TARDIS and MyTARDIS. Synchrotron-TARDIS was designed specifically to work with internal synchrotron systems such as their authorisation system, data transfer mechanisms and metadata extraction software, collectively known under the heading of VBL (Virtual Beam Line) services. The result is a repository hosted at the Australian Synchrotron that allows anyone with a synchrotron login to access their macromolecular crystallography data collected off its instruments with the full capabilities available to TARDIS.edu.au.

By leveraging VBL services within the synchrotron, private data was able to be sent securely to instances of what's known as MyTARDIS, an institutional repository for the storage of private, raw data. Scientists can log into their university instance, download their data from locally-hosted storage, share data and search it at will. They also have the ability to make data public, with the bonus of the entries being consumed by ANDS' Research Data Australia. Recent contributions to the

MyTARDIS codebase have been towards increased support for more types of scientific data, easier ways of depositing data, advanced sharing, and the bringing together of the public TARDIS.edu.au and synchrotron-TARDIS into the common MyTARDIS codebase.

MyTARDIS supports 2 different XML schemas for importing metadata. One method is METS and the other is using a MyTARDIS specific XML format. METS is the preferred format because it is supported by systems other that MyTARDIS so will provide more versatility in the long run.
The Meta-data Encoding and Transmission Standard was recommended by Monash Librarians and ANU in 2008 as the XML description format for MyTARDIS datasets.

http://tardis.edu.au/
http://code.google.com/p/mytardis/
http://www.loc.gov/standards/mets/

4.2.6 AMGA

The ARDA Metadata Grid Application – AMGA – is a general purpose metadata catalogue developed by the EGEE project and part of the gLite middleware distribution. AMGA allows for metadata information to be attached to files stored on the Grid, where metadata can be any relationally organized data typically stored in a relational database system (RDBMS). In addition, the metadata in AMGA can also be stored independently of any associated files, which allows using AMGA as a general access tool to relational databases on the Grid. Data may be accessed with SQL92 or using a simple to learn metadata access language useful in smaller Grid applications. AMGA provides a possibility to replicate metadata between different AMGA instances allowing the federation of metadata when required and increase the scalability and improve the access times on a globally deployed Grid. AMGA features different authentication methods via (Grid-Proxy-) Certificates as well as flexible accesses control mechanisms for individual data items based on ACLs.
The AMGA implementation uses streaming (amgad) to communicate between client and server. To meet the EGEE requirements, an alternative SOAP-based frontend (mdsoapserver) exists. AMGA is distributed as a RPM including: A streaming and a SOAP front-end; An interactive client for the streaming front-end; Client APIs for the streaming client (C++, Java and Python). Additionally, a Perl API client module, a PHP API client and a web frontend are available.

http://amga.web.cern.ch/amga/
http://lcg.web.cern.ch/LCG/activities/arda/arda.html

4.2.7 Artemis

Artemis, a system developed to integrate distributed metadata catalogs on the grid. Artemis exploits several AI techniques including a query mediator, a query planning and execution system, ontologies and semantic web tools to model metadata attributes, and an intelligent user interface that guides users through these ontologies to formulate queries. Metadata catalogs, such as the Metadata Catalog Service (MCS) store the metadata about different objects in different data sources on the grid. Artemis utilizes data integration techniques in conjunction with an interactive query builder to enable the user to easily create complex queries to obtain objects that match user criteria.

http://www.isi.edu/~deelman/artemis_iaai04.pdf

4.2.8 GCube Framework

gCube is a large software framework designed to abstract over a variety of technologies belonging data, process and resource management on top of Grid/Cloud enabled middleware. By exposing them through a comprehensive and homogeneous set of APIs and services, it globally provides access to different storage back-ends in a variety of crucial areas for various purposes. Different storage layers offer facilities for handling: (a) multi-versioned software packages and dependencies resolution among them; (b) large scientific data-sets storable as tables (to be released in the up-coming minor release); (c) Time Series by offering an OLAP interface to operate over them; (e) structured document objects storable as trees of correlated information objects; (f) geo-coded da-tasets compliant with OGC-related standards; (g) and, finally, plain files.

