PaN-data ODI

Deliverable D5.3

Report on the implementation of the three virtual laboratories

Grant Agreement Number	RI-283556
Project Title	PaN-data Open Data Infrastructure
Title of Deliverable	Report on the implementation of the three virtual laboratories
Deliverable Number	D5.3
Lead Beneficiary	STFC
Deliverable Dissemination Level	Public
Deliverable Nature	Report
Contractual Delivery Date	31 Aug. 2014
Actual Delivery Date	XX Oct. 2014

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

Abstract

This report demonstrates and evaluates the services being developed within the PaN-data ODI project. A survey of the implementation status of these services at all participating institutes is included. The metadata catalogue plays a central role in the PaN-data project. Therefore it is discussed how scientists benefit from this service. The status of the tomography database is reviewed. Details of the NeXus deployment at various partners are presented. Eventually, three data processing frameworks, which are developed within the PaN-data community, are compared by implementing two scientific use cases.

Keyword list

PaN-data ODI, virtual laboratories, metadata, data catalogue, standard data format

Document approval

Approved for submission to EC by all partners on XX.10.2014

Revision history

Issue	Author(s)	Date	Description
1.0	Thorsten Kracht	Aug 2014	First version
1.1	Jan Kotanski	Aug 2014	NeXus deployment at DESY
1.2	Gero Flucke	Aug 2014	Comparison of data processing frameworks
1.3	Eugen Wintersberger	Aug 2014	Status of the NeXus library
1.4	Mark Koennecke	Jul 2014	NeXus deployment at PSI-SINQ
1.5	Jens-Uwe Hoffmann	Aug 2014	NeXus deployment at HZB
1.6	Guifre Cuni	Aug 2014	NeXus deployment at ALBA
1.7	Frank Schluenzen	Aug 2014	Science3D section
1.8	Tobias Richter	Sep 2014	NeXus deployment at DLS
1.9	Jean-Francois Perrin	Sep 2014	NeXus deployment at ILL
1.10	Tom Griffin	Sep 2014	NeXus deployment at ISIS

Table of contents

Page

1 INTRODUCTION ... 6

1.1 THE AMENDMENT OF THE DESCRIPTION OF WORK ... 6

1.2 OVERVIEW OF THE DELIVERABLE ... 7

2 SURVEY OF THE IMPLEMENTATION OF THE PAN-DATA ODI SERVICES ... 9

2.1 SCIENTIFIC DATA POLICY ... 9

2.2 COMMON IDENTITY MANAGEMENT SYSTEM UMBRELLA ... 10

2.3 COMMON DATA FORMAT NEXUS/HDF5 ... 10

2.4 METADATA CATALOGUE ICAT-4 ... 12

3 DEMONSTRATION AND EVALUATION OF THE PAN-DATA SERVICES ... 15

3.1 INTRODUCTION ... 15

3.2 VL1: POWDER DIFFRACTION ... 15

3.3 VL2: SMALL ANGLE SCATTERING ... 18

3.4 ISIS ICAT ... 19

3.5 SUMMARY ... 24

4 SCIENCE 3D – THE TOMOGRAPHY DATABASE ... 25

4.1 THE SCIENCE 3D KICK-OFF WORKSHOP ... 26

4.2 CONCEPT OF THE SCIENCE 3D OPEN ACCESS CATALOGUE ... 26

4.3 CURRENT STATUS OF THE SCIENCE 3D OPEN ACCESS CATALOGUE ... 29

4.3.1 Schema ... 30

4.3.2 Catalog ... 30

4.3.3 Documentation & Dissemination ... 31

4.3.4 Summary ... 31

5 NEXUS DEPLOYMENT ... 33

5.1 INTRODUCTION ... 33

5.2 STATUS OF THE NEXUS LIBRARY ... 33

5.2.1 Why implementing a new Nexus library? ... 33

5.2.2 Project status ... 34

5.2.3 Performance markers for libpnicore ... 35

5.2.4 Performance markers for libpniio ... 36

5.3 EXTERNAL FILTERS ... 37

5.3.1 First implementation of an external filters infrastructure ... 38

5.3.2 External filters and user applications ... 39

5.3.3 External filters with Python ... 40

5.4 NEXUS INTEGRATION AT DESY ... 41

5.4.1 Overview ... 41

5.4.2 The configuration procedure ... 42

5.4.3 Component selection 45
5.4.4 Mandatory metadata 47
5.4.5 The data acquisition process 48
5.4.6 Generalizing the DESY NeXus implementation 49
5.5 NEXUS INTEGRATION AT ALBA 49
5.5.1 Overview 49
5.5.2 Application definitions 51
5.5.3 Data and metadata sources selection 51
5.6 NEXUS INTEGRATION AT DLS 53
5.6.1 Overview 53
5.6.2 Device selection 53
5.6.3 Configuration 54
5.6.4 Default metadata 54
5.6.6 Application definitions 55
5.7 NEXUS INTEGRATION AT HZB 55
5.7.1 Overview 55
5.7.2 Mandatory metadata 56
5.7.3 Generalizing the HZB NeXus implementation 57
5.8 NEXUS INTEGRATION AT ILL 57
5.8.1 Configuration 57
5.8.2 Device selection 58
5.8.3 Mandatory metadata 58
5.9 NEXUS INTEGRATION AT ISIS 58
5.9.1 Overview 58
5.9.2 NeXus structure 58
5.9.3 Mandatory metadata 59
5.10 NEXUS INTEGRATION AT PSI-SINQ 60
5.10.1 SINQ Overview 60
5.10.2 5.10.2 Writing NeXus files at SINQ 61
5.10.3 SINQ NeXus standard compliance 62
5.10.4 SINQ auxiliary NeXus handling software 62
5.11 SUMMARY 62
6 COMPARISON OF DATA PROCESSING FRAMEWORKS 63
6.1 INTRODUCTION 63
6.2 DESCRIPTION OF INVESTIGATED TOOLS 64
6.2.1 DAWN 64
6.2.2 DPDAK 68
6.2.3 Mantid 69
6.3 MONITORING DIFFRACTION MEASUREMENTS 73
Description of the science case 73
6.3.1 Diffraction monitoring using DAWN 75

6.3.2 Diffraction monitoring using DPDAK ... *78*
6.3.3 Diffraction monitoring using Mantid ... *80*
6.4 VISUALISATION AND ANALYSIS OF GRAZING INCIDENCE AND SMALL ANGLE SCATTERING DATA ... 85
Description of the science case ... *85*
6.4.1 Small angle scattering analysis with DAWN ... *86*
6.4.2 Small angle scattering analysis with DPDAK ... *97*
6.4.3 Small angle scattering analysis with Mantid: ... *101*
6.5 COMPARING REMARKS ... 104
7 CONCLUSIONS ... 107

1 Introduction

The PaN-data consortium has been founded by European X-ray and Neutron institutes to collaboratively install a common e-infrastructure harmonizing various data management tasks. The relevant services and joint research activities are organized within the PaN-data ODI project. They comprise a common authentication system, a standardized metadata catalogue, a standardized data format, a concept to describe the data provenance, an analysis software database, a data preservation concept and investigations on parallel file systems. Addressing these issues in a collaborative approach saves resources at the participating research infrastructures by avoiding parallel developments. Scientists profit from this project in several ways: analysis programs can uniformly access data from various institutes because the data are organized in a standard format, file discovery can be done across facility borders thanks to the standardized metadata catalogue, authentication to digital user offices and metadata catalogues is simplified by the common identity management system Umbrella and data provenance helps scientists to keep track of the data processing steps.

Workpackage 5 has to setup three virtual laboratories to demonstrate the availability of the services and to evaluate them. This is a list of the virtual laboratories and the related tasks:

- VL1: Structural ‘joint refinement’ against x-ray and neutron powder diffraction data
    - Raw data searched for by an authenticated user through the data catalogues.
    - Access is authorized and data downloaded from facility archives.
    - Relevant analysis software searched for in software database.
    - Software downloaded and run locally or at facility.
    - Analysis carried out.
    - Results (refined structure) and any relevant reduced data uploaded to facility archives.
- VL2: Simultaneous analysis of SAXS and SANS data for large scale structures
    - Raw data searched for by an authenticated user through the data catalogues.
    - Access is authorized and data downloaded from facility archives.
    - Relevant analysis software searched for in software database.
    - Software downloaded and run locally or at facility.
    - Analysis carried out.
    - Results (modeled structure) and any relevant reduced data uploaded to facility archives.
- VL3: Access to tomography data exemplified through paleontological samples
    - Setup a public access database for storing tomographic raw and processed data of paleontological data e.g. 2D tomographs and 3D processed images.
    - Provide authorized access from multiple institutes to store processed data in the database.
    - Enable public access to data in database.
    - Implement long term archiving of database.

1.1 The amendment of the description of work

In the course of the PaNdata project the work being executed within WP5 deviated considerably from the original plan. Therefore the reviewers recommended an amendment of the description of work. The following issues have to be considered:

- VL1 (powder diffraction) and VL2 (small angle scattering) cannot be sensibly deployed as a "set of integrated end-to-end user and data services" mainly because the ICAT (metadata catalogue) adaptation to the local IT infrastructures proved to be very complicated. So far only ISIS and DLS have operational catalogues providing download functionality for authorized users. Most of the other PaNdata partners have ICAT test instances installed with open access data only. Therefore the original plan for VL1 and VL2 was replaced by two tasks:
  - A survey has been carried out presenting the availability of the PaNdata services at all participating institutes. It has been presented during the last annual review. This deliverable contains an updated version of the survey.
  - A section has been included that demonstrates and evaluates PaN-data services with the involvement of scientists working on diffraction (VL1) and small angle scattering experiments (VL2).
- The NeXus deployment at the experimental stations with frequently re-arranged setups turned out to be a complex task. Hence, WP5 was extended to design and implement procedures to solve this problem. Furthermore, the NeXus integration at other participating institutes is presented.
- The original objective of VL3 was to setup a tomography database only storing paleontological samples. It was decided that this database should include also other samples making it suitable for educational purposes. This way the VL3 audience is widened.
- DAWN, Mantid and DPDAK are integrated, modular data processing frameworks developed by PaN-data institutes. In order to evaluate the synergy potential in the area of visualization, scripting and workflows, these tools have been compared using test cases of VL1 and VL2.

To summarize: the proposed modifications of the WP5 description of work result from the experiences made during the project. It turned out that part of the objectives were too ambitious to be implemented. On the other hand WP5 incorporates now additional tasks, like the survey, the description of the NeXus implementation at various PaN-data institutes, the comparison of data processing frameworks and the extension of VL3. The role of VL1 (powder diffraction) and VL2 (small angle scattering) has been redefined. Still, the VLs provide the use cases for various investigations carried out for this report. The main purpose of WP5 to act as a demonstrator for the PaNdata services and their integration remains to be the core-objective of this work package.

