JISC

Project Document Cover Sheet

Project Information			
Project Acronym	I2S2		
Project Title	Infrastructure for Integration in Structural Sciences		
Start Date	1st Oct 2009	**End Date**	31st March 2011
Lead Institution	Universities of Bath and Southampton		
Project Director	Liz Lyon (UKOLN)		
Project Manager & contact details	Manjula Patel 01225 386547; m.patel@ukoln.ac.uk		
Partner Institutions	Universities of Bath, Southampton, Cambridge; STFC; Charles Beagrie Ltd.		
Project Web URL	http://www.ukoln.ac.uk/projects/I2S2/		
Programme Name (and number)	Managing Research Data (Research Data Management Infrastructure)		
Programme Manager	Simon Hodson		

Document Name			
Document Title	Pilot Implementation		
Reporting Period	*N/A*		
Author(s) & project role	Erica Yang, Brian Matthews (STFC Rutherford Appleton Laboratory)		
Date	31st March 2011	**Filename**	I2S2-WP3-D3.3b-PilotImplementation.doc
URL			
Access	Public		x General dissemination

Infrastructure for Integration in Structural Sciences

D3.3 Pilot Implementation

Work Package 3

April 2010 – March 2010

JISC I2S2 Project

Document Details

Author:	Erica Yang, Brian Matthews (STFC Rutherford Appleton Laboratory)
Date:	31st March. 2011
Version:	0.4
File Name:	I2S2-WP3-D3.3.-PilotImplementation.doc
Notes:	

Acknowledgements

The Infrastructure for Integration in Structural Sciences (I2S2) Project is funded by the UK's Joint Information Systems Committee (JISC); the project manager is Simon Hodson. The I2S2 project team comprises:

- Liz Lyon (UKOLN, University of Bath & Digital Curation Centre)
- Manjula Patel (UKOLN, University of Bath & Digital Curation Centre)
- Simon Coles (EPSRC National Crystallography Centre, University of Southampton)
- Martin Dove (Earth Sciences, University of Cambridge)
- Peter Murray-Rust (Chemistry, University of Cambridge)
- Brian Matthews (Science & Technology Facilities Council)
- Erica Yang (Science & Technology Facilities Council)
- Juan Bicarregui (Science & Technology Facilities Council)
- Neil Beagrie (Charles Beagrie Ltd.)

The authors would like particularly like to thank Prof Martin Dove from Earth Sciences at the University of Cambridge, Simon Coles from the UK National Crystallography Service, and Dr. Alan Soper and Dr. Matt Tucker from STFC ISIS facility for providing case studies of the scientific process on STFC facilities .

A shorter version of this report has been published at the IEEE e-Science conference 2010, Brisbane, Australia [23]. A version without appendices has been submitted for publication in *Future Generation Computer Systems.*

Executive Summary

The *Infrastructure for Integration in Structural Sciences (I2S2) Project* is funded under the Research Data Management Infrastructure strand of the JISC's Managing Research Data Programme, with duration of 18 months (Oct 2009 to March 2011).

One of the main aims of the project is to investigate the research and data management infrastructure needs of researchers in the Structural Sciences (incorporating a number of disciplines including Chemistry, Physics, Materials, Earth, Life, Medical, Engineering, and Technology). We define research infrastructure to encompass physical, informational and human resources essential for researchers to undertake high-quality research, including: tools, instrumentation, computer systems and platforms, software, communication networks, technical support (both human and automated); documentation and metadata.

This deliverable describes the pilot implementation of the I2S2 project.

Document Revision History		
Version	**Date**	**Comments**
0.1	15th October 2010	Scope and initial outline
0.2	18th October 2010	Added data ingestion XML schema, XML example, MySQL database schema
0.3	20th October 2010	Added browser interface, final remarks
0.4	23th March 2011	Revisit and revise
1.0	6th April 2011	Final release

Contents

1 Introduction **6**

2 Core Scientific MetaData model **7**

3 Derived Data in the Analysis Process **10**
3.1 Background 10
3.2 Data Analysis 11
3.2.1 Data reduction 12
3.2.2 Initial structural model generation 12
3.2.3 Model fitting 12
3.3 Discussion 13

4 An Enhanced CSMD **13**
4.1 Adding a SoftwareExecution Investigation Type 15
4.2 Linking Software to SoftwareExecution 16
4.3 Linking SoftwareExecutions to Datasets 17
4.4 Associating parameters with a SoftwareExecution 17
4.5 Re-introducing Study and Nested Study 17
4.6 Observation 18

5 ICAT-personal: A Pilot Implementation **18**
5.1 System Architecture 18
5.2 Derived Data Management 20
5.2.1 Data ingestion 20
5.2.2 Data Browsing 22
5.2.3 Data Restoration 22

6 Discussion and Future Work **24**

7 Final Remarks **26**
7.1 Stakeholder Engagement 26
7.2 Stakeholder Feedbacks 26

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, possibly in combination with results from elsewhere.

The Core Scientific MetaData model (CSMD) [20, 10] 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 [5] which is used by the ISIS neutron source[1] and the Diamond Light Source (DLS)[2], both operated at the Harwell Science and Innovation Campus in the UK. The DLS synchrotron generates brilliant beams of light, from infra-red to X-rays, which are used in a wide range of applications, from structural biology through fundamental physics and chemistry to cultural heritage. The ISIS source generates beams of neutrons and muons used to investigate the properties of materials at the scale of atoms for research into subjects ranging from clean energy and the environment, pharmaceuticals and health care, through to nanotechnology, materials engineering and IT. The two target stations of the ISIS neutron source host 30 beamlines with their associated instruments, while DLS currently hosts 13 instruments on separate beamlines. The use of these facilities is not limited to a small coterie of specialists, but between them these instruments are used by many thousands of experimental scientists each year from around the world. As similar large facilities are developed in other countries the data sets they create are becoming more common, and it becomes more urgent to capture that data, and to ensure that all stages of its analysis are accurately recorded. Consequently, facilities such as the Institut Laue-Langevin (ILL)[3] are also adopting the ICAT infrastructure, and the PANDATA initiative[4] is developing best practice in data management across facilities internationally.

