Science & Technology Facilities Council

Technical Report
RAL-TR-2010-028

Accommodating derived data with an enhanced Core Scientific Metadata model

E Yang, B Matthews, M Wilson

October 2010

RAL Library
Science and Technology Facilities Council
Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358- 6254

Accommodating Derived Data with an Enhanced Core Scientific Metadata Model*

Erica Yang, Brian Matthews, Michael Wilson
STFC e-Science department

October 21, 2010

Abstract

The Core Scientific MetaData model (CSMD) is used by large scientific facilities to catalogue scientific data. The current version provides support to experimental scientists to access their raw data, facility managers for accounting for facility usage and other scientists who wish to re-use raw experimental data. Much of the value in scientific data is provided not only in the raw data but through the analysis of that data to derive published results. An analysis of the raw data analysis process for structural science has shown that various data sets derived from the raw data are of use to scientists and should be stored with the raw data. Extensions to the CSMD are presented to describe the analysis process so that the provenance of the derived data can be captured. A pilot implementation incorporating derived data through this extended CSMD model has been trialled with experimental scientists. Remaining challenges to the adoption of CSMD and tools it supports are considered.

Keywords: large scale facilities; neutron sources, scientific process, data management, data sharing, data linking.

*This technical report is an extended version of the paper of the same title published at the IEEE e-Science 2010 Conference in Brisbane, Australia. It also forms the major content of a technical deliverable, named Pilot Implementation, for the Infrastructure Integration for Structural Sciences (**I2S2**) project funded under the UK JISC Managing Research Data Programme between October 2009 and March 2011.

Contents

1 Introduction

Increasing quantities of the raw experimental data generated using large scientific facilities, such as large-scale photon and neutron sources, are being made available in a systematic and secure way. This data is intended for three main users: the experimental scientists who undertook the study need access to the raw data from their universities in order to analyse it further; the facilities managers need access to data to manage the use of their facilities; and other scientists may be able to access the data for re-analysis, either to verify the published results, or to derive new scientific results without the cost of repeating the original experiment.

The Core Scientific MetaData model (CSMD) [13, 8] has been designed to capture information about experiments and the data they produce in what are broadly known as the "structural sciences", such as chemistry or earth science, which consider the molecular structure of matter. It is used by the data cataloguing system ICAT [3] which is used by several large scientific facilities, in particular, the ISIS neutron source[1] the Diamond Light Source (DLS)[2], and the Institut Laue-Langevin (ILL)[3]. Data cataloguing systems support access to scientific data, but the present CSMD only addresses the raw data produced by the facility and it does not support access to the derived data produced during analysis, nor does it allow the provenance of data supporting the final publication to be traced through the stages of analysis to the raw data. At present these intermediary derived data sets must be stored locally by the scientists, and are not archived for other purposes. Thus the support for the intended users is partial.

Bioscientists have used workflow tools to capture and automate the flow of analyses and the production of derived data for many years [9] and can now automatically run many computational workflows [16]. In other structural sciences, such as chemistry and Earth sciences, the management of derived data is less mature, workflows are not standardised and can less readily be automatically enacted. Rather the data needs to be captured as the analysis proceeds so that scientists do not lose track of what has been done. A data management solution is required to capture the data trails that are generated during analysis, with the aim of making the methodologies used by one group of researchers available to others.

Further, the accurate recording of the process so that results can be replicated is essential to the scientific method. However, when data are collected from large facilities, the expense of operating the facility means that the raw data collection effectively cannot be repeated. Therefore tests to replicate results has to come from re-anlysis of raw data as much as repetition of the data capture in experiments.

In order to provide support for the analysis undertaken by the experimental scientists; to permit the tracing of the provenance of published data; and to allow access to derived data for secondary analysis, it is necessary to extend the CSMD to account for derived data and to record the analysis process sufficiently for the needs of each of these use cases. In terms of data provenance [6], the current CSMD approach identifies the source provenance of the resultant data product, but it needs to be extended to describe the transformation provenance as well.

In this paper, after a summary of the existing CSMD, an example scientific process will be described to motivate the extensions to the CSMD. Section 4 will then detail extensions to the CSMD to meet these requirements, before a pilot implementation of the extended CSMD

[1] http://www.isis.stfc.ac.uk
[2] http://www.diamond.ac.uk
[3] http://www.ill.eu

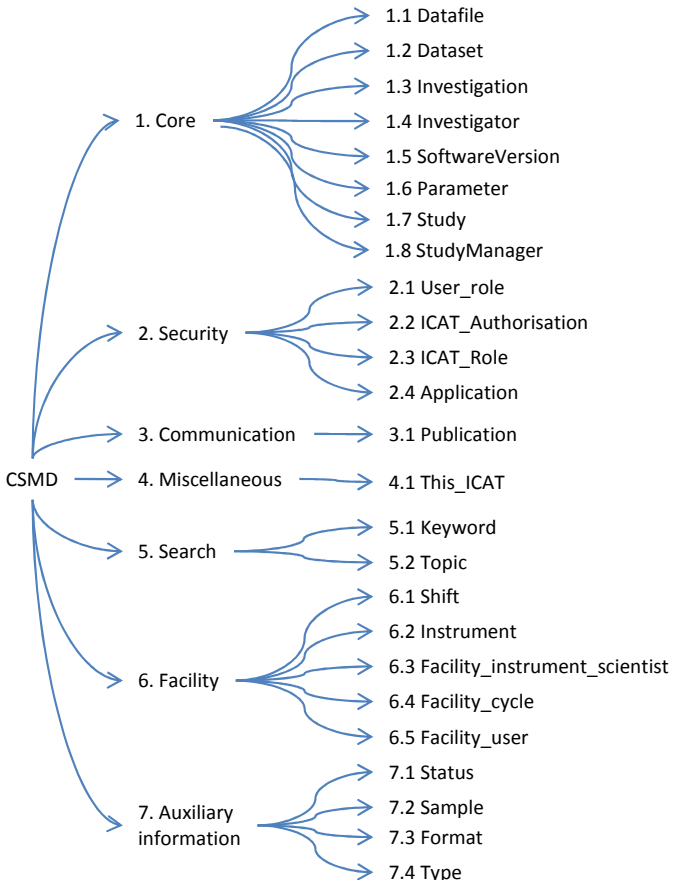

Figure 1: A classification of the concepts in CSMD

is described using the ICAT data catalogue system. Finally the limitations of the proposed extensions, practical limitations on the adoption of the data catalogue system and future work will be considered.

2 Core Scientific MetaData model

The Core Scientific MetaData model (CSMD) [13] is an extensible model of metadata originally designed to capture a common set of information about the data produced by experiments, measurements, and simulations in facilities science. The model is the result of an analysis of science practice over a number of years and a range of projects.

CSMD was developed primarily to allow facility operators, such as STFC, to introduce a systematic approach to manage their data assets across the heterogeneous scientific facilities. Although operators may produce data files of different formats and content resulting from different equipment, experiments, or disciplines, there are commonalities features of the context of the data that can be captured. They include:

1. the description of the data production process (e.g. where/when/by who/how);
2. the format, type, owner, and identifier of the data;
3. the parameters in which the data should be interpreted;

4. the relationships between data.

Having a standardised metadata model underpinning the data management infrastructure that an operator uses, supports a common strategy towards maintaining, searching, and discovering data assets, reducing the overall operating cost.

The model as it currently stands aims to describe the physical raw data files (binary, images, or text containing numeric values) produced by the data acquisition software of a detector within an instrument. These files have formats which depends on the equipment, the facility, or the program that the data is produced from. The Network Common Data Format (netCDF) [10] and Hierarchical Data Format (HDF) [4] are well defined formats used by many laboratories, while NeXus [7], derived from HDF5, is a common data format targetted at neutron, x-ray, and muon sciences which several facilities have adopted to different degrees: not all the data files produced within these communities use this format since many instruments still produce older non-standard formats.