gCube provides management of metadata in any format and schema that can be consumed by the same application in the same Virtual Organization and also the management of Virtual Research Environment (VRE). Through VREs, groups of users have controlled access to distributed data, services, storage, and computational resources integrated under a personalised interface. A VRE supports cooperative activities such as: metadata cleaning, enrichment, and transformation by ex-ploiting mapping schema, controlled vocabulary, thesauri, and ontology; processes refinement and show cases implementation (restricted to a set of users); data assessment (required to make data exploitable by VO members); expert users validation of products generated through data elabora-tion or simulation; sharing of data and process with other users.

http://www.gcube-system.org/

4.2.9 ISPyB (ESRF)

Information System for Protein crystallography Beamlines (ISPyB) is a multisite, generic Laborato-ry Information Management System (LIMS) for synchrotron-based MX experiments. Its initial func-tionality has been enhanced to include improved sample tracking and reporting of experimental protocols, the direct ranking of the diffraction characteristics of individual samples and the archiving of raw data and results from ancillary experiments and post-experiment data processing protocols. This latter feature paves the way for ISPyB to play a central role in future macromolecular structure solution pipelines and validates the application of the approach used in ISPyB to other experimen-tal techniques, such as biological solution Small Angle X-ray Scattering and spectroscopy, which have similar sample tracking and data handling requirements.

ISPyB is a joint development between the ESRF Joint Structural Biology group (JSBG), BM14 (e-HTPX), the EU funded SPINE project.

https://wwws.esrf.fr/ispyb//overviewPage.do
https://forge.epn-campus.eu/projects/ispyb

4.2.10 Twist (Synchrotron Soleil)

Synchrotron Soleil offers to the researchers a web application called Twist to search, visualize and download the data files of their experiments. The format of the data files can only be in Nexus. The Twist application is offered as a java applet which can imply some installation issues at the client site.

At the end of an experiment issued on a beamline, experiment data is saved either in Nexus files or in any other format. But, an indexing program runs periodically to create metadata of only the Nexus files in an Oracle database. This metadata is used to retrieve the experiment data files in Nexus format. Once the researcher finds back the wanted file(s), he can visualize the contents or download it with FTP. Physically the Nexus data files are saved on a server in the internal Soleil network and the other data files on an FTP server in the DMZ. Experiment files are in a Twist database, separate from the SUNset database which contains the submitted proposals, results of reviewed proposals and user information.
SUNset is the SOLEIL User Office Tool. This is the administrative web application used to manage all the life cycle of proposals submitted at SOLEIL: from user creation to proposal submission, proposal reviewing by safety manager or beamline manager or peer review committees.

Current status: Consult Experiment data as a module in the Sunset.

As for researchers the consulting of experiment data is a logical consequence of the actions that they accomplish in SUNset, therefore SOLEIL decided to integrate the functionality offered by Twist in the other administrative application SUNset. So Twist is no more a separate application but it has been integrated in the Sunset application as a new module called "Consult experiment data". Moreover, as a refactoring of the SUNset application is in progress then integration of this module for consulting experiment data has been added naturally.

The Auto-index program runs periodically to create metadata of all experiment files in the Twist database. In the Sunset database (materialized) views make this metadata available to the SUNset application. This metadata is used to search and retrieve the experiment data files. All data files (Nexus and others) will be made available to the application through NFS mounts. The researcher can download and save any type of data files (Nexus and others) via https.

Following functionalities are currently available in the refactored SUNset application:

- a search functionality to find experiments and their corresponding data file(s)
- a functionality to download the data file(s)

https://twist.synchrotron-soleil.fr/

4.2.11 Zentity (MS Research)

Zentity is a repository platform that helps academic, governmental, and scientific organizations conduct and compile research. From a technical standpoint, Zentity is a hybrid store: a combination of a triplestore and a traditional relational database. Zentity is data agnostic, supporting arbitrary data models, and provides semantically rich functionality that enables the user to find and explore interesting relationships between elements both programmatically—by using the Zentity

API—and visually—by using Microsoft Live Lab Pivot Viewer and Microsoft Research Visual Explorer.
Zentity is built on the Microsoft .NET Framework 4.0, which includes features such as OData support, WCF Data Services, and improvements to the Entity Framework. Zentity also uses Microsoft SQL Server 2008, Entity Framework, and LINQ.

http://research.microsoft.com/en-us/projects/zentity/

5 ICAT against the DCS requirements

We consider ICAT a mature Data Catalogue System and is the software of choice for the PaNdata consortium. Its core development is not connected to this project but members of the consortium contribute to it. This has resulted in a system with features that follow closely the requirements. The close interaction with the ICAT team assures that potential future requirements will be in the ICAT roadmap. The current status of ICAT against the list of D5.1 requirements as extended in this deliverable is presented in Table 3.