1.2 Overview of the deliverable

The first deliverable of WP5 is a list of specific requirements for the other work packages. It was released in April 2012. The second deliverable, released in June 2013, describes the status of the implementation and deployment of the elements from the other work packages enabling the virtual laboratories.

This deliverable starts with a survey of the implementation of the PaN-data services at the participating institutes. It is an updated version of what has been presented during the last annual review in Brussels on 28 November 2013. The next section is the central part of this document. It exemplifies the utilization of the PaN-data ODI services with diffraction (VL1) and small angle scattering (VL2) data. A user being authenticated by Umbrella searches files in the metadata catalogue of a remote facility. The local authorization system grants read-access and the data are downloaded. The mandatory metadata are discussed and tools for the inspection of the downloaded files are presented. Section 4 reports on the work on VL3, the tomography database and a related portal

(Science 3D). The portal allows for an upload of raw and processed imaging data of biological samples. Each dataset is accompanied by a description making it generally understandable. Scientists benefit from this platform because they can present their results to a general public using 3D visualization tools. In addition, Science 3D is well suited for educational purposes. The next section gives a detailed description of the NeXus deployment at the Petra III beamlines because a big effort was put in the development of a framework capable of producing NeXus files in an environment with frequently changing experimental setups. An overview of how other PaN-data partners address this issue has been included. The following section presents a comparison of the data processing frameworks DAWN, Mantid and DPDAK using test cases from diffraction (VL1) and small angle scattering (VL2) experiments. The first example is an online monitoring application; the second example is a simple data processing algorithm. Both use cases are implemented in the three frameworks allowing for a detailed comparison.

2 Survey of the Implementation of the PaN-data ODI Services

The PaN-data ODI project creates an open data infrastructure across the participating institutes. The essential parts of this initiative are a generally accepted scientific data policy, a common authentication system, a standard data format and a common metadata catalogue. This section presents the status of the implementation of these components at the various institutes.

2.1 Scientific data policy

Whenever scientists visit a facility to conduct a measurement a couple of questions regarding data treatment arise: who is responsible for data storage, how long are the data preserved, which is the data format, do data become publically accessible after an embargo period, how long is the embargo period and can it be extended. These questions become more important, if investigations are carried out at various institutes. Scientists would clearly benefit from institutes following a common approach in this respect. Therefore, PaN-data Europe proposed a collection of rules, laid down in the deliverable D2.1, stating that the facility should act as a custodian for the data, that NeXus/HDF5 is the standard data format, that the data become publically accessible after an embargo period of three years and that the embargo period can be extended. Table 1 presents a survey regarding the implementation of a data policy at the participating institutes.

	Data Policy	PaN-Data Type
ALBA	In Progress	Yes
CEA/LBB	Yes	
DESY	Yes	No
DLS	Draft	Yes
ELETTRA	Yes	Yes
ESRF	No	
HZB	Draft	In Discussion
ILL	Yes	Yes
ISIS	Yes	Yes
JCNS	No	
MAX IV	Draft	Yes
PSI	Draft	Yes
SOLEIL	Draft	

Table 1: Status of the implementation of the PaN-data scientific data policy

Most of the PaN-data partners have already adopted a data policy or are preparing a draft. The PaN-data type is favoured by the majority. Here are the details:

- ALBA: so far there are some unpublished guidelines which are in accordance with the PaN-data policy.
- DESY: the principle investigator has the full responsibility of the data. As a service for the PI, DESY offers to store the data over a complete data live cycle at the expense of the PIs institute. The data live cycle is expected to be 10 years.
- DSL: a draft policy is defined in lines with PaN-data.
- HZB: a data policy is under evaluation.

- SOLEIL: a draft has been prepared

2.2 Common identity management system Umbrella

The inflation of credentials is a far spread annoyance for everyone using IT services in our community. This statement is particularly true for scientists who conduct measurements at several institutes. Many username/password combinations have to be memorized to gain access to digital user offices, portals, file systems and compute nodes. Keeping track of the credentials is a boring task which can be error prone and which may also involve security risks. One way to overcome this problem is the common identity management and authentication system Umbrella.

Umbrella creates EU-wide unique user IDs using a system of distributed identity providers (IdP). The account information is synchronized by master-master-replication to keep all IdPs up-to-date. Umbrella IDs are persistent in the sense that they remain valid even if a user changes the affiliation. In order to guarantee a sustainable joint operation of the identity management system, a Memorandum of Understanding has been developed between the partners.

	Operational	Test Instance	Planned
ALBA	Yes		
CEA/LBB	No	No	No
DESY	Yes		
DLS	Yes		
ELETTRA	Yes		
ESRF	Yes		
HZB	No	In progress	Yes
ILL	Yes		
ISIS	Yes		
JCNS	No	No	No
MAX IV	No	No	Yes
PSI	Yes		
SOLEIL	Soon		

Table 2: Status of Digital User Office integration of Umbrella

Scientists would appreciate a solution giving them single-sign-on access to digital user offices, data, metadata catalogues, compute resources and experiment control applications. In principle, Umbrella is able to provide this service. However, this requires a big implementation effort. Therefore work started with the Umbrella integration of the digital user offices. Table 2 shows that most of the PaN-data partners have an Umbrella integration of the Digital User Offices.

2.3 Common data format NeXus/HDF5

Raw data have to be accompanied by metadata describing the precise experimental procedure applied including details about the source, the optical components of the beamline, the detectors, the sample environment, etc. Administrative information like the name of the facility, the name of the beamline, an identifier for the beamtime and the names of the investigators has to be captured as well. Calibrations, background subtractions, projections, etc. are another source of metadata because these data processing steps have to be documented in order to generate reproducible results.

The PaN-data community selected NeXus[1] as the common data format because it is capable of managing the aforementioned raw/derived data and metadata. Moreover, the mere fact that the European Photon and Neutron science community agreed on a common format promotes the development of analysis programs and visualization tools because code developers have to program against one file format only. Another aspect is that data and applications can be interchanged between the facilities.

NeXus defines the structure of the data but not the physical file format. By choosing HDF5 PaN-data followed a recommendation of the EC about binary data storage. NeXus can be understood as a configuration scheme for HFD5 meaning that NeXus files can be interpreted by all programs that have a HDF5 interface. IDL, Matlab, Mathematica are examples for general purpose tools that are in this sense NeXus-aware.

Within the PNI-HDRI[2] project a new NeXus C++ library based on HDF5 has been developed creating a strictly object oriented application programmers interface including Python bindings. This work was coordinated with the NeXus International Advisory Committee. The library is described in section 5.2.

Currently there are important new developments in the HDF5 sector: parallel I/O, the integration of external data compression algorithms, the single-writer-multiple-reader feature and the proposal to implement virtual datasets. These improvements are beneficial PaN-data partners.

NeXus was originally proposed in 1994 by members of the photon, neutron and muon science community. As far as the utilization is concerned the neutron and muon facilities initially took the lead. Table 3 shows the current status of the NeXus deployment at the PaN-data institutes.

	In Production	Full Desc.	Mand. MD	Respect ADs	Autom. Ingest
ALBA	Yes	Yes	Yes	Yes	No
CEA/LBB	Yes	Yes	No	No	No
DESY	Yes,In progress	Yes	Yes	No	Yes
DLS	Yes	Yes	Yes	Yes/In progress	Yes
ELETTRA	No, HDF5				
ESRF	No				
HZB	Yes	No	No	Yes	No
ILL	Yes	No	Yes	Yes	Yes
ISIS	Yes	Yes	Yes	Yes	Yes
JCNS	In progress			No	
MAX IV	No				
PSI-SINQ	Yes	Yes	No	Yes	Yes
PSI-SLS	In progress	In progress	No	In progress	In progress
SOLEIL	Yes	Yes		No	No

[1] Nexusformat.org

[2] www.pni-hdri.de

Table 3: The status of the NeXus utilization at the PaN-data partners

Replacing a legacy format by NeXus is a big effort. One has to overcome the inertia caused by the habits of the scientists involved. In addition there is a technical challenge. The NeXus integration makes sense only, if the files contain the complete description of the experiment. This requires a lot of effort, especially for experimental stations which change their setup frequently. The way how this is done at various PaN-data institutes is described in section 5.

NeXus files can respect application definitions (ADs). They are discipline specific schemes defining the precise semantics for a subset of the metadata. They also impose standardizations on the co-ordinates representing the position and orientation of the sample and the detectors. ADs establish in interface for files and analysis program belonging to the same experimental technique, i.e. programs that comply with an AD can process all files that implement this AD no matter where the files have been created.

Table 3 gives an overview of the NeXus utilization at the PaN-data institutes. Here is a description of the columns:

- *In Production*: 'Yes', if at least one experiment at a facility produces NeXus files.
- *Full Desc.*: 'Yes' for those institutes that create NeXus files that fully describe the data.
- *Mand. MD*: 'Yes', if the facility defined a set of metadata that have to be stored in every output file. Section 0 gives examples for the mandatory metadata selection.
- *Respect ADs*: 'Yes', if a facility respects at least one AD.
- *Autom. Ingest*: 'Yes', if the metadata being used for catalogues are extracted from the NeXus files in an automated way. This item has been included in the survey because it reflects the level of integration as far as NeXus and the metadata catalogue are concerned.

We conclude that NeXus/HDF5 is the generally accepted standard data format within the PaN-data community.

2.4 Metadata catalogue ICAT-4

Most PaN-data institutes assumed the responsibility to act as a custodian for all data being produced at their facilities. Therefore comprehensive data management services had to be installed for data storage and archival, data export, data curation and preservation. A metadata catalogue manages the information that is necessary to execute these tasks.

The term ingestion denotes the procedure of inserting records into the metadata catalogue. A record corresponds to a dataset. This can be a single file or a collection of files. File properties like name, size and creation data are part of the metadata. Other information may be extracted from the files or may come from the digital user office database. The amount of the metadata stored varies from facility to facility. However, WP4 proposed a set of mandatory metadata being respected by all participating institutes (D4.3). This way cross-facility file searches are facilitated.

The PaN-data partners agreed to use ICAT-4[3] as the common metadata catalogue. TOPCAT is the related interactive web GUI for file discovery and downloads. It supports file searches across several ICAT-4 instances. Examples of how this interface is used in practice can be found in section 3 of this document.

Table 4 shows the status of the ICAT deployment at the PaN-data institutes.