In CSMD data files are grouped into *datasets*, where a dataset is an abstract notion referring to a set of related data files. How the files are related is determined by the context. For example, if an experiment produces 10 files in a run, which is repeated 100 times in different temperatures, 100 datasets can be created, each with the 10 files produced under a specific temperature. This dataset concept is essential for experiments that produce a large number of files in each run.

Datasets are then grouped into *investigations*, where an investigation - which can be an experiment, a set of measurements, or a simulation - is defined as any data generation activity. Like the dataset, an investigation is not a concept referring to an object of physical presence, but rather an abstract notion referring to a set of related datasets generated from the same data generation activity.

Investigations are further grouped into *studies*, where a study is also an abstract notion referring to a set of related investigations, in other words, a set of related data generation activities. For example, two investigations, an experiment of a sample and a related simulation of the process, can be grouped together to form a study of the sample.

The CSMD has been implemented and deployed in STFC to support scientific data cataloguing and management for its major international facilities. The current production implementation of CSMD, i.e. ICAT 3.3[4], is based on the CCLRC Scientific Metadata Model v2 [13] with extensions. This model forms the core of the ICAT infrastructure to catalogue, manage and distribute data for facilities users.

Although CSMD was originally intended to accommodate data collection and processing a much wider context of scientific studies from raw data collection to downstream data analysis, it is currently only *being used* to support raw data cataloging. In order to focus on the key data management issues throughout the data production pipeline and to clarify the extensions needed for derived data, we identify the core and optional concepts in the model. The concepts in CSMD can be classified into six categories (see Figure 1):

Core these concepts are core to scientific data management. Capturing the data outputs involve four data objects: datafile, dataset, investigation, and study. A datafile is a *physical data object* that is stored on physical storage disks, while datasets, investigations, and studies are *abstract data objects* that encapsulate other (physical or abstract) data objects as

[4]http://code.google.com/p/icatproject/

described above. Investigator and StudyManager are people associated with an investigation and a study, respectively. Process[5] is an activity that produces or consumes data objects. Parameter is the context of the data production process.

Search classifiers which facilitate the search and discovery of core concepts.

Communication entities linking between research objects so that the provenance of a research publication can be traced back to the data holdings.

Security entities which enforce access policies on the data holdings.

Miscellaneous entities which identify the specific instance of ICAT metadata catalogue.

Facility specific concepts related to facilities. They are introduced to capture the contextual information (e.g. which facility, instrument, shift, the data is collected, how it is collected, the instrument settings) associated with the (raw) data collection process.

Auxilliary Information the information associated with data holdings. It is currently being used to store information related to *raw* data files, such as sample, parameters (e.g. temperature, humidity), file format. But it should be possible to extend or adapt them to store any information related to data holdings produced along data analysis pipelines.

Two types of information are left out from Figure 1: links between the concepts within a category; and those between the concepts across categories. We address the former in the rest of this paper. The latter does not directly relate to the paper, and we shall not expand on that further.

3 Derived Data in the Analysis Process

In this section we study in detail an example data analysis pipeline from the raw data gathered at a facility to the final scientific findings suitable for publication.

Along the pipeline, three concepts, raw, derived, and resultant data, are often used to differentiate the roles of data in different stages of the analysis and to capture the temporal nature of the processes involved. *Raw data* are the data acquired directly from the instrument hosted by an facility, in the format support by the detector. *Derived data* are the result of processing (raw or derived) data by one or more computer programs. *Resultant data* are the final findings of an analysis, for example, the structure and dynamics of a new material being studied in an experiment.

3.1 Background

We initially performed a desk study of three experiments involving two different types of facilities: neutron and synchrotron facilities, in the UK. One experiment is in the domain of Chemistry using the Diamond synchrotron and the UK National Crystallography Service (NCS) [2] to determine the structure of atoms in solids using X-ray diffraction. The other two

[5]In CSMD 2.0 and ICAT 3.3, the concept Process is called SoftwareVersion.

experiments aim to determine the structure of atoms of matters (e.g. liquids or solids) using neutron techniques: one uses the neutron diffraction[6] provided by the GEM instrument[7] and the other small angle neutron scattering[8] offered by the Sandals instrument[9]. Both instruments are located at the ISIS neutron spallation source.

The NCS analysis workflow is the most prescriptive among the three experiments because the processes involved are standard and the data formats used are well established [2]. The analysis workflows for the other two experiments are more complicated but the nature of the analysis is similar and both workflows involve

- computationally intensive programs, and
- intensive human oriented activities that demand significant experience and knowledge to direct the programs.

In practice, it can take months from the point that a scientist obtains the raw data to the point where resultant data are obtained. Both workflows overlap in their data correction process as they use the same set of programs to correct the raw data obtained from the instruments (e.g. to identify the data resulting from malfunctioning detectors), though this represents only a small part of the respective workflow.

Given these similarities we shall focus on the details of the data analysis flow of the neutron scattering experiment using the GEM instrument to study derived data problem, although hierarchical task anlaysis [12] has been applied to all the studies and the abstractions do generalise across instruments, techniques, programmes and disciplines.

3.2 Data Analysis

Data analysis is the crucial step transforming raw data into research findings. In a neutron experiment, the objective of the analysis is to determine the structure or dynamics of materials under controlled conditions of temperature and pressure. Figure 2 illustrates a typical flow for analysing raw data generated from the GEM instrument using Reverse Monte Carlo (RMC) based modelling [15]. The RMC method is probabilistic, which means that a) it can only deliver approximated answer and b) in theory, there is always scope to improve the results obtained earlier using the same method.

In the figure, rectangles represent the programs used for the analysis; rounded rectangles without shadow represent the data files generated by computer programs; rounded rectangles with shadow represent data files hand-written by scientists as inputs to the programs; ovals represent human inputs from scientists to drive the programs; solid lined arrows represent the information flow from files to programs, from programs to files, or from human to programs; and the dashed lined arrows are included to highlight the human oriented nature of these programs demanding significant expertise. This is an iterative process that takes considerable human effort.

3.2.1 Data reduction

Three types of raw data are input into the data analysis pipeline: sample, correction, and calibration data. They are first subject to a data reduction process which is facilitated by

[6] http://www.isis.stfc.ac.uk/instruments/neutron-diffraction2593.html
[7] http://www.isis.stfc.ac.uk/instruments/gem/gem2467.html
[8] http://www.isis.stfc.ac.uk/instruments/small-angle-scattering2573.html
[9] http://www.isis.stfc.ac.uk/instruments/sandals/sandals6929.html