Authentication System	The authentication system is a plug-in. A suitable one can be done for Umbrella. Searching across multiple ICAT instances is possible.
Metadata model	The current one was designed with x-ray and neutron experiments in mind (captures the "Beamtime" concept). NXarchive [ref. to later text] has been designed specifically for ICAT.
Querying/Searching methods	Permits keyword based searching. Free-text is supported too.
Software Requirements	Enterprise Java technologies and Oracle RDBMS. The latest version can be deployed on MySQL too as requested by a PaNdata member facility.
User Interaction	The ICAT project has produced an interactive web-frontend to the system (Topcat).
Service API	The ICAT4 API is a layer on top of relational DBMS. The database is wrapped as a SOAP web service so that the tables are not exposed directly. When the web service interface definition (WSDL) is processed by Java then each data structure results in a class definition.
Hardware Requirements	According to the Software Requirements.
Documentation	ICAT is well documented. There is no up-to-date user guide for Topcat but there is a website and wiki for the project (http://code.google.com/p/icatproject/).
Support	The ICAT team is providing the PaNdata consortium with extensive support. Members participate in regular teleconferences and meetings where current and future

	developments are discussed. There is formal agreement between the ICAT project and PaNdata ODI WP4.
Licence	Open source - FreeBSD.
Data Policies	N/A.
Total Cost of Ownership (TCO)	The system is open source and its current version requires only open-source or free technologies. It offers good and responsive support. It will be used among the PaNdata partners as a common system.
Scalability and Performance	An existing installation of Petabyte scale reports satisfactory performance.
Data Ingestion	The API permits simple data ingestion.
Additional Services and Plug-ins	It is open source, has an API that is SOAP based, and is modular.
Deployment Architecture	Mostly single server instances but a federated distribution of ICATs is possible. This would enable the web portal (Topcat) to be on top of the API.
Maintenance and Sustainability	ICAT follows good software sustainability practices and has been reviewed by the Software Sustainability Institute.
Specialisation and Systems Scope	The system is specialised as it mostly realises a data catalogue service. Its scope is well suited for scientific data.

Table 3 The current status of ICAT against the list of D5.1 requirements as extended in this deliverable.