[3] www.icatproject.org

	In Prod.	Test Inst.	Restr. Data	Authent.	Author.	IDS
ALBA	No	Yes	No	Yes	Yes	No
CEA/LBB	No	No				
DESY	No	Yes	No	No	No	No
DLS	Yes	Yes	Yes	Yes	Yes	Yes
ELETTRA	No	Yes	No	Yes	Yes	No
ESRF	Yes	Yes	No	Yes	Yes	No
HZB	No	Yes	No	Yes	No	No
ILL	Yes	Yes	No	Yes	Yes	No
ISIS	Yes	Yes	Yes	Yes	Yes	Yes
JCNS	No	No				
MAX IV	No	No				
PSI	No	No				
SOLEIL	No	Yes	Yes	Yes	Yes	No

Table 4: The deployment of ICAT at the PaN-data institutes

Here are the descriptions of the columns:

- *In Prod.*: ‘Yes’, if a facility has an ICAT-4 instance in production or if it is in a production ready state.
- Test Inst.: ‘Yes’, if a facility runs an ICAT-4 test instance.
- *Restr. Data*: ‘Yes’, if the metadata catalogue contains data with access restrictions.
- *Authent.*: ‘Yes’, if authentication is implemented in the metadata catalogue.
- *Author.*: ‘Yes’, if authorization is implemented in the metadata catalogue.
- *IDS*: ‘Yes’, if the ICAT-4 instance has an integrated data server, allowing file downloads.

Two institutes run fully operational ICAT-4 catalogues:

- ISIS: The catalogue has 12M entries covering all files created since 1984. The size of the data is 40 TB. In August 2014 50 distinct users connected to the service. There were 15130 sessions and 17072 search operations. The data have been catalogued together with the description of the sample and metadata from the proposal. Data and metadata will be made publically available after an embargo period of three years. Citable DOIs are issued for all experiments carried out at ISIS. The ICAT service will accept Umbrella credentials before the end of the PaN-data ODI project.
- DLS: ICAT catalogues 1.6 PB of data distributed across 413M files. About 100.000 files are downloaded on average per month. It is planned to issue DOIs. The Umbrella integration will be available as soon as a database has been created mapping local users to Umbrella ID. Data are owned by the users. There is no procedure to make them publically available. DLS catalogues datasets containing all files that where generated in the same scan.

The list of metadata managed by the DLS and ISIS ICAT instances can be found in section 3.

Two facilities almost reached the ICAT-4 production state:

- ESRF: A catalogue has been installed for ID16A-NI. It will be populated as soon as new experiments are conducted at this experimental station. There are three categories of metadata:

  - Cat-1: Proposal reference, beamtime ID, sample name, experiment name, start and end time. This information is pulled from the digital user office.
  - Cat-2: File properties: name, location, size and format. Currently the ESRF data format is used (EDF).
  - Cat-3: Experiment parameters, like beam energy, etc. These metadata can be specific for the experiment techniques.

  The first two categories will be common for all beamlines.

- ILL: The production state will be reached soon. This includes the integrated data server and the Umbrella authenticator. The plan is to insert also older data into the catalogue, data that has been produced since the data policy has been released. During a period of 3 years the access to the data and the metadata is restricted to the investigators. The embargo period can be extended to 5 years. Metadata captured:
  - Investigation: name, title summary, proposal ID, investigation type, start and end time
  - File properties: name, location, creation and modification dates, size, format, checksum.
  - Datasets: link to the related files of various types (test, calibration, acquisition, backgroundNoise)
  - Facility: metadata as required by the ICAT data model.
  - Instrument: name, full name, description, type, URL
  - Sample: name and description
  - Study: ILL proposal as defined in the ICAT data model

To conclude: it has already been stated that the ICAT deployment is behind the original expectations because the adaptation of ICAT to the local IT infrastructures proved to be a very complex task. Therefore only two PaN-data institutes have fully operational ICAT services by now. Still big progress has been made during the course of this project. The ICAT service verifications are proof of this. The majority of the PaN-data institutes intend to use ICAT as their metadata catalogue.

3 Demonstration and Evaluation of the PaN-data Services

3.1 Introduction

WP5, virtual laboratories, has to demonstrate how the PaN-data developments promote the research carried out at European Photon and Neutron sources. This section elaborates on the practical implications of the new services using concrete examples. It has already been pointed out that the current state of the ICAT deployment, which is the central part of the whole project, is behind the original expectations. Only two of the PaN-data institutes, ISIS and DLS, reached a fully operational ICAT. Therefore it was decided to restrict the demonstration to these institutes.

Whenever measurements have been performed at a remote facility, certain steps have to be executed to retrieve the results: files belonging to an investigation have to be discovered and the relevant data have to be identified and downloaded. Descriptive information has to be inspected. In addition, it is convenient for the researches to have a quick look at the downloaded data to verify that the selection was successful. This procedure is demonstrated using data from two beamtimes at DLS. The experimental techniques employed are powder diffraction and small angle scattering corresponding to VL1 and VL2. The subsections 3.2 and 3.3 illustrate the procedure step-by-step. The scientists involved made an assessment of the services. The results are included.

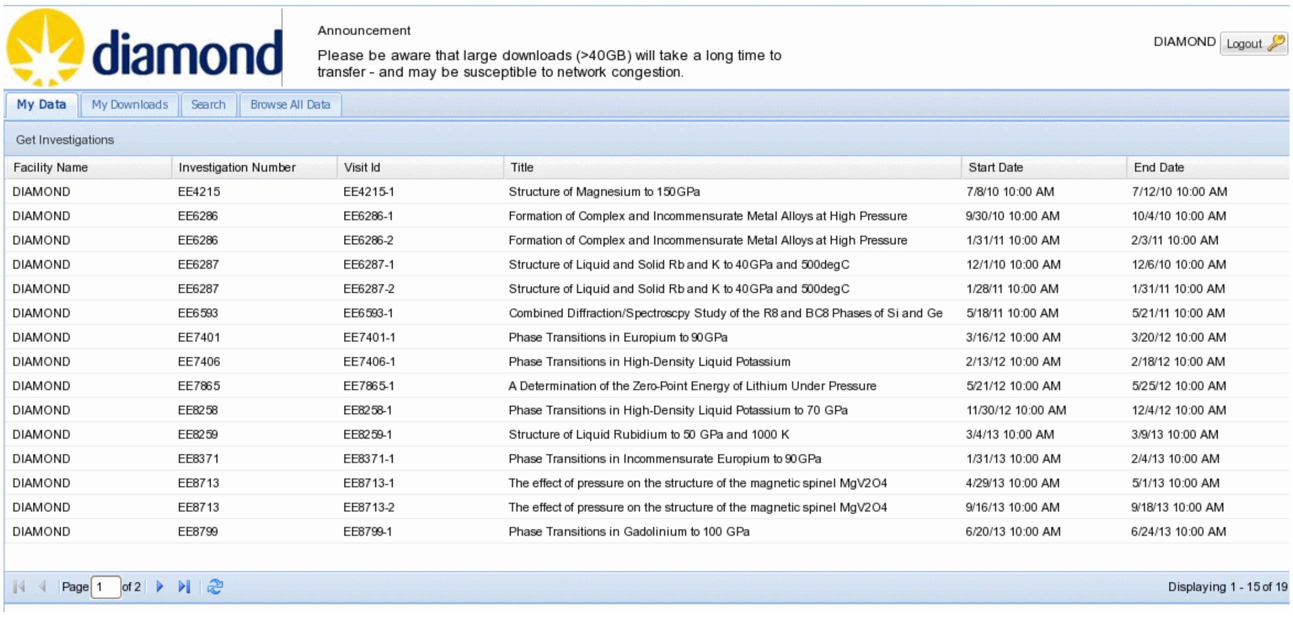
The PaN-data data policy says that all data and metadata become public after a certain time span. Since the ISIS ICAT service implements this requirement, section 3.4 uses this metadata catalogue as an example to elaborate on implications related to the discovery of open access data.

3.2 VL1: Powder diffraction

The following demonstration is based on data collected during a beamtime at DLS early in 2013. The measurement was carried out by a group of 4 researchers from DESY and DLS. The beamtime began on January 17, 2013 and lasted 5 days. The total number of datasets created is 347.

DLS offers two ICAT interfaces for general users: TOPCAT, the web-based ICAT client, and DAWN, the data analysis workbench. TOPCAT is the proper tool for accessing large amounts of data from remote. The ICAT DAWN interface is well suited for on-site (DLS) file access or if smaller amounts of data are processed. The following paragraphs are limited to the work with TOPCAT.

DLS issues persistent user ids, the fedid, for visitors conducting measurements at the facility. All kinds of IT and the access to the metadata catalogue services are controlled by the fedid. The service is available under icat.diamond.ac.uk. After a successful login TOPCAT displays all datasets for which the user has read permission (Figure 1)



Announcement

Please be aware that large downloads (>40GB) will take a long time to transfer - and may be susceptible to network congestion.