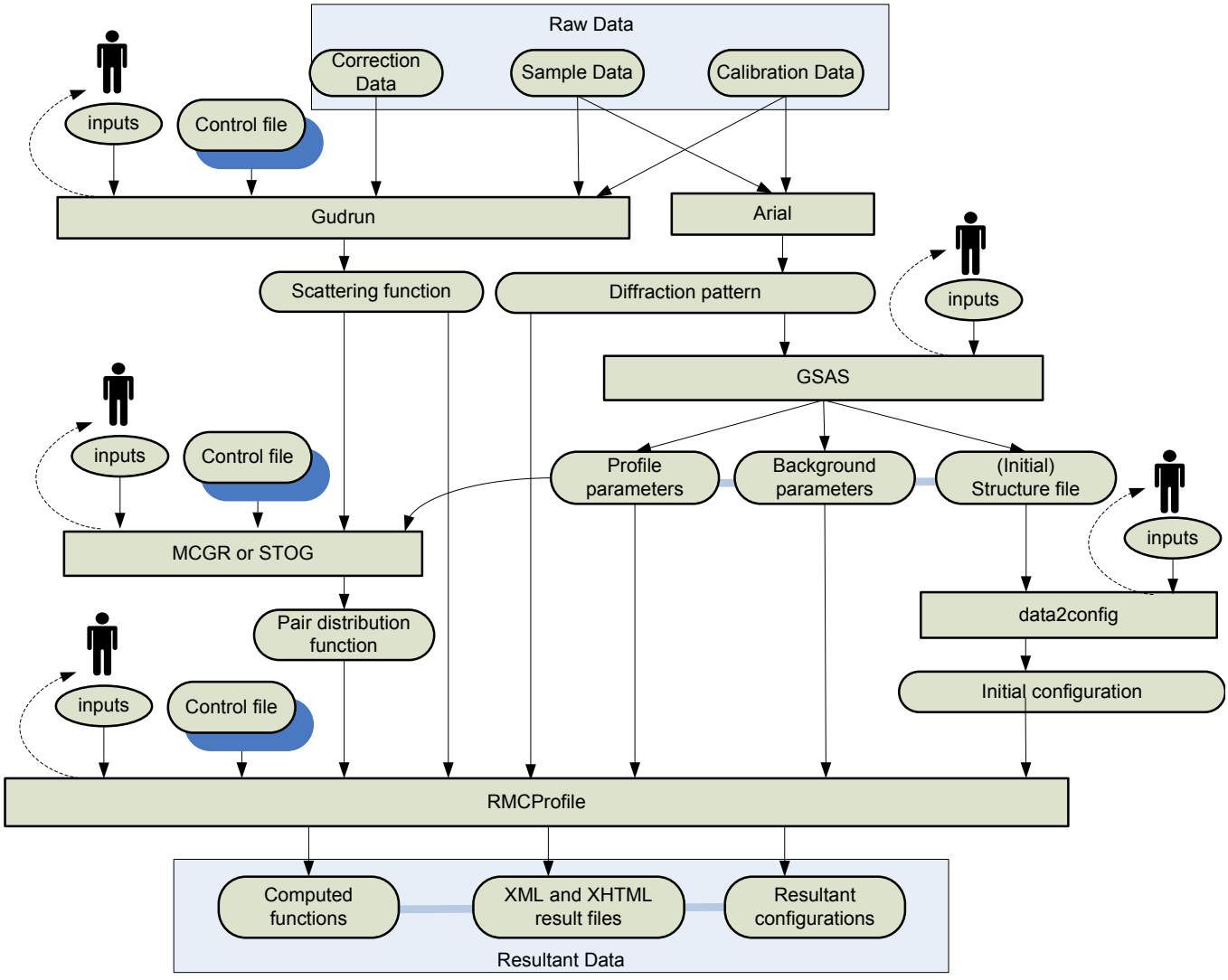

Figure 2: The RMC data analysis flow diagram

two programs: Gudrun, a Fortran program with a Java GUI, and Arial, a IDL program. The outputs from Gudrun[10] are a set of scattering functions, one for each bank of detectors. For Arial[11], the outputs are a set of diffraction patterns, again, one per bank of detectors.

With Gudrun, the human has to subtract any noise in the data going from scattering function to pair distribution function (through the MCGR or STOG program). Noise can arise from several sources, e.g. errors in the program, or noise due to the statistics on the data. In other words, when the other programs use the derived data generated by Gudrun, human expertise is required to steer the way the data is used.

3.2.2 Initial structural model generation

The next step is the process of generating the initial configuration of the structure model that will be used as the input to the rest of the RMC workflow. This step requires three programs (i.e. GSAS, MCGR or STOG, and data2config) to transform the reduced data into structure models that best fit the experimental data. To do this requires determining the structural parameters (e.g. atom positions), illustrated as the sets of data files under GSAS, for all the crystalline phases present, which are: profile parameters, background parameters, and (initial) structure file.

Most neutron and synchrotron experiments use the Rietveld regression analysis method to refine crystal structures. Rietveld analysis, implemented in GSAS, is performed to determine the structural parameters as well as to fit the crystal structure to the diffraction patterns using regression methods. Like all regression methods, it needs to be steered to prevent it following a byeway. Some values in the pair distribution functions produced from MCGR or STOG are compared with their counterparts in the scattering functions to ensure that they are consistent. If they are not, the scientist repeats the analysis.

The data2config program takes the configurations generated from GSAS, or from crystal structure databases to determine the configuration size of the initial structure model.

3.2.3 Model fitting

All the derived data generated up to this point represents an initial configuration of the atoms, random or crystalline, which is fed into the RMCProfile [14] programme which implements the RMC method to refine models of matter that are mostly consistent with experimental data. It is the final step in the analysis process to search for a set of parameters that can best describe experimental data given a defined scope of the search space and computational capacity. This is a compute-intensive activity which is likely to take several days of computer time. It is also a human-oriented activity because human inputs are required to "steer" the refinement of the model.

3.3 Discussion

The scientific process under consideration passes through the main phases of sample preparation, raw data collection, data analysis and result gathering. The overall data analysis process described above passes through the three phases of data reduction, initial structural model generation, and model fitting. This hierarchical structure is common to the different

[10] http://www.isis.rl.ac.uk/disordered/Manuals/gudrun/gudrun_GEM.htm

[11] http://www.isis.stfc.ac.uk/instruments/osiris/data-analysis/ariel-manual9033.pdf

processes analysed. However, as the detailed example above illustrates, within each of these phases there are many different programs involved (with potentially different versions), with varying numbers of input and output objects. Because the analysis method is probabilistic, there is always scope for further improvements to the results so variations on the analysis can always be undertaken.

Throughout the analysis, many of the intermediate results are useful both for the scientists who perform the original experiment and others in the scientific community. The investigators or others can, for example: use them for reference; revisit them when better resources (more powerful computers, better analysis methods or better programs) are available; and revise them when better knowledge about the program behaviours are available.

The scientists consulted are thus not only motivated to publish their final results but also the raw and derived data generated along the analysis flow. This is especially true for new analysis methodologies, such as the RMC method described in this paper which is a relatively new method in the neutron scattering community which those who use it wish to have accepted more widely. In this case, scientists are highly motivated to publish the *entire data trail* along the analysis pipeline and publicise the *methodology* that is used to derive the resultant data. Making their data available potentially can lead to: more citations to their published papers and results; awareness and adoption of their methodology; and the discovery of better atomic models built on the models they have derived.

Data archiving is also of interest to the facilities operators because of the potential of derived data reuse by other researchers who would add more value to the initial experimental time. However, apart from the raw data, neither the ICAT infrastructure nor the CSMD model capture derived data whose management is currently left to the experimental scientist. In the next section we will propose extensions to the CSMD model to capture the derived data on the basis of an abstraction of the detailed workflow described here.

4 An Enhanced CSMD

This section presents how we extend the CSMD model to describe the analysis process so that the provenance of the derived data can be captured. Several factors are important for capturing data provenance, including:

- the *data objects* involved;
- the *programs* that produce or consume data objects;
- the *ordering* of the programs; and
- the *parameters* to the programs.

Figure 3 is an UML object model depicting the extensions and modifications to the core of the existing CSMD model to support derived data. The actors are not included in the diagram because security is not the prime concern in this paper. Specifically, the extensions are introduced to the model underpinning ICAT 3.3 along following directions:

- adding a SoftwareExecution type to investigation;
- linking program to a software execution;
- linking software executions with datasets;
- associating parameters with a software execution; and
- introducing nested study.

We shall now describe the rationales behind these extensions.