Data cataloguing systems support access to scientific data, but the present ICAT only catalogues the raw data produced by the facility, while derived data is managed locally by the scientist carrying out the analysis at the facility or in their home institution. This is on an ad hoc basis, and these intermediary derived data sets are not archived for other purposes. Thus the support for the intended users is partial.

In order to improve the support offered by the facilities data management tool such as ICAT, its underlying data model, CSMD needs to be extended. Currently, 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.

Bioscientists have used workflow tools to capture and automate the flow of analyses and the production of derived data for many years [14] and can now automatically run many computational workflows [24]. In other structural sciences, such as chemistry and Earth sciences, the management of derived data is less mature, workflows are not standardised and

[1] http://www.isis.stfc.ac.uk

[2] http://www.diamond.ac.uk

[3] http://www.ill.eu

[4] PANDATA Photon and Neutron Data Infrastructure. http://www.pan-data.eu/Main_Page

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-analysis 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 [8], 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 report, 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, which are incorporated into the I2S2 Information Model, before a pilot implementation of the extended 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.

In appendices, we give details of the tools and technologies required to support the ICAT-Personal system.

2 Core Scientific MetaData model

The Core Scientific MetaData model (CSMD) [20] 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, and has proved a robust system.

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. This is important to both facility

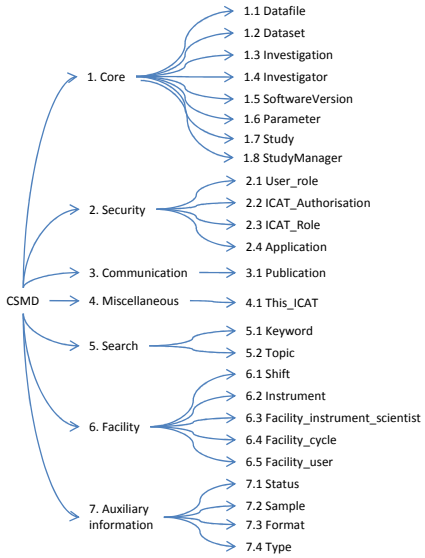

Figure 1: A classification of the concepts in CSMD

providers who host a wide range of scientific facilities and to users who utilize multiple facilities. Metadata are also crucial for scientists other than the ones who design the equipment or run the experiment, to interpret, understand and make use of the data.

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) [17] and Hierarchical Data Format (HDF) [6] are well defined formats used by many laboratories, while NeXus [9], 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 a data generation activity. For example an investigation may represent a particular allocation of time on an instrument to a scientist for the analysis of a sample of a material, which may generate a number of data sets each collected at a different experimental parameter setting. 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 on a sample and a related computer simulation of the experiment, could 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, ICAT 3.3[5], is based on the CCLRC Scientific Metadata Model v2 [20] 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 The concepts which are central to scientific data management. Capturing the data outputs involve four data objects: datafile, dataset, investigation, and study. A datafile corresponds to 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 described above. Other core concepts include the Investigator and StudyManager, representing people associated with an investigation and a study, respectively. The Process concept[6] captures an activity that produces or consumes data objects, while the Parameter concept captures some value which provides context to the data production process, such as environmental characteristics, instrument settings, or measured quantities.

Search Classifiers which can be assigned to data to facilitate the search and discovery of core concepts.

Communication Concepts which link between data and other research outputs so that the provenance of a research publication can be traced back to the data holdings.

Security Concepts which are used to enforce access control policies on the data holdings. These may vary according to the security context of the facility.

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

Facility Concepts related to facilities are introduced to capture the contextual information associated with the (raw) data collection process, such as which facility and instrument was used, which cycle, shift or run of the facility, the instrument-scientist (a specialised role in a large-scale facility) was involved, additional safety information. These concepts are specialised to facility usage, although there are analogues in other experimental contexts, such as university laboratory experiments.

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

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

Auxiliary Information Specific information associated with data holdings. It is currently being used to store information related to *raw* data files, such as the sample under analysis, further parameters (e.g. temperature, humidity), and file format. But it should be possible to extend or adapt these concepts 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 report. The latter does not directly relate to the report, and we shall not expand on this aspect 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) [4] to determine the structure of atoms in solids using X-ray diffraction. The other two experiments aim to determine the structure of atoms of matters (e.g. liquids or solids) using neutron techniques: one uses the neutron diffraction[7] provided by the GEM instrument[8] and the other small angle neutron scattering[9] offered by the Sandals instrument[10]. 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 [4]. 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 collects the raw data to the point where the resultant data are obtained. Both workflows overlap in their data correction

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

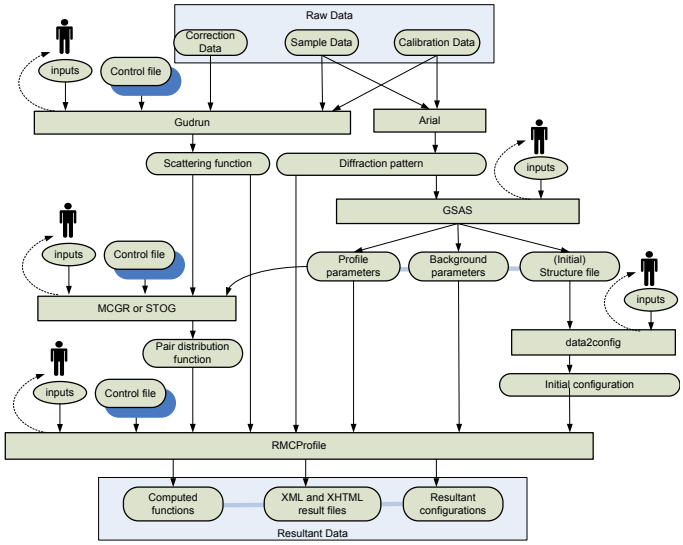

Figure 2: The RMC data analysis flow diagram

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 [19] 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 [22]. The RMC method is probabilistic, which means that a) it can only deliver an 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 two programs: `Gudrun`, a Fortran program with a Java GUI, and `Arial`, a IDL program. The outputs from `Gudrun`[11] are a set of scattering functions, one for each bank of detectors. For `Arial`[12], 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 byway. 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` [21] program implementing 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.