DIAMOND Logout

My Data | My Downloads | Search | Browse All Data

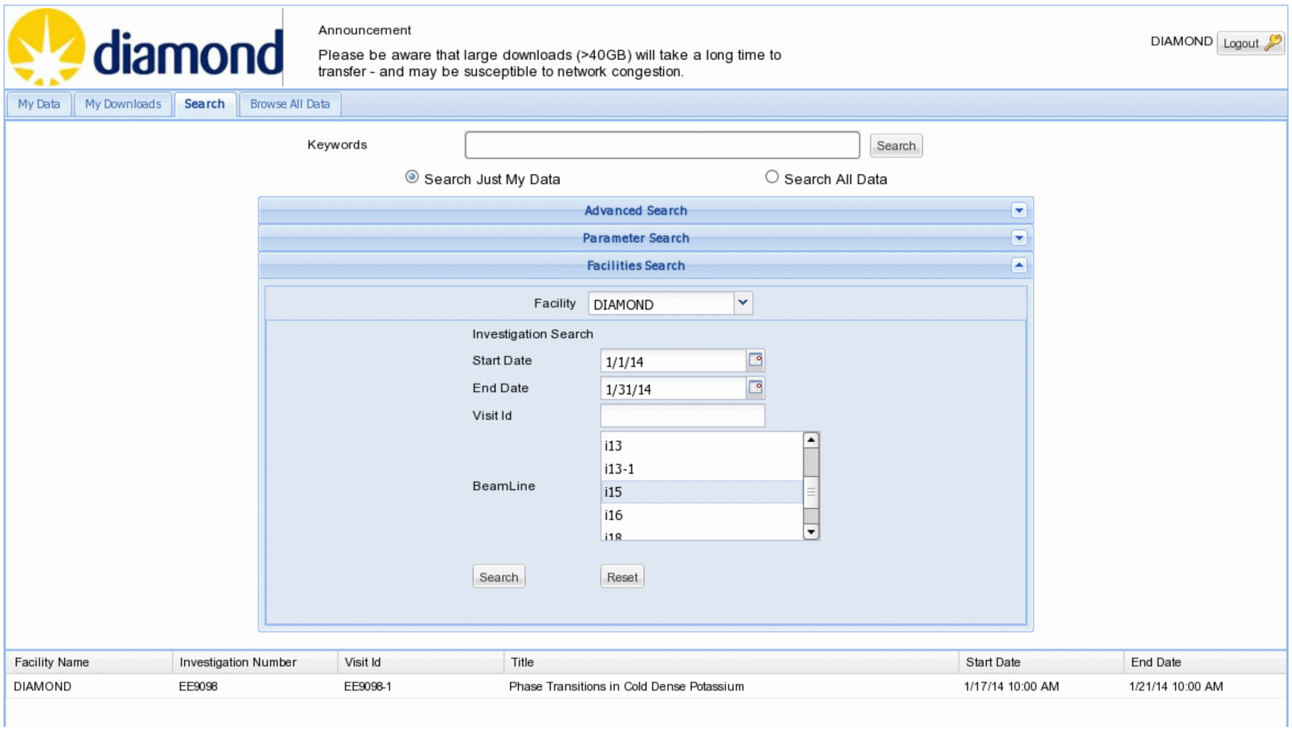
Get Investigations

Facility Name	Investigation Number	Visit Id	Title	Start Date	End Date
DIAMOND	EE4215	EE4215-1	Structure of Magnesium to 150GPa	7/8/10 10:00 AM	7/12/10 10:00 AM
DIAMOND	EE6286	EE6286-1	Formation of Complex and Incommensurate Metal Alloys at High Pressure	9/30/10 10:00 AM	10/4/10 10:00 AM
DIAMOND	EE6286	EE6286-2	Formation of Complex and Incommensurate Metal Alloys at High Pressure	1/31/11 10:00 AM	2/3/11 10:00 AM
DIAMOND	EE6287	EE6287-1	Structure of Liquid and Solid Rb and K to 40GPa and 500degC	12/1/10 10:00 AM	12/6/10 10:00 AM
DIAMOND	EE6287	EE6287-2	Structure of Liquid and Solid Rb and K to 40GPa and 500degC	1/28/11 10:00 AM	1/31/11 10:00 AM
DIAMOND	EE6593	EE6593-1	Combined Diffraction/Spectroscpy Study of the R8 and BC8 Phases of Si and Ge	5/18/11 10:00 AM	5/21/11 10:00 AM
DIAMOND	EE7401	EE7401-1	Phase Transitions in Europium to 90GPa	3/16/12 10:00 AM	3/20/12 10:00 AM
DIAMOND	EE7406	EE7406-1	Phase Transitions in High-Density Liquid Potassium	2/13/12 10:00 AM	2/18/12 10:00 AM
DIAMOND	EE7865	EE7865-1	A Determination of the Zero-Point Energy of Lithium Under Pressure	5/21/12 10:00 AM	5/25/12 10:00 AM
DIAMOND	EE8258	EE8258-1	Phase Transitions in High-Density Liquid Potassium to 70 GPa	11/30/12 10:00 AM	12/4/12 10:00 AM
DIAMOND	EE8259	EE8259-1	Structure of Liquid Rubidium to 50 GPa and 1000 K	3/4/13 10:00 AM	3/9/13 10:00 AM
DIAMOND	EE8371	EE8371-1	Phase Transitions in Incommensurate Europium to 90GPa	1/31/13 10:00 AM	2/4/13 10:00 AM
DIAMOND	EE8713	EE8713-1	The effect of pressure on the structure of the magnetic spinel MgV2O4	4/29/13 10:00 AM	5/1/13 10:00 AM
DIAMOND	EE8713	EE8713-2	The effect of pressure on the structure of the magnetic spinel MgV2O4	9/16/13 10:00 AM	9/18/13 10:00 AM
DIAMOND	EE8799	EE8799-1	Phase Transitions in Gadolinium to 100 GPa	6/20/13 10:00 AM	6/24/13 10:00 AM

Page 1 of 2 Displaying 1 - 15 of 19

Figure 1: DLS ICAT Available Investigations

The data are sorted by the investigation number. An investigation is roughly equivalent to a beam-time. The titles of the investigations and the start and end date are displayed. The total number of investigations of the example user is rather limited, therefore the relevant data could be located without additional tools, just by going through the displayed list. However, since this section is a demonstration, we select the Search tab and supply the start and end date of the visit. The results can be found in the bottom line of Figure 2: “Phase Transitions in Cold Dense Potassium”.



Announcement

Please be aware that large downloads (>40GB) will take a long time to transfer - and may be susceptible to network congestion.

DIAMOND Logout

My Data | My Downloads | Search | Browse All Data

Keywords: Search

(● Search Just My Data) ○ Search All Data

Advanced Search

Parameter Search

Facilities Search

Facility: DIAMOND

Investigation Search

Start Date: 1/1/14

End Date: 1/31/14

Visit Id:

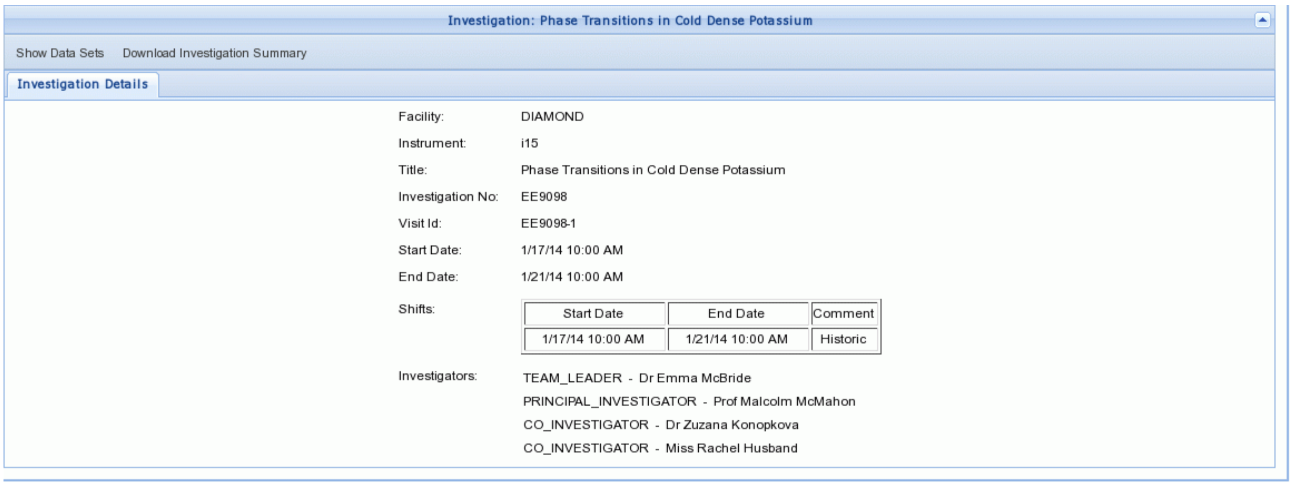
BeamLine: i13, i13-1, [✓ i15], i16, i18

Search Reset

Facility Name	Investigation Number	Visit Id	Title	Start Date	End Date
DIAMOND	EE9098	EE9098-1	Phase Transitions in Cold Dense Potassium	1/17/14 10:00 AM	1/21/14 10:00 AM

Figure 2: DLS ICAT search window

A right-click on the investigation number offers the options “show investigation” and “show data sets”. In addition to the information mentioned so far, the investigation summary (Figure 3) contains the name of the instrument (i15) and the names of the participants.



Investigation: Phase Transitions in Cold Dense Potassium

Show Data Sets | Download Investigation Summary

Investigation Details

Facility:	DIAMOND
Instrument:	i15
Title:	Phase Transitions in Cold Dense Potassium
Investigation No:	EE9098
Visit Id:	EE9098-1
Start Date:	1/17/14 10:00 AM
End Date:	1/21/14 10:00 AM

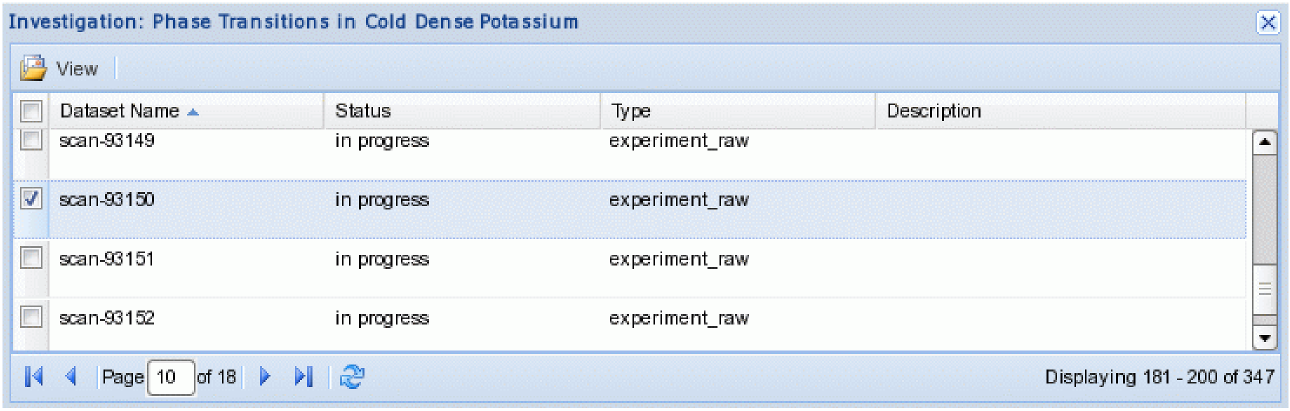
Shifts:

Start Date	End Date	Comment
1/17/14 10:00 AM	1/21/14 10:00 AM	Historic

Investigators:
TEAM_LEADER - Dr Emma McBride
PRINCIPAL_INVESTIGATOR - Prof Malcolm McMahon
CO_INVESTIGATOR - Dr Zuzana Konopkova
CO_INVESTIGATOR - Miss Rachel Husband

Figure 3: DLS ICAT investigation summary

The data sets window, Figure 4, reveals that an investigation is a collection of scans. In our example the list has 347 entries. The user selects the dataset of interest on the basis of the scan number. This quantity has to be extracted from the log book of the experiment. Each data set consists of several files. They are displayed by selecting one or more data sets and clicking the "View" button in the data set window.