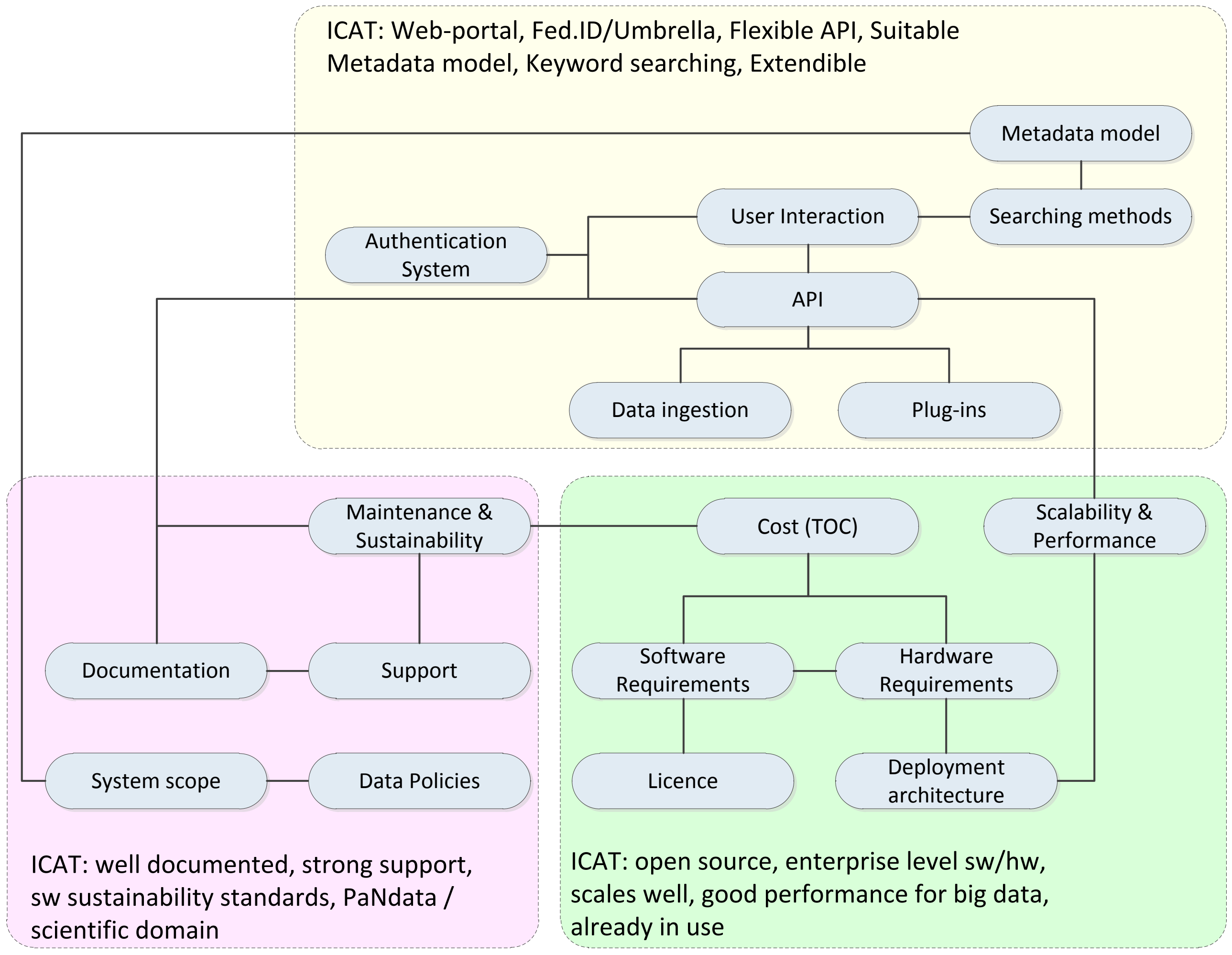
ICAT: Web-portal, Fed.ID/Umbrella, Flexible API, Suitable Metadata model, Keyword searching, Extendible
Metadata model
User Interaction
Searching methods
Authentication System
API
Data ingestion
Plug-ins
Maintenance & Sustainability
Cost (TOC)
Scalability & Performance
Documentation
Support
Software Requirements
Hardware Requirements
System scope
Data Policies
Licence
Deployment architecture
ICAT: well documented, strong support, sw sustainability standards, PaNdata / scientific domain
ICAT: open source, enterprise level sw/hw, scales well, good performance for big data, already in use

Figure 2 Grouping of DCS evaluation criteria and notes on ICAT compliance

6 Metadata elements, keywords, and data formats

The systems that are described in previous sections of this document provide services of data cataloguing. The scope of the survey is on frameworks specialised on scientific data. The nature of the data defines a common class of metadata elements. These elements as keywords and predefined fields provide meta content to the data of a scientific process. This adds structure and permits easier organisation, management and cataloguing.

The standardisation of a set of keywords as metadata elements is required in order to permit data exchange among different entities (i.e. indexing systems). The standardisation in a broader context is an agreement among the involved parties for a common set of elements. For scientific data we categorise them in two main classes. The first class carries well defined information that is often numerical or of standard options.

Example:

```
source_type:    X-ray XOR Neutron
sample_state:   A XOR B XOR C    (D would not be valid)
energy_value:   10033            (only numbers)
```

The second class of metadata may provide non structured information like a textual description of the specific scientific experiment that provided the data.