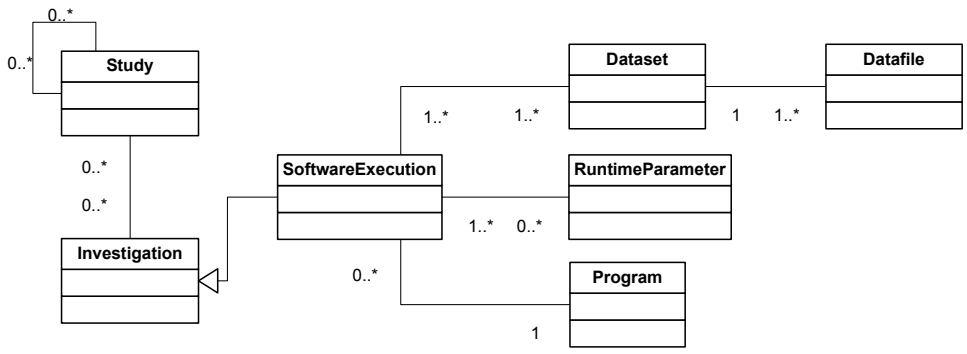

Figure 3: An Extended CSMD UML Model for Supporting Derived Data

4.1 Adding a SoftwareExecution investigation type

As discussed in Section 2, an investigation models a data handling activity, which, in the current model, means three types of investigations: measurements, experiments, and simulations [8]. None relates to the data handling activities in an analysis process.

A new type of investigation, *SoftwareExection* is introduced to model the *runtime* processing units in the process. This extension provides an end-to-end support for data management covering the experimental data gathered from instruments, to intermediate data generated in the process, to the resultant data finally appeared in papers.

4.2 Linking program to SoftwareExecution

A software execution represents an execution of a computer program for a part of the analysis in the process. It is a runtime notion meaning that it is not only associated with a static software program but also inputs (including data files and the parameters) that drive the program and the corresponding outputs resulted from running the program using those inputs. A software execution comprises of: one (*and only one*) program, one or more input datasets, one or more output datasets, and zero or more parameters to the program.

4.3 Linking software executions to datasets

4.3.1 Input and output datasets

Two types of datasets are introduced to denote the inputs to and outputs from an execution of a program. *They are associated with an execution not the program.* This is an important aspect of the analysis we would like to capture reflecting the the open ended nature of scientific research.

4.3.2 Associating multiple software executions to an input dataset

In the current model, there is an one to many relationship between investigation and dataset. However, a program can run many times using different sets of parameters but with the same input dataset. Hence, the relationship between investigation and dataset is extended to be many to many so that it accommodates this scenario.

4.4 Associating parameters with SoftwareExecution

One program can be executed several times resulting in several (program) executions. All can correspond to the same input dataset(s) but with different output datasets and runtime parameters. A program can take zero or more parameters, but a parameter must be associated with at least one software execution. The linkage between RuntimeParameter and Program is through SoftwareExecution.

4.5 Study and nested study

Study is a notion for grouping related investigations. It is the means by which SoftwareExecutions are related to each other and SoftwareExecutions are related to other types of investigations. For example, in the analysis process, a study is used to group investigations in a particular order, which can be *sequential, parallel, and adjunctive*. The ordering depicts explicitly the relationship between the investigations reflecting the sequence of the data handling activities involved in a scientific endeavour.

Through a study, the investigations can be chained together to form a connected sequence of analysis activities in the process. For example, using the same set of programs, executions can be chained together to form an analysis flow reflecting the use of a set of input data files and parameters. A different chain can be formed reflecting the use of a different set of files and parameters.

It is not uncommon that iterations of analyses are performed before a satisfied set of results can be obtained. Several of such "chains" can be formed when conducting an analysis process. A nested study is a notion for grouping related studies (or chains). Such relationship can be *adjunctive* in that the output from one study is used as the input to another. The studies can be *parameter sweeps* in that two studies use the same set of programs and input data files but with different runtime parameters. They can also be *functionally equivalent* when two studies use the same set of inputs (data files, parameters) but with a set of functionally equivalent programs.

5 ICAT-personal: A Pilot Implementation

A pilot implementation of the extended CSMD model, named **ICAT-personal**, has been developed and is available through sourceforge website[12]. It is a lightweight version of ICAT because it implements the core of the extended CSMD model to demonstrate the feasibility of capturing and cataloguing derived data. We describe its design and development focussing on the current capabilities of the implementation.

5.1 System Architecture

Figure 4 illustrates the system architecture for ICAT-personal. It consists of three layers: client, utility programs, and a repository. It supports two types of clients: command line scripts and or native OS context menu. The client tool, including the client-side of the utility programs, needs to be installed on users' computer. The client interacts with the server-side utility programs which connects to the persistent data repository through Java entity beans

[12] http://icatlite.sourceforge.net/

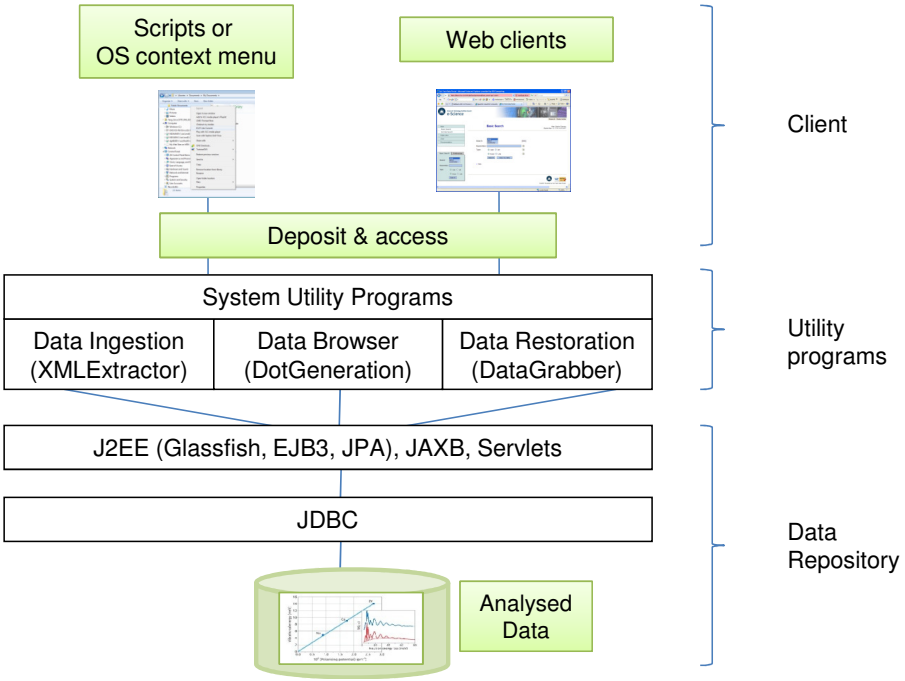

Figure 4: ICAT-personal System Architecture

and other classes hosted by Glassfish, a J2EE container provided by Sun/Oracle. JAXB is used to parse the XML ingestion file and generate Java entity beans from the XML.

Three capabilities of data organization are supported, they are: data ingestion, browsing, and restoration. The targeted audience of this implementation is individual scientists who need a data management tool to assist their own research. Future releases will investigate how well the model accommodates issues of data reuse (e.g. secondary analysis and cross analyses study), and data sharing (e.g. derived data publication, linked data, and its relevance to automated experimentation). As a pilot implementation, data annotation, searching and discovery, although important, are not considered in the implementation.

The UML model presented in the previous section is mapped into two data models: a XML schema and a database schema. Both are available through the sourceforge website. The former is used to guide the ingestion of data files and programs into an ICAT-personal repository whilst the latter is the structure underpinning the repository. We use the Gudrun program in the RMC workflow to explain their role in managing derived data.