Investigation: Phase Transitions in Cold Dense Potassium

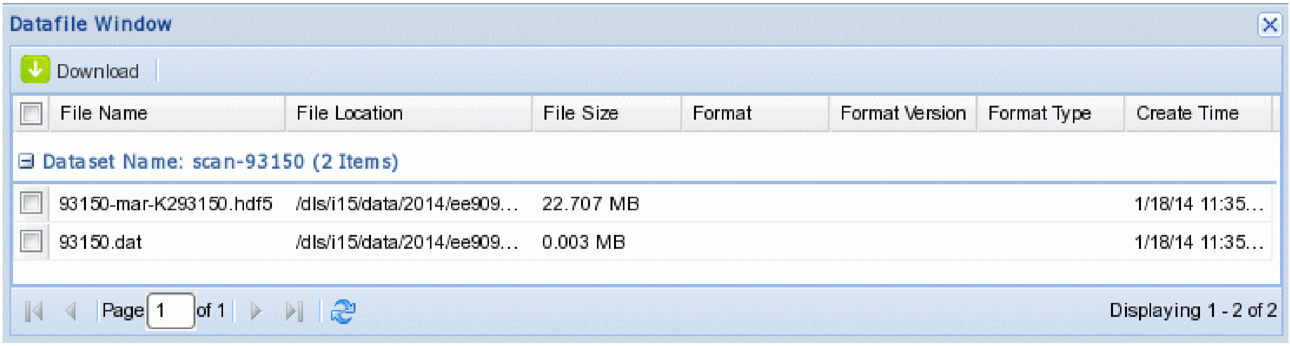
View

	Dataset Name	Status	Type	Description
☐	scan-93149	in progress	experiment_raw	
✓	scan-93150	in progress	experiment_raw	
☐	scan-93151	in progress	experiment_raw	
☐	scan-93152	in progress	experiment_raw	

Page 10 of 18 — Displaying 181 - 200 of 347

Figure 4: DLS ICAT datasets window

The data file window, Figure 5, shows the file names, their location, the file size and the create time. This window allows downloading files.



Datafile Window

Download

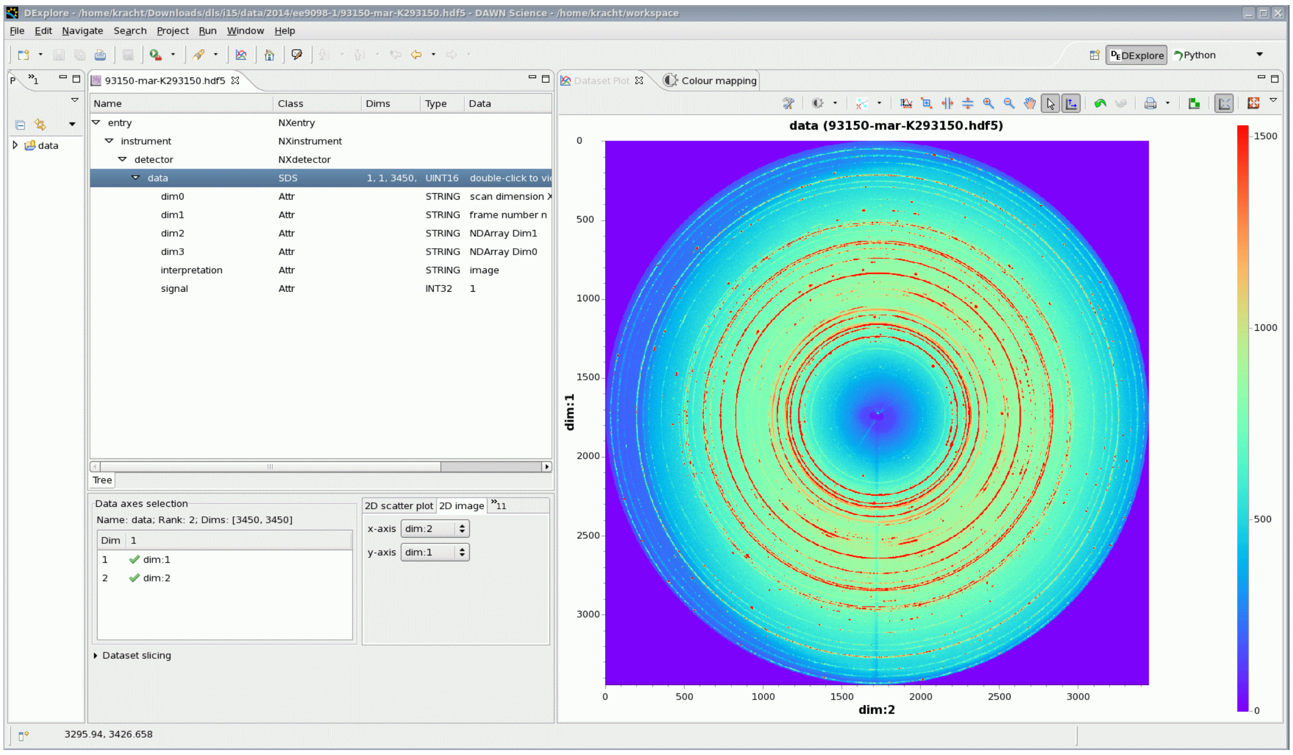
	File Name	File Location	File Size	Format	Format Version	Format Type	Create Time
	Dataset Name: scan-93150 (2 Items)						
☐	93150-mar-K293150.hdf5	/dls/i15/data/2014/ee909...	22.707 MB				1/18/14 11:35...
☐	93150.dat	/dls/i15/data/2014/ee909...	0.003 MB				1/18/14 11:35...

Page 1 of 1 — Displaying 1 - 2 of 2

Figure 5: DLS ICAT datafile window

The datasets are transferred as a zip archive preserving path names consisting of the facility name, the instrument, the year and the investigation identifier, e.g.: dls/i15/data/2014/ee9098-1.

This naming convention generates a consistent directory structure for all files that are downloaded from DLS, even if they come from different investigations. The file size is comparatively small leading to fast downloads of single datasets.


DExplore - /home/kracht/Downloads/dls/i15/data/2014/ee9098-1/93150-mar-K293150.hdf5 - DAWN Science - /home/kracht/workspace
File Edit Navigate Search Project Run Window Help
DExplore
Python
data
93150-mar-K293150.hdf5
Dataset Plot
Colour mapping
data (93150-mar-K293150.hdf5)
dim:1
dim:2
Tree
Dataset slicing

Name	Class	Dims	Type	Data
entry	NXentry			
instrument	NXinstrument			
detector	NXdetector			
data	SDS	1, 1, 3450,	UINT16	double-click to vi
dim0	Attr		STRING	scan dimension X
dim1	Attr		STRING	frame number n
dim2	Attr		STRING	NDArray Dim1
dim3	Attr		STRING	NDArray Dim0
interpretation	Attr		STRING	image
signal	Attr		INT32	1

Data axes selection

Name: data; Rank: 2; Dims: [3450, 3450]

Dim	1
1	✓ dim:1
2	✓ dim:2

2D scatter plot | 2D image

x-axis dim:2

y-axis dim:1

3295.94, 3426.658

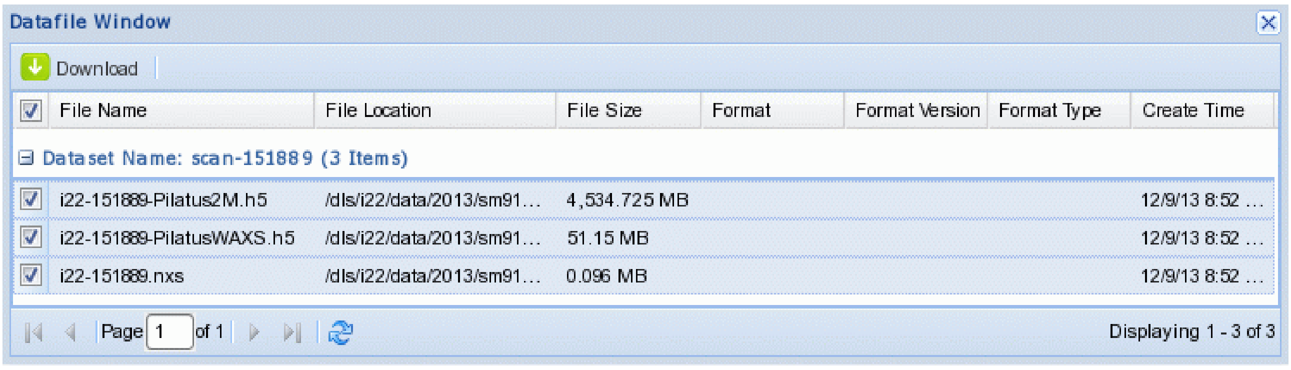
Figure 6: DAWN diffraction image

Eventually the files have to be inspected to verify that the correct data have been selected. Figure 6 shows the contents of the HDF5 file displayed by DAWN. It is an image taken by a 2D detector without any descriptive information. The metadata are stored in the associated .dat file: the date of the measurement, the ring current, the wiggler field, the monochromator energy, the exposure time and other details.

3.3 VL2: Small angle scattering

This example is based on data which have been measured during a beamtime at DLS in autumn 2013. The measurement was carried out by a group of 9 researchers from DESY and DLS. The beamtime began on December 7, 2013 and lasted 6 days. The total number of datasets created is 1438.

Since the VL2 data discovery procedure is very similar to VL1, the whole process is not repeated. This section focusses instead on the differences. This time each dataset comprises a NeXus file holding general information and two HDF5 files with the frames created by a Pilatus2M and a Pilatus300k detector, Figure 7. The Pilatus2M covers the small-angle regime, the Pilatus300k the wide-angle area. The NeXus file contains details about the sample (name and thickness), the monochromator (energy), the source (beam current, beam energy), the detectors (beam center, counting time, sample distance, pixel size), the x-ray flux, the version of the experiment control program (GDA), a user-supplied title, etc. The sample distance, the energy and the detector tilt are crucial parameters for deriving reciprocal space positions from pixel positions. Therefore, additional calibration measurements are conducted to determine these parameters with the best possible resolution.