[11] `http://www.isis.rl.ac.uk/disordered/Manuals/gudrun/gudrun_GEM.htm`
[12] `http://www.isis.stfc.ac.uk/instruments/osiris/data-analysis/ariel-manual9033.pdf`

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 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.
- the *people* who drive the programs.

In a production system, the people element is as important, if not more important, than the others, for accountability, security, attribution, and archival reasons. However, because this element has been well captured in the current CSMD model and implemented in ICAT version 3.3, we shall not include it in this presentation of the extended CSMD model.

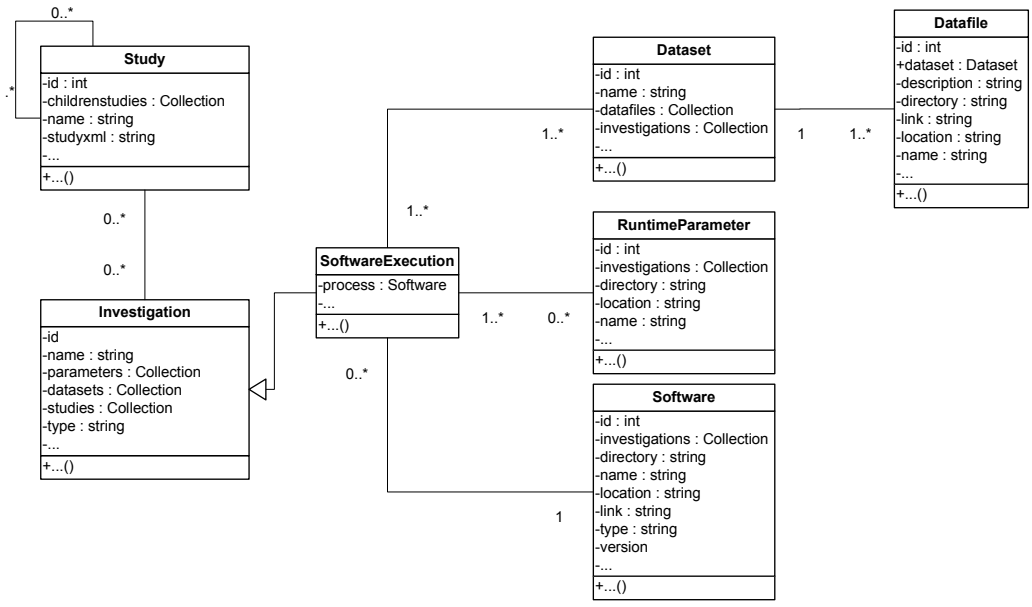

Figure 3: An Extended CSMD Object Model for Supporting Derived Data

Figure 3 is a (concise) object model depicting the extensions and modifications to the core of the existing CSMD model to support derived data[13]. In order to keep it digestible, the connections between the entities presented here and those in the current model[14] are omitted. For example, in the current model, the entities - `datafile` and `dataset`, are linked to the entity `parameter` (via `datafileParameter` and `datasetParameter`) to describe, the set of instrument parameter settings related to them.

An object model is an *abstraction* of the objects involved and the relationship between the objects. In real world systems, the object model is manifested[15] as data models, which can be implemented in all kinds of object-oriented programming languages (e.g. Java, C++ or C#), relational databases (e.g. MySQL, Oracle), RDF, XML Schemas, and even an XML databases (e.g. eXist).

A data model can also be implemented in scripting and interpretive programming languages, such as Perl, PHP, or even Javascript, in a non-object oriented fashion, although this is not recommended because it will lose the benefits of object-orientation. However, this highlights an important point: the model can be mapped to various data models, as long as they all conform to the same object model. The benefit of doing so is to enable the interoperation among the implementations upon the data models. In practical terms, this means that the derived data provenance captured in one data model, for example, implemented in Java, can be pushed to a data repository implemented in another data model, for example, implemented as a relational database. Conversely, the provenance stored in a database can be presented in another data model, for example, in a XML schema. The transformation

[13]For a complete set of the attributes and operations of each object, readers are referred to the sourceforge website: `http://icatlite.sourceforge.net/`.

[14]ICAT 3.3 Database schema: `http://icatproject.googlecode.com/svn/icat3_api/trunk/icat3-database/IcatDB/jdeveloper/icat/schemadiagrams/model/icat_v3.png`

[15]Hereafter, we use the following terms interchangeably: manifest and map.

between the data models can be facilitated by, for example, Java. In the next section, we shall present our implementation of such an example, showcasing how these mappings can be realised and are used to the benefit of capturing derived data provenance.

Specifically, the extensions and modifications are introduced to the model underpinning ICAT 3.3 along the following directions:

- adding a *SoftwareExecution* subclass of investigation;
- linking program to a software execution;
- linking software executions with datasets;
- associating parameters with a software execution;
- re-introducing the *study*; and
- introducing study nesting.

We shall now describe these extensions and the rationales behind them.

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 activities: measurements, experiments, and simulations [10]. None relates to the data handling activities in an data analysis process.