The data format used to store the scientific data is of crucial importance as it often includes the metadata. When the involved parties use a common format, they assume common internal structures and keywords. Since there is a large number of scientific communities with different needs, agreeing upon a common standard format is difficult even if it may be desirable. Towards this direction there are initiatives like the Dublin Core® Metadata Initiative (DMCI) that provides a set of vocabulary terms which can be used to describe resources for the purposes of discovery. The effort of the initiative is towards the actual terms rather than the data storage technologies that a data format has to encapsulate. Naturally, in the attempt to be a broad standard[2] the simple element set includes only the following:

```
Title, Creator, Subject, Description, Publisher, Contributor, Date, Type,
Format, Identifier, Source, Language, Relation, Coverage, Rights.
```

The DMCI provides a suitable abstract model that defines in detail the i) resource, ii) description set, and iii) vocabulary models. A cataloguing system that will support the standard should comply with these independently of the way the information is stored. A digital asset management (DAM) system consists of management tasks and decisions surrounding the ingestion, annotation, cataloguing, storage, retrieval and distribution of digital assets. In such a workflow, the use of a standard metadata elements such those of DMCI is necessary. Nevertheless for specialised applications in the scientific domain it may be required to have an extended set that will define a specific scientific application (e.g. X-ray Computed Tomography and Small-angle X-ray scattering).

[2]ISO Standard 15836-2009, IETF RFC 5013, NISO Standard Z39.85

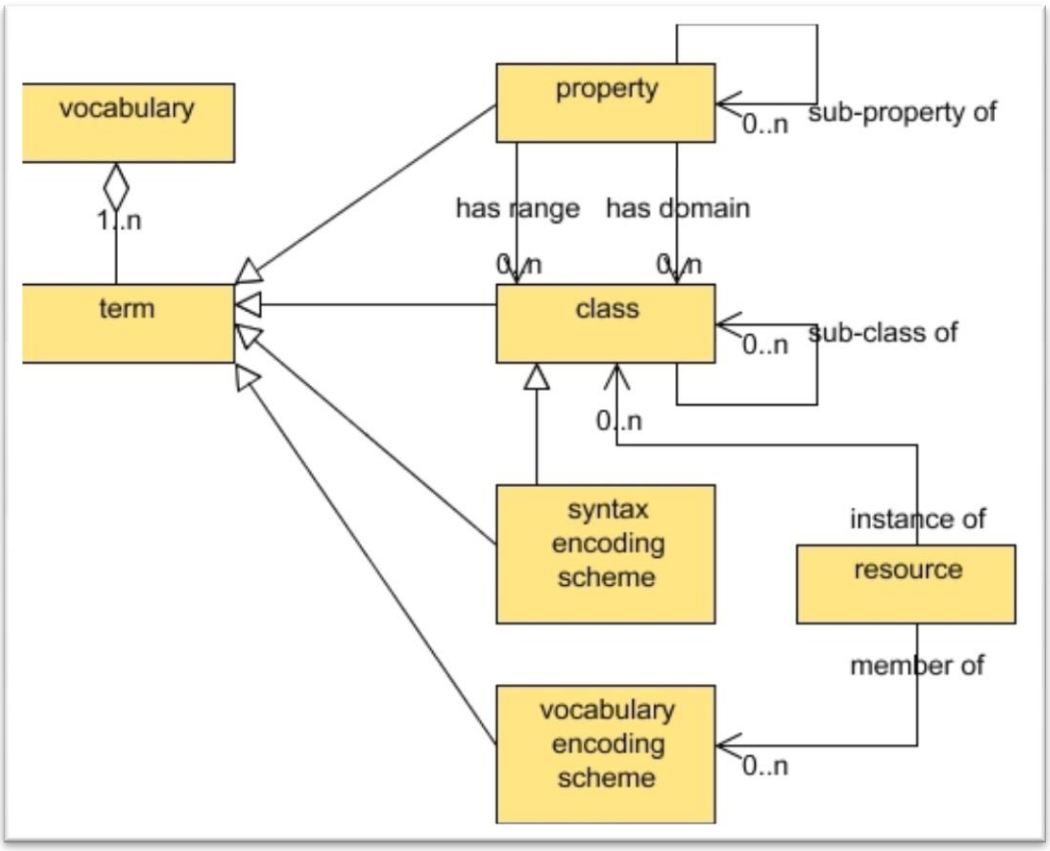

Figure 3: The DCMI Vocabulary Model: Constructing a valid set of terms requires respecting certain predefined constraints