Datafile Window

Download

Dataset Name: scan-151889 (3 Items)

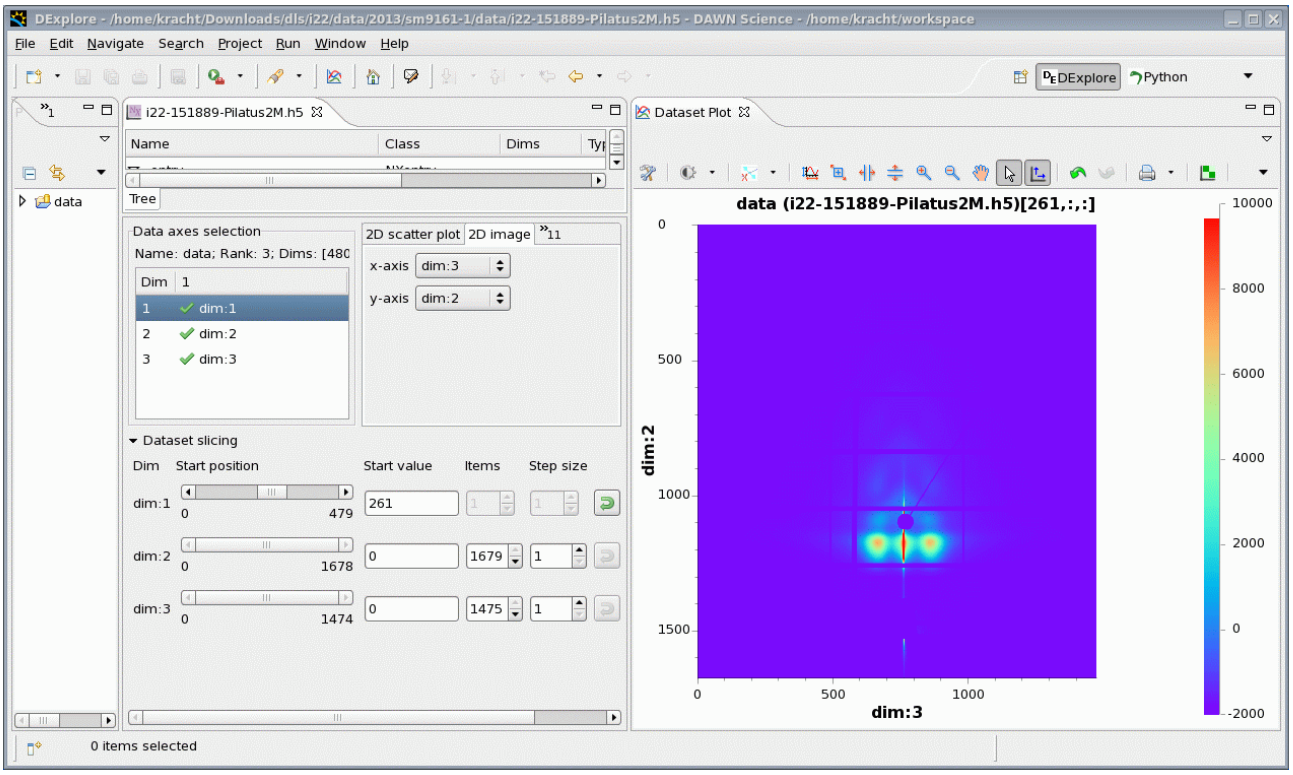
File Name	File Location	File Size	Format	Format Version	Format Type	Create Time
i22-151889-Pilatus2M.h5	/dls/i22/data/2013/sm91…	4,534.725 MB				12/9/13 8:52 …
i22-151889-PilatusWAXS.h5	/dls/i22/data/2013/sm91…	51.15 MB				12/9/13 8:52 …
i22-151889.nxs	/dls/i22/data/2013/sm91…	0.096 MB				12/9/13 8:52 …

Page 1 of 1 Displaying 1 - 3 of 3

Figure 7: DLS ICAT VL2 datafile window

The dataset scan-151889 has been downloaded to measure the transfer speed. The size of this dataset is 4.8 GB. The transfer speed varies from 5 to 30 MB/sec with an average of 15 MB/s. Data retrieval, which may involve tape access, adds 2 – 5 min to the total download time.

Like in VL1, the file discovery and download is finished by a quick inspection using DAWN, Figure 8.



DExplore - /home/kracht/Downloads/dls/i22/data/2013/sm9161-1/data/i22-151889-Pilatus2M.h5 - DAWN Science - /home/kracht/workspace

Data axes selection
Name: data; Rank: 3; Dims: [480

Dim	1
1	✓ dim:1
2	✓ dim:2
3	✓ dim:3

2D image: x-axis dim:3, y-axis dim:2

Dataset slicing

Dim	Start position	Start value	Items	Step size
dim:1	0 – 479	261	1	1
dim:2	0 – 1678	0	1679	1
dim:3	0 – 1474	0	1475	1

0 items selected

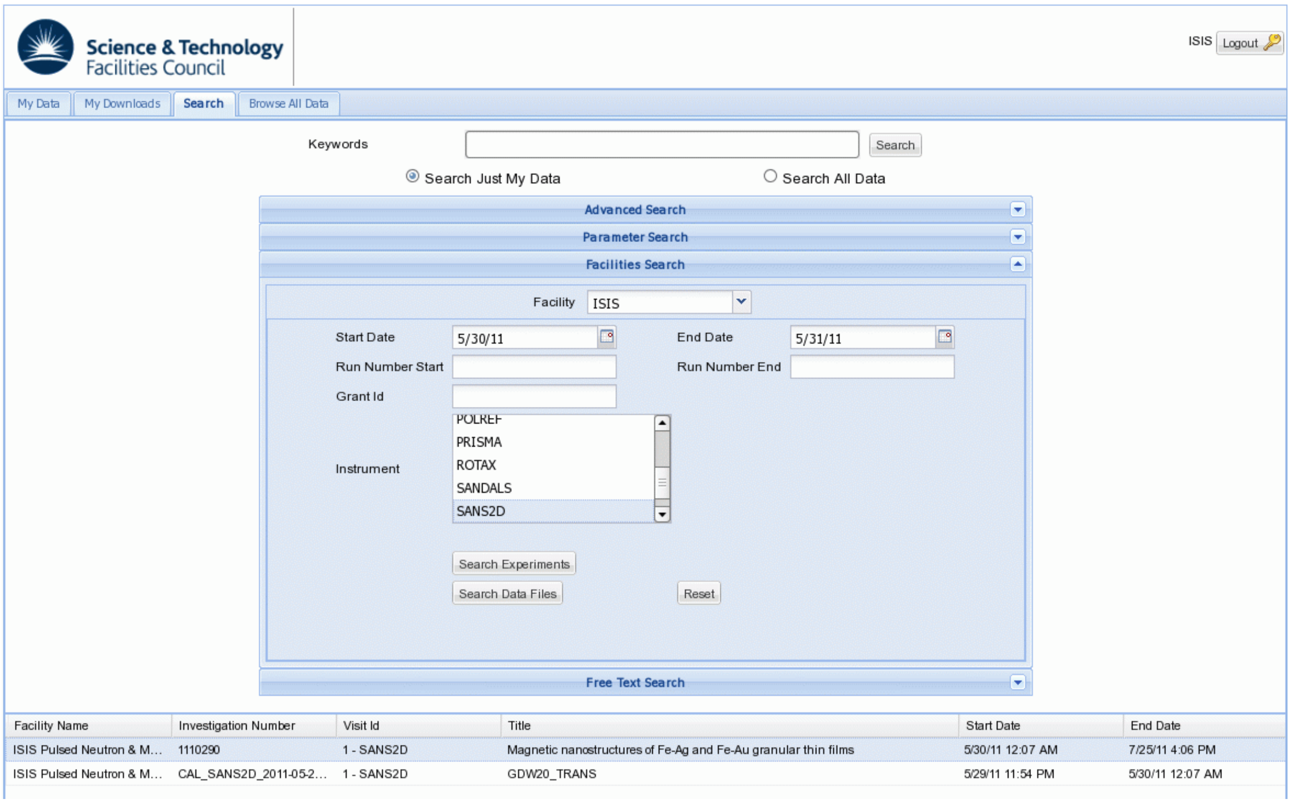
Figure 8: Pilatus image displayed with DAWN

3.4 ISIS ICAT

The two preceding demonstrations deal with data measured at DLS, a synchrotron radiation facility. This section deals with data management at a Neutron source. ISIS is an excellent example because it produces Neutrons (and Muons) and it implemented the PaN-data policy making data public after an embargo period of 3 years. Since it is interesting to study how open access data is discovered by other researchers, we will give special attention to this subject. Last but not least, it is worth mentioning in this context that ISIS is the place where ICAT was created.

Let’s assume that we are interested in a measurement of magnetic nanostructures which has been carried out by a team at the SANS2D instrument more than 3 years ago. Hence the data are public. We can register to the ISIS ICAT service with our email addresses without any other formalities. After a successful login, TOPCAT offers us several tools to search for the data. We will choose one of these depending on our knowledge about the investigation of interest:

- If the instrument name, SANS2D, is the only known detail, we select it in the Facilities Search window and click on Search Experiments. TOPCAT returns 200 matching entries. This number is quite big but we would still be able to locate the dataset of interest by going through the entire list.
- If we know the start or end date, the returned list becomes shorter, Figure 9
- If we enter the Run Number Start, we have an exact match.
- We can also use the general keyword search provided at the top of the screen, Figure 9. In our case we could enter the string “nanostructures”. This would result in a list with two entries.



Keywords

(● Search Just My Data) ○ Search All Data

Facility	ISIS		
Start Date	5/30/11	End Date	5/31/11
Run Number Start		Run Number End	
Grant Id			
Instrument	POLREF, PRISMA, ROTAX, SANDALS, [✓ SANS2D]		

Facility Name	Investigation Number	Visit Id	Title	Start Date	End Date
ISIS Pulsed Neutron & M...	1110290	1 - SANS2D	Magnetic nanostructures of Fe-Ag and Fe-Au granular thin films	5/30/11 12:07 AM	7/25/11 4:06 PM
ISIS Pulsed Neutron & M...	CAL_SANS2D_2011-05-2...	1 - SANS2D	GDW20_TRANS	5/29/11 11:54 PM	5/30/11 12:07 AM