A new type of investigation, *SoftwareExecution* is introduced to model the *executions* of *one data analysis task* in the process. In modern research, a task is typically the running of a piece of software. Our model does *not* mandate what a piece of software might be. It can be a system of programs, for example `MATLAB`, which consist of many programs implementing various functionalities. Or, it can be one program implementing a specific function as part of a system of programs, for example, a Fast Fourier transformation function in `MATLAB`. The decision of such is left to the users of our software, the researchers, because only they know about what is the most suitable and useful representation of the execution of a task in their data analysis process.

As illustrated in Figure 4, the SoftwareExecution concept captures the scenario that a piece of software is executed many times, yielding results, each corresponding to a combination of the following components:

- the software used for the execution,
- the parameters (e.g. the settings of experiments or simulations where an input dataset is obtained, or the settings for an analysis methodology),
- the input datasets (e.g. readings captured from instruments or simulations), and
- the output datasets (e.g. the results of an analysis function).

We use an analogy to explain the rationale behind the modelling of this concept. A (research) problem can be analogous to a 'puzzle', a task the building of a 'jigsaw' in order to solve one part of the 'puzzle', and the execution of a task a 'jigsaw' built with one combination of parameters, datasets, and program settings to solve that part of the 'puzzle'. Every

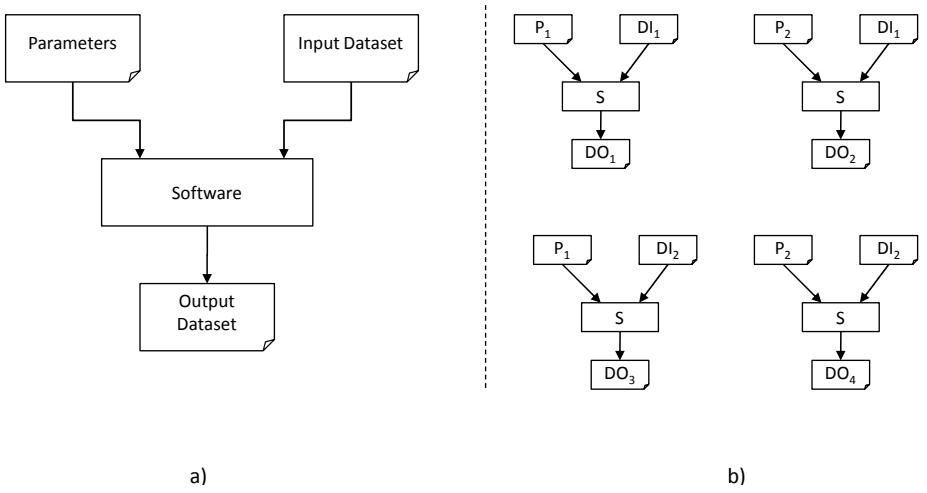

Figure 4: a) The SoftwareExecution Concept; b) Four SoftwareExecutions depicting the scenario of running the same piece of software four times, each taking a set of parameters and input datasets, and yielding different output datasets

researcher has an idea of what tasks are required to solve a problem and how to execute the tasks. Because research is often open-ended and iterative, the usefulness and impact of an (early-stage) analysis is often difficult to judge. Therefore, keeping track of all the potentially useful 'jigsaws' along a research trial is not only important but also valuable to researchers.

'Jigsaws' can be built by the same person or independently by different people in solving a part of a puzzle. Once the 'jigsaws' are built, they can be put together to a) either lead to different solutions to the same problem; or b) form new 'jigsaws' to solve a different problem, leading to unexpected new discoveries. For the former, judging which solution is the best is an issue beyond the scope of this paper. However, we believe that capturing the different 'paths' presented in derived data provenance trails can be a powerful approach for addressing research problems. Sections 4.5 present how we put these 'jigsaws' together to form a big picture of a solution to solve a 'puzzle'.

4.2 Linking Software to SoftwareExecution

SoftwareExecution is a runtime notion meaning that it is not only associated with a 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,[16] one or more input datasets, one or more output datasets, and zero or more parameters to the program. Capturing computer software related to data is a complex issue; see for example [11, 12] for a consideration of the complexity of characterising software. For example, even with the same software, there are often different versions in existence and they may or may not compatible with each other, and may behave differently in different execution environments. In this paper, we take a simplified view of software by focusing on the data aspect of the derived data provenance

[16]the program could be formed from a number of programs linked together, but intermediate input and outputs are not persisted and catalogued.

trails. In practical terms, this means that the inputs (parameters or datasets) used for one version of a program may not be workable with another one. We are aware of this issue and consider it a topic for further research.

4.3 Linking SoftwareExecutions to Datasets

Software executions are linked to datasets to capture their context in the provenance process as follows.

Input and output datasets

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

Associating Multiple SoftwareExecutions to Input Datasets

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 a 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 Re-introducing Study and Nested Study

The *Study* is a concept designed for grouping related investigations together to capture a common intentionality, such as a research programme, or the analysis of a particular compound. It is a part of the existing CSMD model, although not currently implemented in the current ICAT 3.3 as that it tailored to capture the generation of raw data from a facility instrument, thus each investigation is the unit of intentionality and investigation and study are in a one to one relationship, so study is seen as superfluous. However, this is not the case when we need to consider derived data, and we use it to capture the means by which SoftwareExecutions are related to each other. For example, in the analysis process, a study is used to group SoftwareExecutions together in a particular order. The ordering depicts explicitly the relationship between the investigations reflecting the sequence of the data handling activities involved in a scientific process.

Through a study, SoftwareExecutions 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. The exended study concept provides an end-to-end support for data management covering the experimental data gathered from instruments, to intermediate derived data generated during the analysis process, to the resultant data finally appeared in papers.

A nested study is a notion for nesting a study inside one or more *other* studies. When a study nests inside another study, the former is called a `childstudy` and the latter a `parentstudy`. When a study is a child of several studies, the parent studies share the same task in their analysis process.