5.2 Derived Data Management

Figure 5 illustrates the "before" and "after" scenarios of using ICAT-personal tools to manage derived data. The left hand side is a number of hierarchical file folders where scientists store the programs, run scripts, raw data files, instrument settings, and initial parameter inputs to programs, each in a separate directory. The last one is called a working directory where the parameters (stored in a configuration file), raw data input files, intermediate and final output files reside. Each execution of the programs corresponds to a separate working directory. As

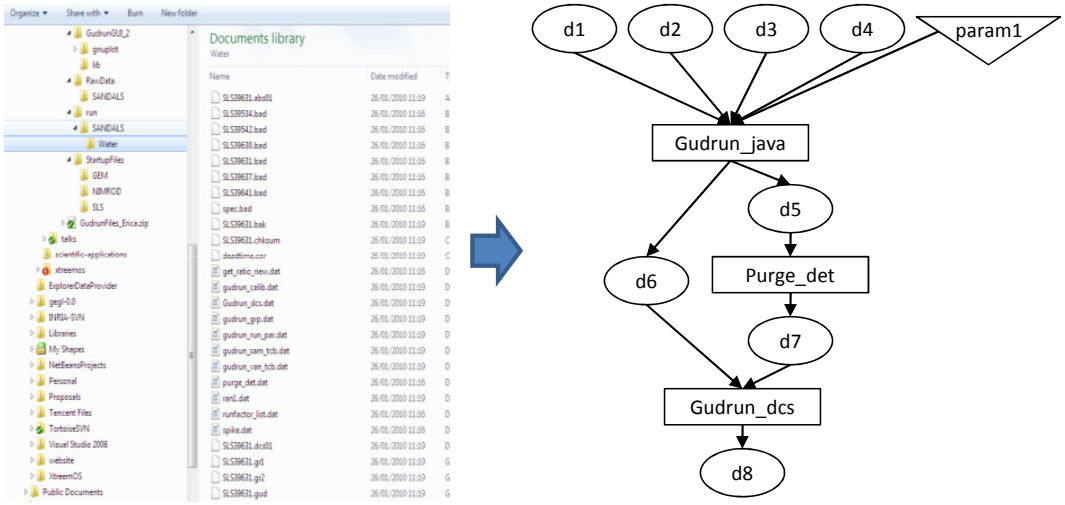

Figure 5: Derived Data Management: An Example

with the RMC analysis process, many scientific analyses involve several programs. Scientists often end up with many directories, each storing the data resulted from one execution.

Managing the working directories is challenging because:

- Most programs are run many times. Until the final results at the end of the analysis process are available, it is sometimes difficult to tell which executions are useful. So, all the potentially useful ones need to be kept.
- Scientists also need to keep track of the linkages between the executions. Again, until the final results are available, all the linkages (which often mean many directories, and sub-directories) have to be kept.
- Different scientists have their own way of keeping the parameters (e.g. storing in the working directory, on a paper notebook). Without the parameters, it is hard to understand the outputs from the programs or continue other researchers' analyses. Even with the raw data, it is difficult for other people to reproduce derived data.

5.2.1 Data Ingestion

On the right hand side of the diagram is a *structured representation of the execution* of the programs involved in Gudrun. The structure represents how the executions of different programs inside Gudrun are linked together. ICAT-personal tools store the structure as well as the contents inside the structure into an ICAT-personal repository underpinned by the J2EE technologies depicted in Figure 4. This process is called ICATlite data ingestion which is guided by an ICAT-personal XML schema compliant XML file. The file captures:

- the inputs, including data files and parameters (or parameter files), to and the outputs (e.g. data files, plots) from the programs;
- which files are produced or consumed in the same context (e.g. belonging to the same SoftwareExecution);
- the programs in the process; and
- the execution order of the programs.

Figure 6: The Gudrun example in a Web browser view with details of data files involved

The tools provide two further capabilities: browsing and restoration of the archived executions.

5.2.2 Data Browsing

An ICAT-personal tool, named DotGeneration, provides data browsing capability. It takes an ICATlite data ingestion XML file, transforms it into a Graphviz[13] dot file, and generates a flow diagram as depicted on the right hand side of Figure 5.

Datasets, depicted as `d1` to `d8` in the diagram, are used to capture the relationship between data files produced or consumed by one execution. In Figure 5, among all the input data files to Gudrun_java, four datasets are formulated, they represent four groups/types of data: raw data, sample and vanadium metadata, instrument data, and neutron/x-ray information, respectively. Other scientists may consider different types of relationships between the files by classifying them into three datasets: raw, correction, and calibration data. Such grouping is important because the relationships between the files are not self evident by examining them directly.

Figure 6 depicts another view of the above Gudrun example presented in a Firefox Web browser, expanded with the detailed data files involved in each program.

5.2.3 Data Restoration

As presented in the previous section, a SoftwareExecution is an encapsulation of the objects (the program, and the inputs and parameters to and outputs from the program) involved in running a software application. Three ordered SoftwareExecutions, corresponding to Gudrun_java, Purge_det, and Gudrun_dcs, respectively, are grouped into one study, which represents an *instance* of the data reduction process, involving

- all the programs, and
- all the raw and derived data, comprising of:
  - all the initial input data files,
  - environment and instrument settings,
  - parameters that used to drive the programs,
  - all the intermediate outputs, and
  - finally to the reduced data files.

This process can be repeated many times leading to many studies (i.e. execution instances) of the process. Each corresponds to a combination of three SoftwareExecutions captured by the ICATlite data management tool. Structured data at various levels (dataset, investigation, and study) can then be restored using the ICAT-personal DataGrabber tool from the repository.

6 Discussion and Future Work

The data management approach to handling the analysis process would seem well matched to the infrastructure supporting structural science in facilities and potentially a wider scientific community. Storing and retrieving data from throughout the scientific process is a common

[13] http://www.graphviz.org

problem across many disciplines that exploit computational methodologies and high throughput data handling techniques. The analysis presented here in detail only addresses a single study in earth sciences, while other studies in chemistry and crystalography have contributed to the analysis leading to the proposals for changes to the CSMD, and the approach described is also now being generalised into a common information model for structural science in the I2S2 project[14].

It is nevertheless a concern whether the breadth of tasks analysed reflects the whole scope of the target system. At present the usage patterns of the facilities considered are reflected in the sample of tasks analysed, but that may change over time. Other facilities may need to be supported by the CSMD which will introduce further disciplines and different data transformation processes. In particular, if disciplines such as astronomy and earth observation data were to be included, the data collection and analysis processes from those disciplines might lead to further suggestions for change to the CSMD.

The changes proposed to the CSMD capture the source of the data, and the transformation process that is has gone through, but the implementation does not provide a comprehensive provenance management system. [6] argues that a provenance management system can only be useful for a real world application if it allows querying of provenance information for resultant data items. It is unrealistic to expect a complete provenance management system which will use provenance data to automatically recreate resultant data items by executing the transformations that were used in its creation [5].

It would be possible to enhance the ICAT prototype to allow the propagation of the complete provenance of resultant data so that researchers can query it for the transformations used without having to successively unpack the datasets involved. In a simple example, if it becomes known that a particular version of a piece of software was unsafe for a parameter range, the provenance could be queried to provide all resulting data that was produced by using that software in its unsafe range. A more complex example would query for a combination of transformations within the provenance from different datasets in a study, e.g. programs X and Y were used consecutively in the transformation when their underlying models have been found to be incompatible and the resultant data could be unsafe. Such advances on the current implementation would clearly add to the safety of the scientific results derived from the transformations recorded in the provenance, although beyond the scope of the current development.