A typical way to store DCMI metadata is in XML. The following is an example of such metadata:

```
▼<metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dc="http://purl.org/dc/elements/1.1/">
   <dc:title>PaNdata ODI Deliverable 4.1</dc:title>
   <dc:creator>PaNdata consortium</dc:creator>
   <dc:subject>data catalogues</dc:subject>
   <dc:description>Survey of scientific Data Catalogues.</dc:description>
   <dc:contributor>PaNdata consortium</dc:contributor>
   <dc:date>2012-06</dc:date>
   <dc:type>text</dc:type>
   <dc:identifier>1.234.567</dc:identifier>
   <dc:language>en</dc:language>
   <dc:rights>Open Access</dc:rights>
 </metadata>
```

Often in the scientific domain the data and the metadata are bundled and stored in the same digital data format that acts as a container. Binary formats are more efficient than the above XML for storing large data. Such a binary container format with advanced characteristics is the Hierarchical Data Format (HDF5) which has versatile data model that can represent complex data objects and metadata. When the data/metadata structure and vocabulary are not defined or enforced, DAM services as data cataloguing cannot parse them efficiently. By using HDF5 as the default file container, an advanced format NeXus[3] results in data files that comply with a predefined/standardised

[3]NeXus aims at being a common data format for neutron, x-ray and muon science.

set of “application definitions”. NeXus data files contain four entity types: data groups, data fields, attributes, and links organised in a hierarchical way.