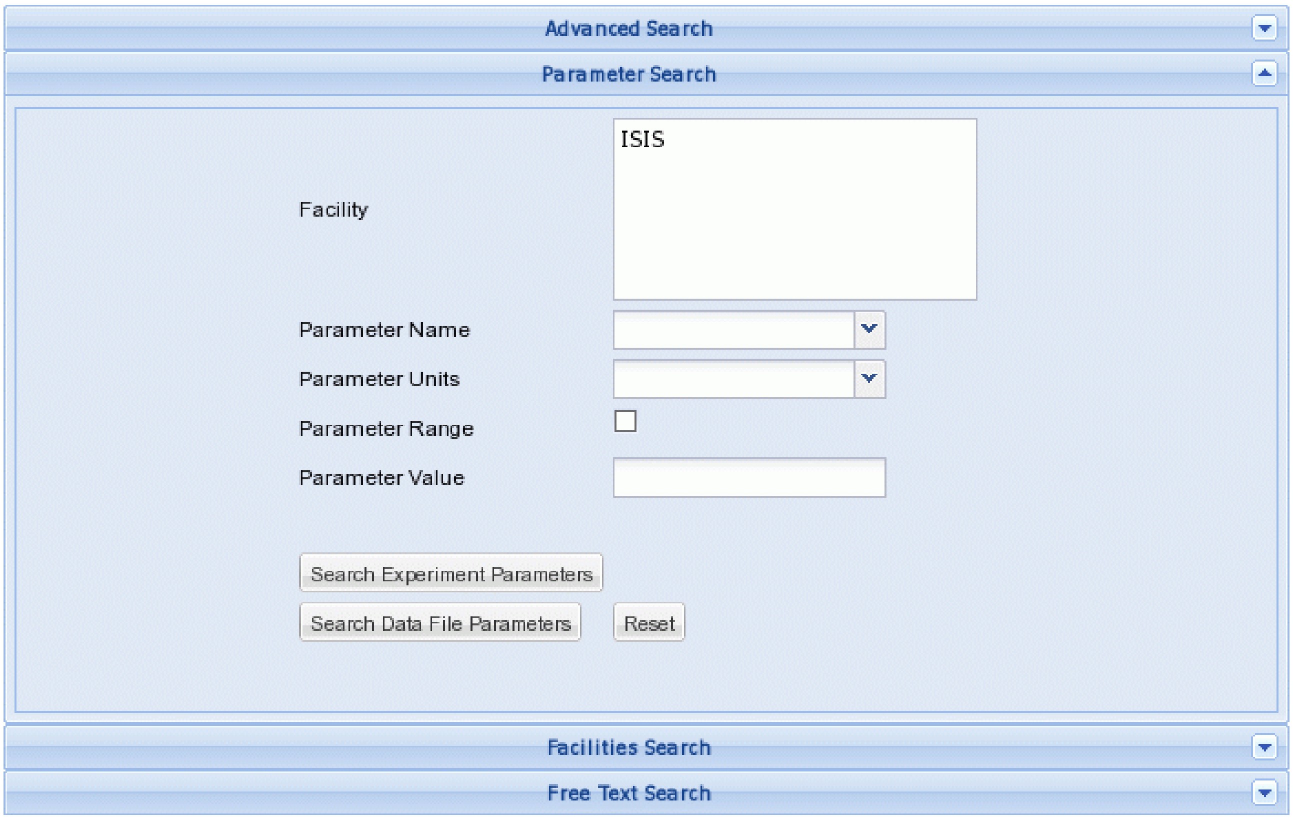
Figure 9: ISIS ICAT facilities search

Sophisticated searches are also supported by means of Advanced Search and Parameter Search windows. Even if these searches are so far not fully implemented it is still interesting to see how e.g. the Parameters Search can be used, Figure 10. The scientist may specify a parameter name and search for a specific value or a range of values. The TOPCAT interface has to be prepared in a way that users can select valid parameter names from a list to be provided in a drop-down menu. Since one of the goals of PaN-data ODI is to facilitate federated searches across institute borders, the participating research infrastructures have to agree on controlled vocabularies before parameter-based searches can reasonably be implemented.

In the foregoing paragraphs we made no assumptions about the relations between the original PI and the researcher intending to re-analyze the data. The PaN-data policy proposes that the original

PI should be contacted to inform them and suggest a collaboration if appropriate. Experience shows that data can only be analyzed, if the whole experimental procedure is known. In general, this requires also insight into hand-written annotations recorded by the original research team. Therefore both parties have definitely to come to an agreement. If we assume that TOPCAT is used under this condition the search tools are perfectly suited to fulfill their task.

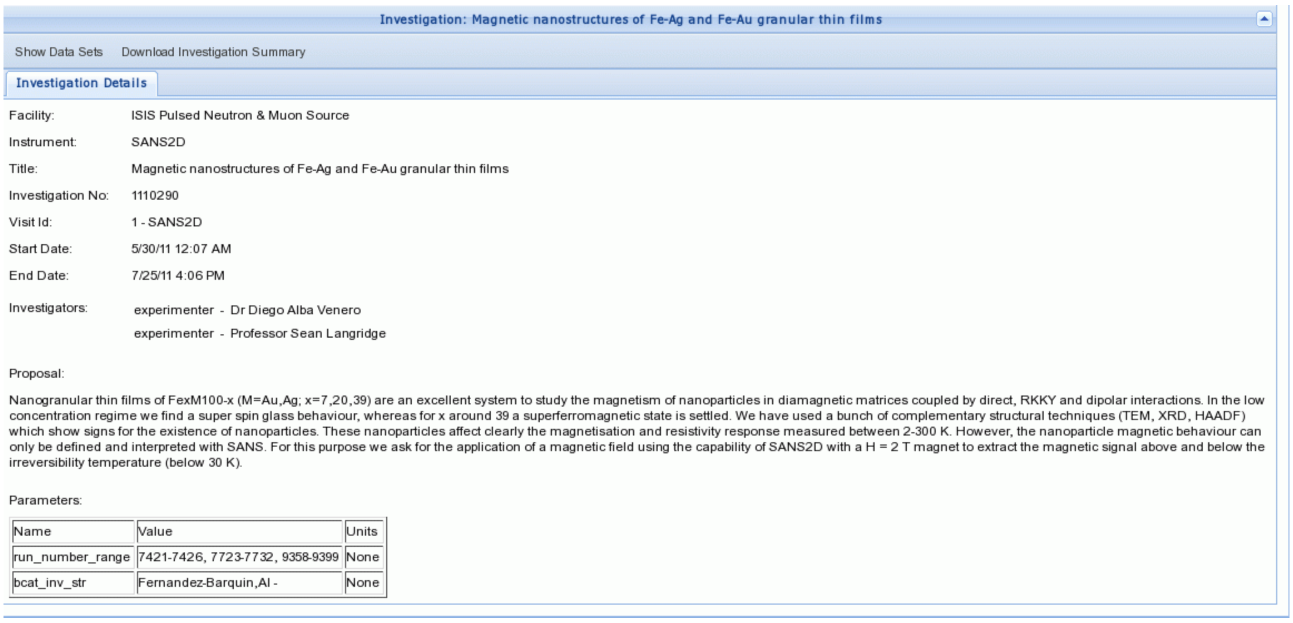
Let us eventually consider the blind-search scenario meaning that the data discovery is carried out by someone how has no contact to ISIS staff members or to participants of previous investigations. Our candidate will visit the ISIS homepage, read the descriptions of the instruments and find those of interest. We have seen that data searches based on the name of an instrument alone deliver useful results. We expect that our candidate will study the investigation summaries, as in Figure 3, find the PI, contact them and proceed as described before.



Facility: ISIS

Figure 10: ISIS ICAT parameters search

A researcher who managed to discover datasets as presented in Figure 9 needs to inspect other investigation details. The information can be retrieved by clicking on the investigation name. Figure 11 shows the investigation summary. A comparison with the corresponding DLS summary shows hardly any difference: a few names deviate and ISIS has a small parameter section containing the run numbers.



Investigation: Magnetic nanostructures of Fe-Ag and Fe-Au granular thin films

Show Data Sets Download Investigation Summary

Investigation Details

Facility:	ISIS Pulsed Neutron & Muon Source
Instrument:	SANS2D
Title:	Magnetic nanostructures of Fe-Ag and Fe-Au granular thin films
Investigation No:	1110290
Visit Id:	1 - SANS2D
Start Date:	5/30/11 12:07 AM
End Date:	7/25/11 4:06 PM
Investigators:	experimenter - Dr Diego Alba Venero experimenter - Professor Sean Langridge

Proposal:

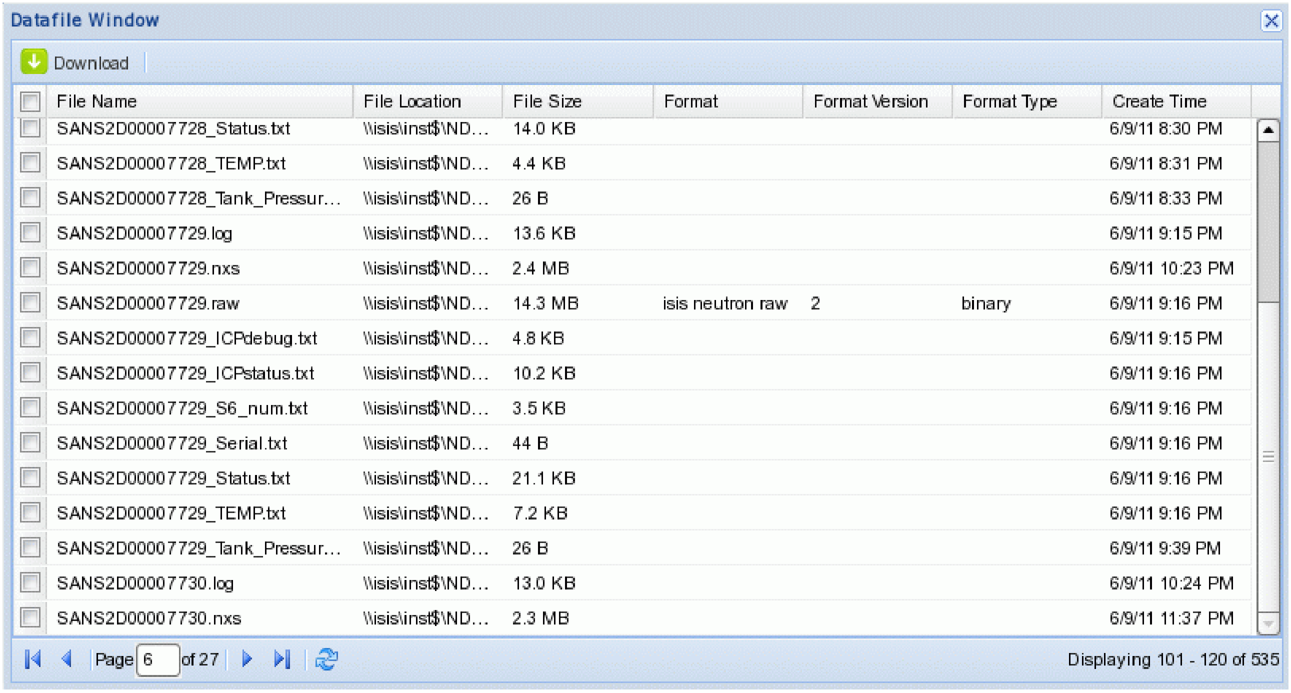
Nanogranular thin films of FexM100-x (M=Au,Ag; x=7,20,39) are an excellent system to study the magnetism of nanoparticles in diamagnetic matrices coupled by direct, RKKY and dipolar interactions. In the low concentration regime we find a super spin glass behaviour, whereas for x around 39 a superferromagnetic state is settled. We have used a bunch of complementary structural techniques (TEM, XRD, HAADF) which show signs for the existence of nanoparticles. These nanoparticles affect clearly the magnetisation and resistivity response measured between 2-300 K. However, the nanoparticle magnetic behaviour can only be defined and interpreted with SANS. For this purpose we ask for the application of a magnetic field using the capability of SANS2D with a H = 2 T magnet to extract the magnetic signal above and below the irreversibility temperature (below 30 K).

Parameters:

Name	Value	Units
run_number_range	7421-7426, 7723-7732, 9358-9399	None
bcat_inv_str	Fernandez-Barquin,Al -	None

Figure 11: ISIS ICAT investigation summary

The next question to be addressed is how ISIS organizes the data. Figure 12 shows the ISIS data-file window for data being measured within scan number 7729.