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. This last category is particularly interesting because it opens up the possibility of comparing existing analyses with future analyses. However, this is beyond the scope of this report.

4.6 Observation

Although the extended model we have presented has been developed for solving the data management problem in context of "big science" carried out in large-scale facilities, we believe that it is a discipline neutral approach that can be used to solve derived data management problems in other disciplines, and also in the small-scale context found in the university research laboratory.

5 ICAT-personal: A Pilot Implementation

A pilot implementation of the extended CSMD model has been developed for the purpose of supporting the capturing, cataloguing and storing of derived data for *individual researchers*, typically working in university research laboratories. Such researchers may have limited capability for systematic data management, and thus this approach offers a rigorous but feasible method to capture data generated in laboratory analysis and make it retrievable and reusable. Because it is designed to tackle data management problems of individuals, it is named **iCat-Personal**. It is available through the sourceforge website.[17] 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 shall describe its design and development focusing on the current capabilities of the implementation.

5.1 System Architecture

Figure 5 illustrates the system architecture of iCat-Personal, an Java-based implementation for data ingestion and restoration, and a PHP-based web application for data browsing. It

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

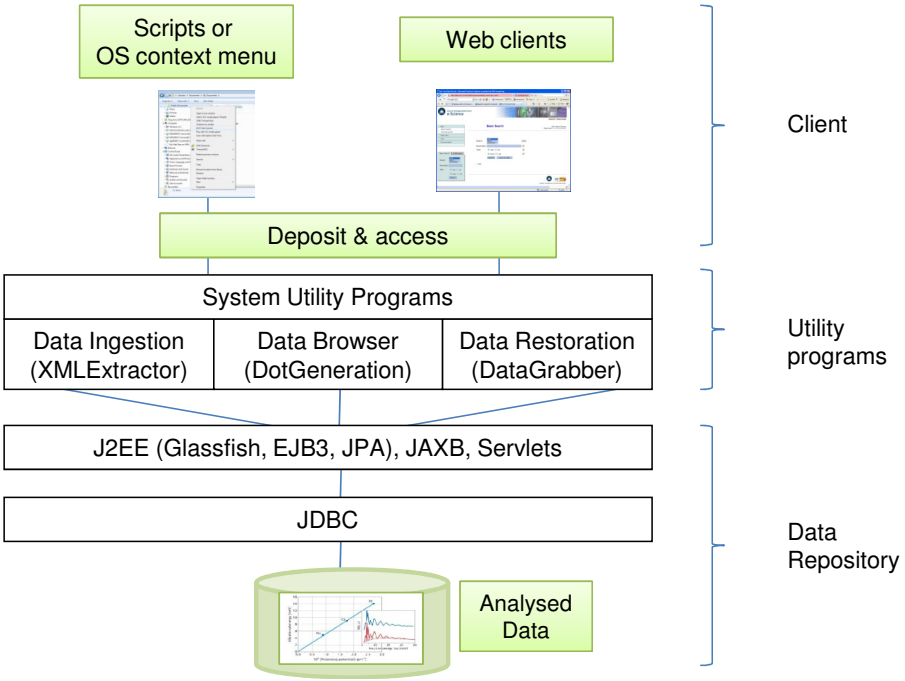

Figure 5: ICAT-personal System Architecture

consists of three layers: clients, utility programs, and a repository. Two types of clients are supported: command line scripts for getting the data into the repository and restoring data from a repository, and a web browser interface for browsing and navigating derived data stored in the repository. The utility program transforms the data sent from the script and ingest them into a persistent data repository through Java entity beans over a Hibernate-based persistence layer underpinning by a MySQL relational database.

Three key functions of data management are supported, they are: data ingestion, data browsing, and data restoration; all are underpinned via a data catalogue. The targeted audience of this implementation is individual scientists who need a data management tool to assist their research. Future research will investigate how well the model accommodates issues of remote data storage (instead of storing the data locally on the same computer as the source of the derived data), 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 object model presented in the previous section is mapped into two data models: a XML schema (see Appendix A for complete transcript of the XML Schema) and a database schema (see Appendix B for a complete transcript of the Database Schema). The former is used by the client to guide the ingestion of derived data provenance into an iCAT-Personal data repository; whilst the latter is the database structure underpinning the repository. We use the Gudrun program in the RMC workflow to explain their role in managing derived data.