The scientific process described above was undertaken as publically funded university research for which the main security concerns are to embargo release of data until after the scientists undertaking the experiments have publshed their results and then to make them as publically open as possible to gain maximum value from the investment. However, large facilities of the class considered in this paper are also used by commercial organisations, or academics funded by commercial organisations. In these cases there may be more exacting security concerns. The modifications proposed here to account for derived data address the Core part of the CSMD only. The second main module of the CSMD addresses security metadata. It is common in these circumstances for all derived data to be required to be handled as the original data received in which case a single data policy would apply to the whole CSMD record. However, security policies are becoming more sophisticated and it is possible for the derivation process to either reduce or, more likely, increase the security constraints on data as it moves through the scientific process and its value increases. When

[14] http://www.ukoln.ac.uk/projects/I2S2/

Table 1: Stakeholder Engagement

When	What	Stakeholders
June 2010	Project internal meeting	NCS + I2S2 use case investigators/partners
June 2010	1st Demo and basic functionalities	ISIS GEM & SANDALS instrument scientists
July 2010	Project internal meeting	NCS + I2S2 use case investigators/partners
Aug. 2010	Telco	I2S2 project manager with JISC programme manager
Sept. 2010	Demo discussion	ISIS GEM instrument scientist
Oct. 2010	functionality refinement	ISIS instrument scientist, facility IT personnel

different policies apply to the derived data from the original data then the current single CSMD security node will not be enough, but would have to link policies to individual datasets. Alternatively, the current single security node could be maintained with the use of more sophisticated policies that refer to differently labelled data items explicitly [11]. As commercial use of large facilities becomes more common security issues will become increasingly important to resolve and standardise.

A recent proposal advocates encapsulating published data files in self-contained units of knowledge which they term research objects - semantically rich aggregations of resources, that possess some scientific intent or support some research objective [1]. An RO bundles together essential information relating to experiments and investigations. This includes not only the data used, and methods employed to produce and analyse that data, but also the people involved in the investigation. The authors present a number of principles that they expect such objects and their associated services to follow: reusable, repurposeable, repeatable, reproducable, playable, tracable. These are indeed the properties which the CSMD records have in principle after the inclusion of the modifications proposed in this paper. The authors propose the use of rich ontologies to encode these properties as an essential requirement for their usability. The current CSMD lacks such semantically rich encoding, but this again would appear to be a clear direction for further development.

7 Final Remarks

The work described in this report has been presented to and discussed with various stakeholders of the I2S2 project since May 2010. The development, especially the software design and development, has gone through an iterative process, guided by the continuous feedbacks and comments from our stakeholders. Features were planned in about 2 months in advance based on the requirements we have gathered in the early stage of the project (i.e. the requirement deliverable [17, 18]). The features were showcased to perspective stakeholders to gather feedbacks which were fed immediately into the next round of the feature planning, implementation and revision.

7.1 Stakeholder Engagement

Table 1 describes a list of meetings and informal discussions we have with our stakeholders related to the development of the pilot implementation.

7.2 Stakeholder Feedbacks

This section briefly summaries some desirable features extracted from the discussion we have with the stakeholders. These features can be used to guide the next phase of the pilot implementation till the end of December 2010, depending on the availability of project resources.

Data Browsing

1. The browsing interface should provide flexibility allowing scientists to have a detailed view of the data files (i.e. zoom in) but also have an abstract view of the dataset (i.e. zoom out). This is not supported by the current implementation (as of October 2010).
2. The interface should also allow scientists to flag up the important components (e.g. key inputs or key outputs) of an analysis.

Data Provenance Versioning Scientists have also commented it would be good to allow them to 'roll back' to a previous version of an analysis. This is because in the day-to-day data analysis, it is often in a later stage of an analysis one realises the mistakes they have made in the early stage of the analysis. The 'rolling' back operation would allow them to go back to take a different path (e.g. with different parameters, or reducing the previously unidentified noise in raw data files) down the analysis pipeline.

Annotation During the August telco with the JISC programme manager, an interesting comment upon "automated or guided metadata capture" were raised. The current design and implementation largely hide the complexity of metadata capturing and ingestion from users. However, by metadata, we mean specifically data provenance. This seems to be sufficient for the present target users, i.e. individual scientists, as identified in the implementation plan. That is why the tools are call ICAT-"personal". It is intended for personal use. Hence, there is no security model in place in the current infrastructure design to enforce security functionalities, such as authentication, access control, embargo control.

However, if other types of metadata, like those defined in Dublin Core (e.g. the creator of a data file), about analysed data are required, more sophisticated infrastructure components (e.g. user identity management, authentication) have to be introduced into the software system.

Acknowledgments

This research was supported by the JISC's Managing Research Data Programme under the Infrastructure for Integration in Structural Sciences (I2S2) project.

8 Appendix

8.1 The Schema for Data Ingestion XMLs

This section presents a XML schema based on the extended CSMD model to facilitate the data ingestion functionality of the ICAT-Personal implementation.

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0" x
  <xsd:element name="root" type="rootType" />
  <xsd:complexType name="rootType">
    <xsd:sequence>
      <xsd:element name="processes" type="processesType" />
      <xsd:element name="parameters" type="parametersType" />
      <xsd:element name="datafiles" type="datafilesType" />
      <xsd:element name="datasets" type="datasetsType" />
      <xsd:element name="investigations" type="investigationsType" />
      <xsd:element name="studies" type="studiesType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="studiesType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="study" type="studyType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="studyType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="investigationref" type="investigationrefType"
      <xsd:element maxOccurs="unbounded" name="studyref" type="studyrefType" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" />
    <xsd:attribute name="name" type="xsd:string" />
  </xsd:complexType>
  <xsd:complexType name="studyrefType">
    <xsd:attribute name="idref" type="xsd:string" />
  </xsd:complexType>
  <xsd:complexType name="investigationrefType">
    <xsd:attribute name="idref" type="xsd:string" />
  </xsd:complexType>
  <xsd:complexType name="investigationsType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="investigation" type="investigationType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="investigationType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="datasetref" type="datasetrefType" />
      <xsd:element name="processref" type="processrefType" />
      <xsd:element maxOccurs="unbounded" name="parameterref" type="parameterrefType" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" />
    <xsd:attribute name="type" type="xsd:string" />
  </xsd:complexType>
  <xsd:complexType name="parameterrefType">
```

```
  <xsd:attribute name="idref" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="processrefType">
  <xsd:attribute name="idref" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="datasetrefType">
  <xsd:attribute name="idref" type="xsd:string" />
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="datasetsType">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="dataset" type="datasetType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="datasetType">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="datafileref" type="datafilerefType" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="datafilerefType">
  <xsd:attribute name="idref" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="datafilesType">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="datafile" type="datafileType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="datafileType">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string" />
    <xsd:element minOccurs="0" name="directory" type="xsd:string" />
    <xsd:element minOccurs="0" name="description" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="parametersType">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="parameter" type="parameterType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="parameterType">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string" />
    <xsd:element minOccurs="0" name="directory" type="xsd:string" />
    <xsd:element name="parameterfile" type="xsd:string" />
  </xsd:sequence>
```