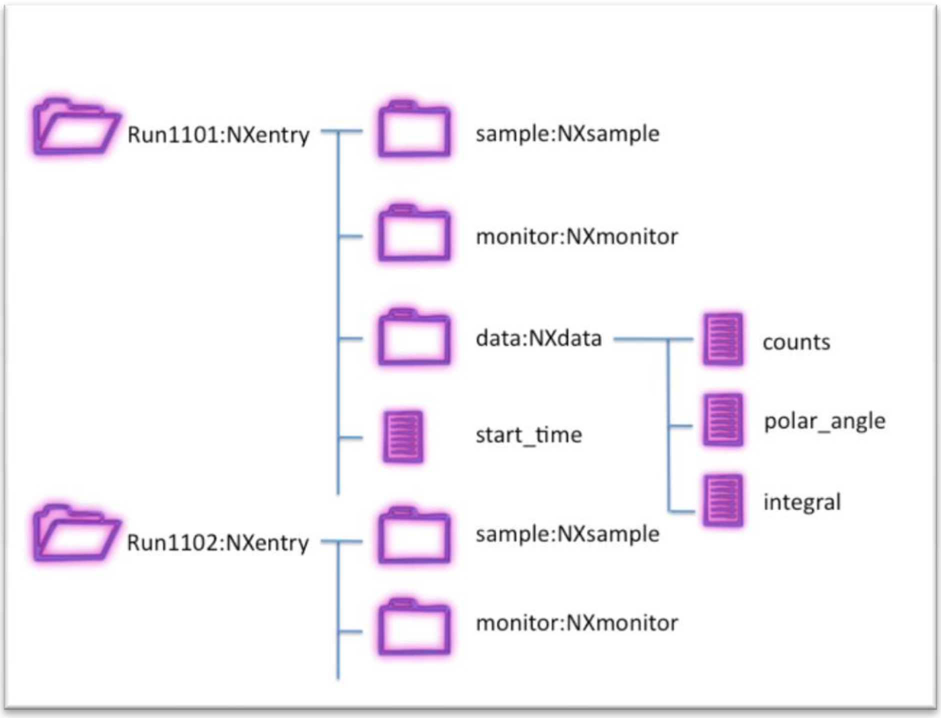

Figure 4: Simple NeXus file structure

Each field is identified by a name, such as polar_angle but each group is identified both by a name and the class type (e.g. monitor:NXmonitor). The class types define the sort of fields that the group should contain. The NeXus application definitions (classes) have certain metadata element and keywords that are required. The following is application basic class for tomography.

```
      NXtomo (application definition, version 1.0b)
 (overlays NXentry)
 entry:NXentry
   definition:NX_CHAR
   end_time:NX_DATE_TIME
   start_time:NX_DATE_TIME
   title:NX_CHAR
   data:NXdata
     data --> /NXentry/NXinstrument/data:NXdetector/data
     rotation_angle --> /NXentry/NXsample/rotation_angle
   instrument:NXinstrument
     bright_field:NXdetector
       data:NX_INT[nBrightFrames,xsize,ysize]
       sequence_number:NX_INT[nBrightFrames]
     dark_field:NXdetector
       data:NX_INT[nDarkFrames,xsize,ysize]
       sequence_number:NX_INT[nDarkFrames]
     sample:NXdetector
       data:NX_INT[nSampleFrames,xsize,ysize]
       distance:NX_FLOAT
       sequence_number:NX_INT[nSampleFrames]
       x_pixel_size:NX_FLOAT
       y_pixel_size:NX_FLOAT
     NXsource
       name:NX_CHAR
       probe:NX_CHAR
       type:NX_CHAR
   control:NXmonitor
     data:NX_FLOAT[nDarkFrames + nBrightFrames + nSampleFrame]
   sample:NXsample
     name:NX_CHAR
     rotation_angle:NX_FLOAT[nSampleFrames]
     x_translation:NX_FLOAT[nSampleFrames]
     y_translation:NX_FLOAT[nSampleFrames]
     z_translation:NX_FLOAT[nSampleFrames]
```

An advanced data cataloguing pipeline[4] should be able to parse the file and implement the above definition class. This should validate the structure of the file and get a set of metadata too. In the case of ICAT a well defined NeXus application class exists (NXarchive).

```
NXarchive (application definition, version 1.0b)
(overlays NXentry)
entry:NXentry
  @index
  collection_description:NX_CHAR
  collection_identifier:NX_CHAR
  collection_time:NX_FLOAT
  definition:NX_CHAR
  duration:NX_FLOAT
  end_time:NX_DATE_TIME
  entry_identifier:NX_CHAR
  experiment_description:NX_CHAR
  experiment_identifer:NX_CHAR
  program:NX_CHAR
    @version
  release_date:NX_CHAR
  revision:NX_CHAR
  run_cycle:NX_CHAR
  start_time:NX_DATE_TIME
  title:NX_CHAR
  instrument:NXinstrument
    description:NX_CHAR
    name:NX_CHAR
    NXsource
      name:NX_CHAR
      probe:NX_CHAR
      type:NX_CHAR
  sample:NXsample
    chemical_formula:NX_CHAR
    description:NX_CHAR
    electric_field:NX_FLOAT
    magnetic_field:NX_FLOAT
    name:NX_CHAR
    preparation_date:NX_CHAR
    pressure:NX_FLOAT
    sample_id:NX_CHAR
    situation:NX_CHAR
    stress_field:NX_FLOAT
    temperature:NX_FLOAT
    type:NX_CHAR
  user:NXuser
    facility_user_id:NX_CHAR
    name:NX_CHAR
    role:NX_CHAR
```

[4]A data catalogue may require additional systems that do the data parsing prior the data ingestion

Even if this class is designed to satisfy the specific needs of the ICAT system, it is missing certain metadata elements of the DCMI like Language and Rights while it adds additional ones.

The metadata elements are of high importance to the data management process. A data catalogue system will harvest them to get the required information. The establishment of a common set of elements can result to standard ways for managing these data. Towards this direction, NeXus, a suitable file format for scientific data, has established a standardised class of elements (NXarchive) for use in data cataloguing by the leading system ICAT. Nevertheless even mature metadata initiatives like the Dublin Core® can result in a minimal and generic set that often fails to be adopted due to the very wide variety of needs.

7 Summary

In this document we proposed a set of criteria for evaluation of scientific data catalogues. We have briefly surveyed a number of data catalogue implementations, some rather generic and other heavily specialised. In particular, we have focused our attention on the STFC's ICAT, which has been selected by our consortium as the data catalogue of choice. ICAT meets almost all of the selection criteria which have been discussed in this document. Many of the active members of the ICAT collaboration project are also active members of PaNdata. It is expected that requirements which emerge during the deployment of ICAT in PaNdata will be addressed in a timely manner.