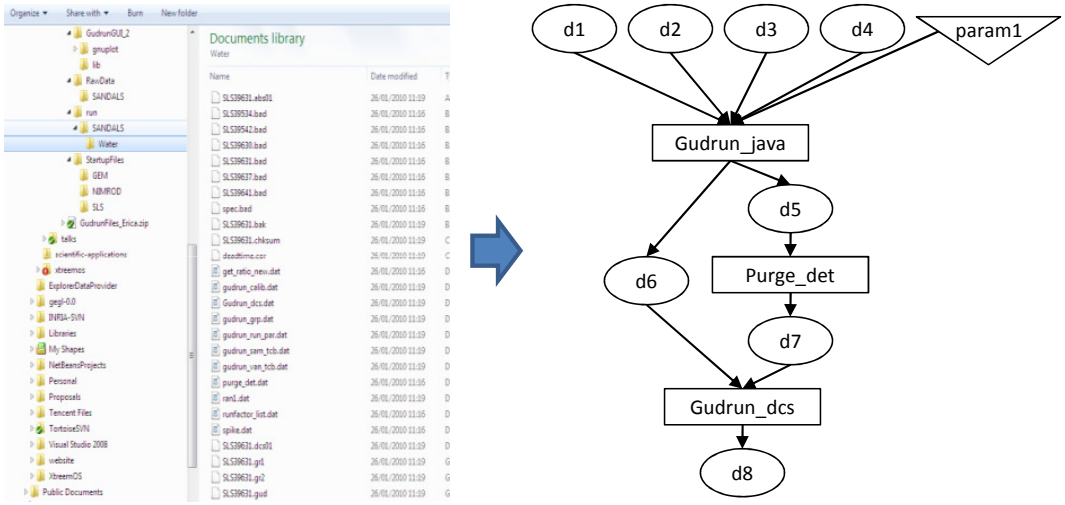

Figure 6: Derived Data Management: An Example

5.2 Derived Data Management

Figure 6 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, 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 with the RMC analysis process, many scientific analyses involve several programs. Scientists often end up with many working directories, each storing the data resulted from one execution. Managing such 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 relationships between the executions. Again, until the final results are available, all the links (which often mean many directories, and sub-directories) have to be kept.
- Different scientists have their own way of keeping the parameters and settings of each execution. These may be stored in annotation files in the working directory, or in a paper lab notebook for example. Without the parameters, it is hard to understand the outputs from the programs or continue other researchers' analyses.

As a consequence, 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 Figure 6 is a *structured representation of the executions* of the programs involved in Gudrun. The structure represents how the executions of different pro-

```
1    <processes><process id="gudrun_java" type="java program">...</process>...</processes>
2   <parameters><parameter id="param1"> ... </parameter> ... </parameters>
3    <datafiles>
4        <datafile id="df1"><name>Gudrun_dcs.txt</name> ... </datafile>
5        ...
6        <datafile id="df18"><name>SLS39542.RAW</name> ... </datafile>
7    </datafiles>
8    <datasets>
9     <dataset id="d1">
10            <datafileref idref="df1"/>
11            <datafileref idref="df6"/>
12            ...
13       </dataset>
14       <dataset id="d2">
15           <datafileref idref="df2"/>
16       </dataset>
17       ...
18   </datasets>
19   <investigations>
20       <investigation id="i1" type="softwareexecution">
21           <processref idref="gudrun_java"/>
22           <datasetref idref="d1" type="output"/>
23           <datasetref idref="d2" type="output"/>
24       </investigation>
25       <investigation id="i2" type="softwareexecution">
26           <datasetref idref="d2" type="input"/>
27           <processref idref="purge_det"/>
28           <datasetref idref="d3" type="output"/>
29       </investigation>
30       ...
31   </investigations>
32   <studies>
33       <study id="s1">
34           <name>Gudrun Data Reduction Study</name>
35           <investigationref idref="i1" />
36           <investigationref idref="i2" />
37           <investigationref idref="i3" />
38       </study>
39   </studies>
```

Figure 7: Example XML file capturing a Gudrun execution process

grams 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 5. This process is called ICAT-Personal data ingestion. It is guided by an ICAT-Personal XML schema compliant XML file. Figure 7 illustrates a snippet of such an XML file, which captures:

- programs in the process (line 1),
- inputs, including data files and parameters (or parameter files), to and outputs (e.g. data files, plots) from the programs (lines 2 - 7),
- datasets, the logical groupings of the datafiles (lines 8 - 18),
- SoftwareExecutions and their linkage with datasets (lines 19 - 31), and
- the order of the software executions (lines 32 - 39)

A complete transcript of this XML file is given in Appendix B. The XML Schema used to define this XML format is given in Appendix A.

A Java based `XMLExtractor` program, built upon the JAXB technology, is used to parse the XML file and generates Java entity beans from the XML. The beans are ingested into the database via a Hibernate-based persistence layer over MySQL.

5.2.2 Data Browsing

An ICAT-Personal tool, named DotGeneration, provides data browsing capability. It takes an ICAT-Personal data ingestion XML file, transforms it into a Graphviz[18] dot file, and generates a flow diagram as depicted on the right hand side of Figure 6. In the current database schema, the dot file and the corresponding snippet of the XML are also ingested into the database. A PHP based web application has also been developed to display the relationship between programs, parameters, datafiles, datasets, SoftwareExecutions and studies.

Datasets, labelled as `d1` to `d8` in the Figure 6, are used to capture the relationship between data files produced or consumed by one execution. Among all the input data files to Gudrun_java, four datasets are used, 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 8 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 ICAT-Personal 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.

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

Figure 8: The Gudrun example in a Web browser view with details of data files involved

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 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 crystallography 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[19]. This common model combines the expressive power in describing the context and structure of data collections offered by the CSMD with the conceptual framework for modelling experimental process planning and enactment offered by the ORE-CHEM [3], and models a wide range of activities within the scientific life cycle.

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 changes to the CSMD.

The changes proposed to the CSMD capture the source of the data, and the transformation process that is has gone through, and reflects the Open Provenance Model [13], but the implementation does not provide a comprehensive provenance management system. [8] 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 [7].

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. However, such a query system would require the automated splitting of datasets to isolate the transformation data and the subsequent merging of that transformation data for each stage into a single provenance item describing the overall process. Such provenance records would then have to be open to be queried for co-occurring transformations or transformation parameter values Such a system is beyond the scope of the current development, although it could not be attempted without the work

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

presented here to build upon, and they could be a topic for future work.

The tools considered in the analysis simply consume and produce files. Some more sophisticated, but very commonly used, tools such as Chimera [16] offer a Virtual Data Catalog as a relational database for provenance information where users register transformations, data objects and derivations. Such records could be incorporated into datasets in CSMD, but to use them profitably would require the access by the appropriate RDBMS to provide a query interface. This would add further complexity to any attempt to develop a complete provenance management system around the CSMD.

The scientific process described above was undertaken as publicly funded research for which the main security concerns are to embargo release of data until after the scientists undertaking the experiments have published their results and then to make them as publicly 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 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 [18]. 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, 2]. A research object 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, traceable. 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.