```
    <xsd:attribute name="id" type="xsd:string" />
  </xsd:complexType>
  <xsd:complexType name="processesType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="process" type="processType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="processType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element minOccurs="0" name="directory" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" />
    <xsd:attribute name="type" type="xsd:string" />
  </xsd:complexType>
</xsd:schema>
```

8.2 An Example Data Ingestion XML

An example data ingestion XML file is shown below. It corresponds to the diagram on the right hand side of Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>

<root id ="An Example Data Ingestion XML">
    <processes>
        <process id="gudrun_java" type="java program">
            <name>GudrunGUI_2.jar</name>
            <directory>GudrunGUI_2</directory>
    </process>
        <process id="purge_det" type="fortran program">
            <name>purge_det.ex</name>
            <directory>GudrunGUI_2</directory>
        </process>
        <process id="gudrun_dcs" type="fortran program">
            <name>Gudrun_dcs.ex</name>
            <directory>GudrunGUI_2</directory>
        </process>
    </processes>
    <parameters>
        <parameter id="param1">
            <parameterfile>
                f1.param
            </parameterfile>
            <name>f1.param</name>
            <directory></directory>
```

```
    </parameter>
    <parameter id="param2">
        <parameterfile>
            f2.param
        </parameterfile>
        <name>f2.param</name>
        <directory></directory>
    </parameter>
    <parameter id="param3">
        <parameterfile>
            f3.param
        </parameterfile>
        <name>f3.param</name>
        <directory></directory>
    </parameter>
    <parameter id="param4">
        <parameterfile>
            f4.param
        </parameterfile>
        <name>f4.param</name>
    </parameter>
    <parameter id="param5">
        <parameterfile>
            f5.param
        </parameterfile>
        <name>f5.param</name>
    </parameter>
    <parameter id="param6">
        <parameterfile>
            f6.param
        </parameterfile>
        <name>f6.param</name>
    </parameter>
</parameters>
<datafiles>
    <datafile id="df1">
        <name>Gudrun_dcs.txt</name>
        <directory>run.SANDALS.Water</directory>
    </datafile>
    <datafile id="df2">
        <name>purge_det.dat</name>
        <directory>run.SANDALS.Water</directory>
    </datafile>
    <datafile id="df3">
        <name>spec.bad</name>
        <directory>run.SANDALS.Water</directory>
    </datafile>
```

```
        <datafile id="df4">
            <name>SLS39631.mgor01</name>
            <directory>run.SANDALS.Water</directory>
        </datafile>
        <datafile id="df5">
            <name>SLS39631.mint01</name>
            <directory>run.SANDALS.Water</directory>
        </datafile>
        <datafile id="df6">
            <name>Detector_withNIMROD.dat</name>
            <directory>StartupFiles.SLS</directory>
<!-- description:  Detector calibration file name -->
        </datafile>
        <datafile id="df7">
            <name>groups_18_clean2.dat</name>
            <directory>StartupFiles.SLS</directory>
<!-- description:  Groups file name -->
        </datafile>
        <datafile id="df8">
            <name>SLSdeadtime.cor</name>
            <directory>StartupFiles.SLS</directory>
<!-- description:             Deadtime constants file name -->
        </datafile>
        <datafile id="df9">
            <name>sears91_gudrun.dat</name>
            <directory>StartupFiles.SLS</directory>
<!-- description:  Neutron scattering parameters file -->
        </datafile>
        <datafile id="df10">
            <name>spectrum000.dat</name>
            <directory>StartupFiles.SLS</directory>
<!-- description:  Filename containing incident beam spectrum parameters -->
        </datafile>
        <datafile id="df11">
            <name>SLS39629.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  NORMALISATION data files -->
        </datafile>
        <datafile id="df12">
            <name>SLS39630.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  NORMALISATION BACKGROUND data files -->
        </datafile>
        <datafile id="df13">
            <name>slsvanadium.bragg</name>
            <directory>StartupFiles.SLS</directory>
<!-- description:           Normalisation differential cross section filename -->
```

```
        </datafile>
        <datafile id="df14">
            <name>SLS39621.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  SAMPLE D2O 25C data files -->
        </datafile>
        <datafile id="df15">
            <name>SLS39637.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  SAMPLE D2O 25C data files -->
        </datafile>
        <datafile id="df16">
            <name>SLS39641.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  SAMPLE D2O 25C data files -->
        </datafile>
        <datafile id="df17">
            <name>SLS39534.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  CONTAINER 1mm TiZr can data files -->
        </datafile>
        <datafile id="df18">
            <name>SLS39542.RAW</name>
            <directory>RawData.SANDALS</directory>
<!-- description:  CONTAINER 1mm TiZr can data files -->
        </datafile>
        <datafile id="df19">
            <name>GNUplot.plt</name>
            <directory>run.SANDALS.Water</directory>
<!-- description: Gnuplot files -->
        </datafile>
    </datafiles>
    <datasets>
        <dataset id="d1">
            <datafileref idref="df1"/>
            <datafileref idref="df6"/>
            <datafileref idref="df7"/>
            <datafileref idref="df8"/>
            <datafileref idref="df9"/>
            <datafileref idref="df10"/>
            <datafileref idref="df11"/>
            <datafileref idref="df12"/>
            <datafileref idref="df13"/>
            <datafileref idref="df14"/>
            <datafileref idref="df15"/>
            <datafileref idref="df16"/>
            <datafileref idref="df17"/>
```

```
            <datafileref idref="df18"/>
            <datafileref idref="df19"/>
        </dataset>
        <dataset id="d2">
            <datafileref idref="df2"/>
        </dataset>
        <dataset id="d3">
            <datafileref idref="df3"/>
        </dataset>
        <dataset id="d4">
<!-- description: -->
            <datafileref idref="df4"/>
            <datafileref idref="df5"/>
        </dataset>
        <dataset id="d5">
            <datafileref idref="df19"/>
        </dataset>
    </datasets>
    <investigations>
        <investigation id="i1" type="analysis">
            <processref idref="gudrun_java"/>
            <datasetref idref="d5" type="others"/>
            <datasetref idref="d1" type="output"/>
            <datasetref idref="d2" type="output"/>
        </investigation>
        <investigation id="i2" type="analysis">
            <datasetref idref="d2" type="input"/>
            <processref idref="purge_det"/>
            <datasetref idref="d3" type="output"/>
        </investigation>
        <investigation id="i3" type="analysis">
            <datasetref idref="d1" type="input"/>
            <datasetref idref="d3" type="input"/>
            <processref idref="gudrun_dcs"/>
            <datasetref idref="d4" type="output"/>
        </investigation>
    </investigations>
    <studies>
        <study id="s1">
            <investigationref idref="i1" />
            <investigationref idref="i2" />
            <investigationref idref="i3" />
        </study>
    </studies>
</root>
```

8.3 ICAT-Personal Database Schema

This section presents a database schema (MySQL) based on the extended CSMD model for the ICAT-Personal implementation.

```
drop database if exists icatlite;
create database icatlite;

use icatlite;

create table process (
id bigint(20) not null auto_increment primary key,
type varchar(20),
directory varchar(100),
name varchar(255),
location varchar(255),
link varchar(255),
creationtime timestamp(8) default now(),
creator int(20)
);

create table dataset (
id bigint(20) not null auto_increment primary key,
creationtime timestamp(8) default now(),
creator int(20)
);

create table investigation (
id bigint(20) not null auto_increment primary key,
process bigint(20) not null,
type varchar(20),
creationtime timestamp(8) default now(),
creator int(20),
foreign key (process) references process(id) on update cascade on delete restrict
);

create table datafile (
id bigint(20) not null auto_increment primary key,
dataset bigint(20) not null,
directory varchar(255),
name varchar(255),
location varchar(255),
description varchar(255),
creationtime timestamp(8) default now(),
creator int(20),
link varchar(255),
index (dataset),
foreign key (dataset) references dataset(id) on update cascade on delete restrict
```

```
);

create table parameter (
id bigint(20) not null auto_increment primary key,
directory varchar(100),
location varchar(255),
name varchar(255),
creationtime timestamp(8) default now(),
creator int(20)
);

create table datafile_parameter (
datafile bigint(20) not null,
parameter bigint(20) not null,
creationtime timestamp(8) default now(),
creator int(20),
index(datafile, parameter),
foreign key (datafile) references datafile(id) on update cascade on delete restrict,
foreign key (parameter) references parameter(id) on update cascade on delete restrict,
primary key (datafile, parameter)
);


create table dataset_investigation (
dataset bigint(20) not null,
investigation bigint(20) not null,
type varchar(10),
creationtime timestamp(8) default now(),
index(dataset, investigation),
creator int(20),
foreign key (dataset) references dataset(id) on update cascade on delete restrict,
foreign key (investigation) references investigation(id) on update cascade on delete restrict
primary key (dataset, investigation)
);


create table dataset_parameter (
dataset bigint(20) not null,
parameter bigint(20) not null,
creationtime timestamp(8) default now(),
index(dataset, parameter),
creator int(20),
foreign key (dataset) references dataset(id) on update cascade on delete restrict,
foreign key (parameter) references parameter(id) on update cascade on delete restrict,
primary key (dataset, parameter)
);
```

```
create table investigation_parameter (
investigation bigint(20) not null,
parameter bigint(20) not null,
creationtime timestamp(8) default now(),
creator int(20),
index(investigation, parameter),
foreign key (investigation) references investigation(id) on update cascade on delete restrict
foreign key (parameter) references parameter(id) on update cascade on delete restrict,
primary key (investigation, parameter)
);

create table investigator (
id bigint(20) not null auto_increment primary key,
creationtime timestamp(8) default now(),
creator int(20)
);

create table investigator_investigation (
investigator bigint(20) not null,
investigation bigint(20) not null,
creationtime timestamp(8) default now(),
index(investigator, investigation),
creator int(20),
foreign key (investigator) references investigator(id) on update cascade on delete restrict,
foreign key (investigation) references investigation(id) on update cascade on delete restrict
primary key (investigator, investigation)
);

create table study (
id bigint(20) not null auto_increment primary key,
manager bigint(20) default null,
xml blob,
dot blob,
name varchar(200),
creator int(20),
creationtime timestamp(8) default now()
);

create table study_childstudy (
parent_study bigint(20) not null,
child_study bigint(20) default null,
creationtime timestamp(8) default now(),
creator int(20),
index(parent_study, child_study),
foreign key (parent_study) references study(id) on update cascade on delete restrict,
foreign key (child_study) references study(id) on update cascade on delete restrict,
primary key (parent_study, child_study)
```

```
);

create table study_investigation (
study bigint(20) not null,
investigation bigint(20) not null,
creationtime timestamp(8) default now(),
creator int(20),
index(study, investigation),
foreign key (study) references study(id) on update cascade on delete restrict,
foreign key (investigation) references investigation(id) on update cascade on delete restrict
primary key (study, investigation)
);

create table studyManager (
id bigint(20) not null auto_increment primary key,
creationtime timestamp(8) default now(),
creator int(20)
);
```

8.4 ICAT-Personal sourceforge

ICAT-Personal sourceforge home `http://sourceforge.net/projects/icatlite/`

ICAT-Personal sourceforge svn `http://icatlite.svn.sourceforge.net/viewvc/icatlite/`

ICAT-Personal sourceforge wiki `http://sourceforge.net/apps/mediawiki/icatlite/index.php?title=Main_Page`

References

[1] Bechhofer, S., De Roure, D., Gamble, M., Goble, C. and Buchan, I. (2010) Research Objects: Towards Exchange and Reuse of Digital Knowledge. In: The Future of the Web for Collaborative Science (FWCS 2010), April 2010, Raleigh, NC, USA.

[2] Simon J. Coles, Jeremy G. Frey, Michel B. Hursthouse, Mark E. Light, Andrew J. Milsted, Leslie A. Carr, David DeRoure, Christopher J. Gutteridge, Hugo R. Mills, Ken E. Meacham, Michael Surridge, Elizabeth Lyon, Rachel Heery, Monica Duke, and Michael Day, *An E-Science Environment for Service Crystallography - from Submission to Dissemination*, J. Chem. Inf. Model., 2006, 46(3), pp.1006 - 1016.

[3] Damian Flannery, Brian Matthews, Tom Griffin, Juan Bicarregui, Michael Gleaves, Laurent Lerusse, Roger Downing, Alun Ashton, Shoaib Sufi, Glen Drinkwater, Kerstin Kleese, *ICAT: Integrating Data Infrastructure for Facilities Based Science*, e-science, pp.201-207, 2009, Fifth IEEE International Conference on e-Science, IEEE Computer Society.

[4] M Folk, A Cheng, K Yates (1999) HDF5: A file format and I/O library for high performance computing applications, Proceedings of Supercomputing'99, ACM SIGARCH and IEEE, (Portland, OR), Nov. 1999.

[5] Ian T. Foster. *The virtual data grid: a new model and architecture for data-intensive collaboration*. In SSDBM 2003: Proceedings of the 15th international conference on Scientific and statistical database management, Washington, DC, USA, 2003. IEEE Computer Society.

[6] Boris Glavic and Klaus R. Dittrich. *Data Provenance: A Categorization of Existing Approaches*. In Datenbanksysteme in Business, Technologie und Web (BTW 2007), pp. 227 - 241, 2007.

[7] P. Klosowski, M. Koennecke, J. Z. Tischler and R. Osborn, *NeXus: A common format for the exchange of neutron and synchroton data*, Physica B: Condensed Matter, Vol 241-243, Dec 1997, pp151-153, Proceedings of the International Conference on Neutron Scattering.

[8] Brian Matthews, Shoaib Sufi, Damian Flannery, Laurent Lerusse, Tom Griffin, Michael Gleaves, Kerstin Kleese. *Using a Core Scientific Metadata Model in Large-Scale Facilities*. The 5th International Digital Curation Conference, London, England, 2-4 December 2009.

[9] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat and Peter Li, *Taverna: a tool for the composition and enactment of bioinformatics workflows*, Bioinformatics, Vol. 20(17) 2004, pp3045-3054.

[10] R Rew, G Davis, *NetCDF: an interface for scientific data access IEEE Computer Graphics and Applications*, 10(4), 76-82, 1990.

[11] Enrico Scalavino, Vaibhav Gowadia, and Emil C. Lupu (2010) A Labelling System for Derived Data Control, in Sara Foresti and Sushil Jajodia (Eds.) Data and Applications Security and Privacy XXIV: Proceedings of DBSec 2010, the 24th Annual IFIP WG 11.3 Working Conference, LNCS, Springer-Verlag:Berlin.

[12] Andrew Shepherd (2001) Hierarchical task analysis, Taylor & Francis: London.

[13] Shoaib Sufi and Brian Matthews, *A Metadata Model for the Discovery and Exploitation of Scientific Studies*. In Domenico Talia, Angelos Bilas and Marios D. Dikaiakos (Eds.) Knowledge and Data Management in GRIDs, 2007, pp135-149, Springer: Berlin.

[14] MG Tucker, DA Keen, MT Dove, AL Goodwin and Q Hui. *RMCProfile: Reverse Monte Carlo for polycrystalline materials*. Journal of Physics: Condensed Matter 19, art no 335218 (16 pp), 2007, available at: `http://wwwisis2.isis.rl.ac.uk/rmc/`.

[15] Erica Yang. *Martin Dove's RMC Workflow Diagram*. Project Requirement Report (supplementary report) for the I2S2 project, July 2010. Available at: `https://www.jiscmail.ac.uk/cgi-bin/filearea.cgi?LMGT1=I2S2&f=/Deliverables/RequirementsReport`.

[16] Yu, J. and Buyya, R. 2005. *A taxonomy of scientific workflow systems for grid computing*, SIGMOD Rec. 34, 3 (Sep. 2005), pp44-49.

[17] Manjula Patel. *D1.1 Requirements Report/Deliverable from the JISC I2S2 Project*, July 2010, available at `http://www.ukoln.ac.uk/projects/I2S2/documents/I2S2-WP1-D1.1-RR-Final-100707.pdf`.

[18] Erica Yang. *D1.1 Requirements Report - A Supplementary Report from the JISC I2S2 Project*, July 2010, available at `http://www.ukoln.ac.uk/projects/I2S2/documents/ISIS%20RMC%20workflow.pdf`.