Finally, we should point out that the current work only captures several fairly limited aspects of software and software executions. At this stage, our aim is to understand its relevance to data provenance. It is not our aim to realise the so-called "one-click" execution dimension of scientific process management. We feel that this is just the beginning to unveil the challenges of dealing with software and executions (e.g. hardware, OS, environment variables, support libraries) in the process, which embrace issues such as handling the relationship between a software execution and a software version, deciding what aspects of a software and executions are needed to be captured, and how to capture them.

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

Table 1: Stakeholder Engagement

7 Final Remarks

The work described in this report has been presented to and discussed with various stakeholders of the I2S2 project between May 2010- March 2011. 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 [15, 22]). 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.

Appendices

Appendix A: XML Schema for Data Ingestion

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"
                      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                      >
  <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" />
```

```
42      </xsd:sequence>
43    </xsd:complexType>
44    <xsd:complexType name="investigationType">
45      <xsd:sequence>
46        <xsd:element maxOccurs="unbounded" name="datasetref"
47                               type="datasetrefType" />
48        <xsd:element name="processref" type="processrefType" />
49        <xsd:element maxOccurs="unbounded" name="parameterref"
50                               type="parameterrefType" />
51      </xsd:sequence>
52      <xsd:attribute name="id" type="xsd:string" />
53      <xsd:attribute name="type" type="xsd:string" />
54    </xsd:complexType>
55    <xsd:complexType name="parameterrefType">
56      <xsd:attribute name="idref" type="xsd:string" />
57    </xsd:complexType>
58    <xsd:complexType name="processrefType">
59      <xsd:attribute name="idref" type="xsd:string" />
60    </xsd:complexType>
61    <xsd:complexType name="datasetrefType">
62      <xsd:attribute name="idref" type="xsd:string" />
63      <xsd:attribute name="type" type="xsd:string" />
64    </xsd:complexType>
65    <xsd:complexType name="datasetsType">
66      <xsd:sequence>
67        <xsd:element maxOccurs="unbounded" name="dataset"
68                               type="datasetType" />
69      </xsd:sequence>
70    </xsd:complexType>
71    <xsd:complexType name="datasetType">
72      <xsd:sequence>
73        <xsd:element maxOccurs="unbounded" name="datafileref"
74                               type="datafilerefType" />
75      </xsd:sequence>
76      <xsd:attribute name="id" type="xsd:string" />
77    </xsd:complexType>
78    <xsd:complexType name="datafilerefType">
79      <xsd:attribute name="idref" type="xsd:string" />
80    </xsd:complexType>
81    <xsd:complexType name="datafilesType">
82      <xsd:sequence>
83        <xsd:element maxOccurs="unbounded" name="datafile"
84                               type="datafileType" />
85      </xsd:sequence>
86    </xsd:complexType>
87    <xsd:complexType name="datafileType">
88      <xsd:sequence>
```

```
89        <xsd:element name="name" type="xsd:string" />
90        <xsd:element minOccurs="0" name="directory" type="xsd:string" />
91        <xsd:element minOccurs="0" name="description" type="xsd:string" />
92      </xsd:sequence>
93      <xsd:attribute name="id" type="xsd:string" />
94    </xsd:complexType>
95    <xsd:complexType name="parametersType">
96      <xsd:sequence>
97        <xsd:element maxOccurs="unbounded" name="parameter"
98                                 type="parameterType" />
99      </xsd:sequence>
100   </xsd:complexType>
101   <xsd:complexType name="parameterType">
102     <xsd:sequence>
103       <xsd:element name="name" type="xsd:string" />
104       <xsd:element minOccurs="0" name="directory" type="xsd:string" />
105       <xsd:element name="parameterfile" type="xsd:string" />
106     </xsd:sequence>
107     <xsd:attribute name="id" type="xsd:string" />
108   </xsd:complexType>
109   <xsd:complexType name="processesType">
110     <xsd:sequence>
111       <xsd:element maxOccurs="unbounded" name="process"
112                                 type="processType" />
113     </xsd:sequence>
114   </xsd:complexType>
115   <xsd:complexType name="processType">
116     <xsd:sequence>
117       <xsd:element name="name" type="xsd:string" />
118       <xsd:element minOccurs="0" name="directory" type="xsd:string" />
119     </xsd:sequence>
120     <xsd:attribute name="id" type="xsd:string" />
121     <xsd:attribute name="type" type="xsd:string" />
122   </xsd:complexType>
123 </xsd:schema>
```

Appendix B: 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 6.

```
<?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>
```

Appendix C: 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)
);

```

Appendix D: ICAT-Personal sourceforge

The code for ICAT-Personal has been made available as open-source software at the following locations.

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`

An Overview of the SVN Code Structure

In the ICATlite svn, under a directory called Code, you should find all the code related to ICAT-personal (or icatlite). In the following, we shall give a brief overview of the directories under the main trunk directory. Most of these directories correspond to a Netbeans project (i.e. Desktop,).

ICAT_LITE_CORE: this is the ingestion part of icatlite, which uses an XML file to guide the ingestion of the data files into the icatlite database. This is a core part of the system.

Major source code subdirectories of ICAT_LITE_CORE:

Entities: these are (database) entity beans/classes generated by Netbeans. Every time the database structure is modified, developers need to generate them again to get them in syn with the db.

Icat.lite.graphviz: DotGeneration.java: is a Java utility for parsing the XML ingestion file to generate graphviz dot file and images. It can be used independently from the rest of the system or be invoked by a backend J2EE application through a WebService calls.

icat.lite.jaxb: this is where the XML ingestion schema (icat_lite_jaxb.xsd) locates.

Icat.lite.jaxb.utils: contains two Java programs: JaxbUtil.java and XMLExtract.java. JaxbUtil.java is the jaxb utility class to parse the xml ingestion file. XMLExtract.java uses the XML file to guide the ingestion of the information extracted from the XML and the actual datafiles into the database.

Icat.lite.saxon: Saxon is abandoned from the use in the system because it relies on some commercial components which are not available in the free Saxon parser.

Utils: some database utilities for accessing the database. GetMyStudy.java is a java program for getting studies out of the database.

ICAT_LITE_WS: a partial implementation of a Web Services Interface to icatlite.

Icatlite-web: this is the Web interface for presenting the information from the database. It is a bunch of PHP files with some css, javascript (not really much at all).

Appendix E:Installation Guide to ICAT-Personal

ICAT-personal consists of two functionalities: data ingestion and data presentation. This instruction assumes that both functionalities are installed on the same machine. Prerequisite software for these functionalities are as follows.

ICAT-p Ingestion (ingesting data into the database):

- Mysql server
- Hibernate
- Java (developed on version 1.6/6)
- Mysql jdbc driver

ICAT-p Web (displaying the ingested data in a web browser, tested with firefox):

- Mysql server
- icatlite database script
- Php (`http://php.net/`)
- Apache
- Graphviz (`http://www.graphviz.org/`) AND Image_graphviz (a php module)

Instructions for installing graphviz and image_graphviz: `http://drupal.org/project/graphviz_filter` (official instruction, but ubuntu doesnt seem to have Image_Graphviz package available 08 Oct. 2010) and `http://drupal.org/node/883114`

Important settings

- Mysql database name: icatlite
- Dump the data from Mysql database: mysqldump u username p icatlite ¿ icatlite.dump (The file icatlite.dump will be generated in the directory where you run the command)
- Import the data back to mysql database: mysql u username p icatlite ¡ icatlite.dump (Assume the file icatlite.dump exists in the directory where you run the command)
- If there is a problem: permission denied php cannot write to new file. (Solution: we need to give apache, i.e. www-data, the permission to all the directories and files) see `http://www.karakas-online.de/forum/viewtopic.php?t=2235`

If there are permissions problem, change both user and group to www-data (the defaults for apache server) in the web server:

```
chown www-data directory/
chgrp www-data directory/
```

Step-by-step instructions

- Download the trunk from the svn

- Under a directory called ICAT_LITE_CORE/database, you should find a script: icatlite-mysql.sql

Set up ICAT-personal demo website.

- Unzip icatlite-web.zip (this is available from the ESC svn under Code/trunk) to an Apache accessible directory
- Under the directory, you can find the following php files: clickable_svg.php html_head.php sections_head.php sidebar.php db_fns.php html_menu.php sections.php sidebar_tail.php demo_standalone.php html_tail.php sections_tail.php study_svg_backup.php demo_website.php index.php sidebar_head.php study_svg.php (AND a directory named /web, which holds css, javascript, and other web page related files.)
- db_fns.php: change the mysql database username password in this php
- modify apache httpd.conf so that it can the apache server know how to interpret svg type

To modify the apache to interpret the SVG type, follow the instructions at `http://kaioa.com/node/45`. The following are added to the /etc/apache2/apache2.conf:

```
  <IfModule mime_module>
      AddType image/svg+xml .svg
      AddType image/svg+xml .svgz
      AddEncoding gzip .svgz
      <FilesMatch \.svgz$>
            <IfModule mod_gzip.c>
                  mod_gzip_on No
            </IfModule>
      </FilesMatch>
</IfModule>
```

References

[1] S. Bechhofer, D. De Roure, M. Gamble, C. Goble, and I. Buchan. *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] Sean Bechhofer, John Ainsworth, Jiten Bhagat, Iain Buchan, Philip Couch, Don Cruickshank, David De Roure, Mark Delderfield, Ian Dunlop, Matthew Gamble, Carole Goble, Danius Michaelides, Paolo Missier, Stuart Owen, David Newman, Shoaib Sufi, *Why Linked Data is Not Enough for Scientists*, pp. 300-307, 6th IEEE International Conference on e-Science, 2010

[3] Mark Borkum, Carl Lagoze, Jeremy Frey, and Simon Coles. *A Semantic eScience Platform for Chemistry*, pp.316-323, 6th IEEE International Conference on e-Science, 2010

[4] 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.

[5] 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*, pp.201-207, 5th IEEE International Conference on e-Science, 2009.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] Brian Matthews, Arif Shaon, Juan Bicarregui, Catherine Jones, Jim Woodcock, Esther Conway, *Towards a Methodology for Software Preservation*. 6th International Conference on Preservation of Digital Objects (iPres 2009), San Francisco, USA, 05-06 Oct 2009.

[12] Brian Matthews, Arif Shaon, Juan Bicarregui, Catherine Jones, *A Framework for Software Preservation*, International Journal of Digital Curation, 5 (1), 2010.

[13] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. Van den Bussche. *The Open Provenance Model core specification (v1.1)*. Future Generation Computer Systems, in Press, 2010.

[14] 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.

[15] Manjula Patel *Requirements Report*, I2S2 project report D1.1, July 2010. `http://www.ukoln.ac.uk/projects/I2S2/documents/I2S2-WP1-D1.1-RR-Final-100707.pdf`.

[16] Eric F. Pettersen, Thomas D. Goddard, Conrad C. Huang, Gregory S. Couch, Daniel M. Greenblatt, Elaine C. Meng, Thomas E. Ferrin, *UCSF Chimera - A visualization system for exploratory research and analysis*, Journal of Computational Chemistry, 25(13), 1605 - 1612, 2004.

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

[18] 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.

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

[20] 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.

[21] M.G. Tucker, D.A. Keen, M.T. Dove, A.L. 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/`.

[22] 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`.

[23] Erica Yang, Brian Matthews, Michael Wilson, *Enhancing the Core Scientific Metadata Model to Incorporate Derived Data*, pp.145-152, 6th IEEE International Conference on e-Science, 2010

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