Datafile Window

Download

File Name	File Location	File Size	Format	Format Version	Format Type	Create Time
SANS2D00007728_Status.txt	\\isis\inst$\ND...	14.0 KB				6/9/11 8:30 PM
SANS2D00007728_TEMP.txt	\\isis\inst$\ND...	4.4 KB				6/9/11 8:31 PM
SANS2D00007728_Tank_Pressur...	\\isis\inst$\ND...	26 B				6/9/11 8:33 PM
SANS2D00007729.log	\\isis\inst$\ND...	13.6 KB				6/9/11 9:15 PM
SANS2D00007729.nxs	\\isis\inst$\ND...	2.4 MB				6/9/11 10:23 PM
SANS2D00007729.raw	\\isis\inst$\ND...	14.3 MB	isis neutron raw	2	binary	6/9/11 9:16 PM
SANS2D00007729_ICPdebug.txt	\\isis\inst$\ND...	4.8 KB				6/9/11 9:15 PM
SANS2D00007729_ICPstatus.txt	\\isis\inst$\ND...	10.2 KB				6/9/11 9:16 PM
SANS2D00007729_S6_num.txt	\\isis\inst$\ND...	3.5 KB				6/9/11 9:16 PM
SANS2D00007729_Serial.txt	\\isis\inst$\ND...	44 B				6/9/11 9:16 PM
SANS2D00007729_Status.txt	\\isis\inst$\ND...	21.1 KB				6/9/11 9:16 PM
SANS2D00007729_TEMP.txt	\\isis\inst$\ND...	7.2 KB				6/9/11 9:16 PM
SANS2D00007729_Tank_Pressur...	\\isis\inst$\ND...	26 B				6/9/11 9:39 PM
SANS2D00007730.log	\\isis\inst$\ND...	13.0 KB				6/9/11 10:24 PM
SANS2D00007730.nxs	\\isis\inst$\ND...	2.3 MB				6/9/11 11:37 PM

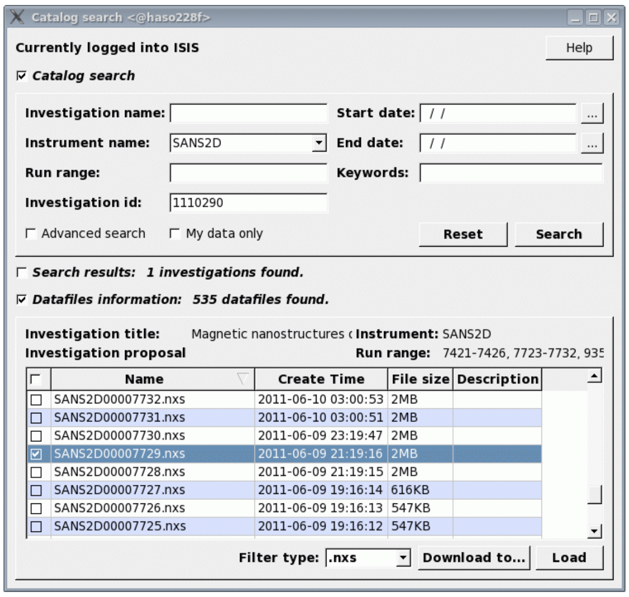
Page 6 of 27 Displaying 101 - 120 of 535

Figure 12: ISIS ICAT datafile window

The dataset contains files in different formats: raw, ASCII and NeXus. In fact, the NeXus file holds the complete information. ISIS provides the other files only to assist users running software which is not NeXus compliant. This strategy is certainly well suited to avoid lengthy discussions about incompatibilities between the NeXus format and analysis software.

Transferring files from ISIS is faster compared to DLS because the datasets are smaller. This is basically due to the fact that neutron sources generate less flux than x-ray facilities. As a consequence all files belonging to an investigation can be packed into a single zip-archive before it is downloaded. Again, the directory structure is preserved helping researches to organize files from different beamtimes or different instruments.

The fact that ISIS generates comparatively small datasets has another great advantage. Investigators can run Mantid local at their home institutes and connect to the ISIS ICAT to directly access data for inspection and download. Here is an example. Figure 13 shows the result of an ICAT search using Mantid. A filter has been set. Therefore only NeXus files are displayed. A click on the Load button loads the data into the Mantid workspace where it can be visualized or processed.



Catalog search <@haso228f>

Currently logged into ISIS

☑ *Catalog search*

Investigation name:		Start date:	/ /
Instrument name:	SANS2D	End date:	/ /
Run range:		Keywords:	
Investigation id:	1110290		

☐ Advanced search ☐ My data only

☐ *Search results: 1 investigations found.*

☑ *Datafiles information: 535 datafiles found.*

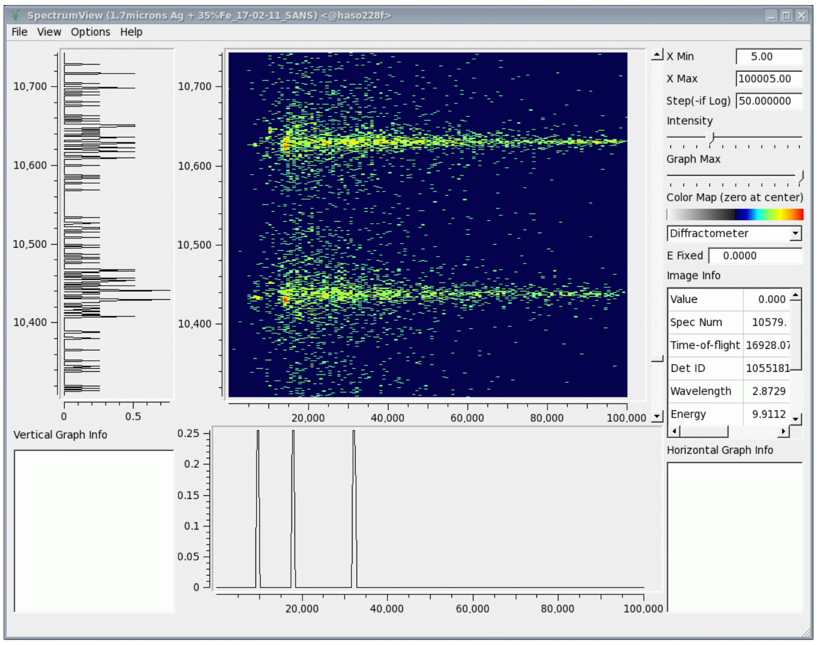
Investigation title: Magnetic nanostructures (Instrument: SANS2D
Investigation proposal Run range: 7421-7426, 7723-7732, 935

☐	Name	Create Time	File size	Description
☐	SANS2D00007732.nxs	2011-06-10 03:00:53	2MB	
☐	SANS2D00007731.nxs	2011-06-10 03:00:51	2MB	
☐	SANS2D00007730.nxs	2011-06-09 23:19:47	2MB	
☑	SANS2D00007729.nxs	2011-06-09 21:19:16	2MB	
☐	SANS2D00007728.nxs	2011-06-09 21:19:15	2MB	
☐	SANS2D00007727.nxs	2011-06-09 19:16:14	616KB	
☐	SANS2D00007726.nxs	2011-06-09 19:16:13	547KB	
☐	SANS2D00007725.nxs	2011-06-09 19:16:12	547KB	

Filter type: .nxs

Figure 13: ISIS ICAT Mantid search

The Mantid SpectrumView, Figure 14, displays an image with 420 time-of-flight histograms each one with 152 bins. The total number of histograms is 73736. They are ordered in y-direction. The display area can be moved with a scroll bar. The user selects an individual histogram with the cursor function by clicking into the display area. It is displayed at the bottom of the widget. The click into the display area also creates a spectrum containing a cut through the histograms for a given time-of-flight. It is displayed on the left side of the widget.



SpectrumView (1.7microns Ag + 35%Fe_17-02-11_SANS) <@haso228f>

X Min 5.00
X Max 100005.00
Step(-if Log) 50.000000
Color Map (zero at center): Diffractometer
E Fixed 0.0000

Image Info

Value	0.000
Spec Num	10579.
Time-of-flight	16928.07
Det ID	1055181
Wavelength	2.8729
Energy	9.9112

Figure 14: ISIS ICAT Mantid SpectrumView

ISIS has a list of metadata that is common to all NeXus files: beamline, experiment identifier, name, proton charge, run number, run cycle and details about the sample (distance, height, name, shape, thickness, type and width), etc. In addition instrument-specific information and user-supplied annotations are written to the output files.

3.5 Summary

The following list summarizes essential facts related to data searches and downloads as described in the foregoing sections. Comments from researchers are included:

- The ICAT entries are hierarchically organized in terms of investigations, datasets and files.
- TOPCAT offers powerful web tools to perform data searches. The following metadata play an important role for the identification of files:
    - Investigation number and title
    - Start date, end date
    - Instrument name
    - Names of the principal investigator and co-investigators
    - Dataset name, path, size, create date
- The response times to metadata queries are short making the discovery procedure very efficient.
- The average data transfer speed, from DLS to DESY, has been measured to be 15 MB/s
- Datasets are identified by the scan number. Scientists have to retrieve this number from the log book. They would appreciate if the dataset window were extended to display also user-supplied comments, which are stored in the files. This additional piece of information would help the researchers to recognize more easily the data of interest.
- The contents of the datasets show some differences. In both VLs image frames are stored in HDF5 files. VL1 uses ASCII files for the metadata whereas VL2 stores them in NeXus files. We learned that ISIS provides the researchers with NeXus files containing all data and metadata. This service is definitely preferable but if larger data volumes have to be managed the ISIS approach may not be adequate.
- In the preceding section we have seen how raw data are accompanied by technical and administrative information. However, these metadata are not sufficient to gain a complete insight in the experimental procedure performed. What is missing are the hand-written notes recorded by the original investigators. It is important to make this information available for scientists intending to re-analyze data. Work package 7 (preservation) developed the prototype of an electronic logbook and deployed it at various ILL stations. ILL will generate a report about the experiences with the prototype in September 2014 allowing other partners to benefit from the pilot project.
- So far neither the DLS nor the ISIS ICAT support Umbrella authentication. However, there is no principle obstacle.
- Researchers may want to download all datasets belonging to an investigation. The selection is easily possible using the datasets window, Figure 4. Unfortunately, there is no information about the total data size. Furthermore, the download process is split into multiple tasks transferring single datasets which leads to resource conflicts between these tasks. It is not possible to download the whole investigation.

4 Science 3D – The Tomography Database

Imaging experiments at light- or neutron-sources can reveal intricate details of the 3-dimensional composition of materials and biological specimen. In particular micro-tomography allows illuminating all kind of specimen in a non-destructive manner. The application of tomography, in particular synchrotron radiation X-ray micro-tomography, permits for example to determine the stunning anatomy and morphology of ancient insects captured in amber or dinosaurs embryos in fossil dinosaur eggs. Comparative studies can shed light on evolutionary pathways and help to shape phylogenetic trees or assign proper evolutionary taxonomies.

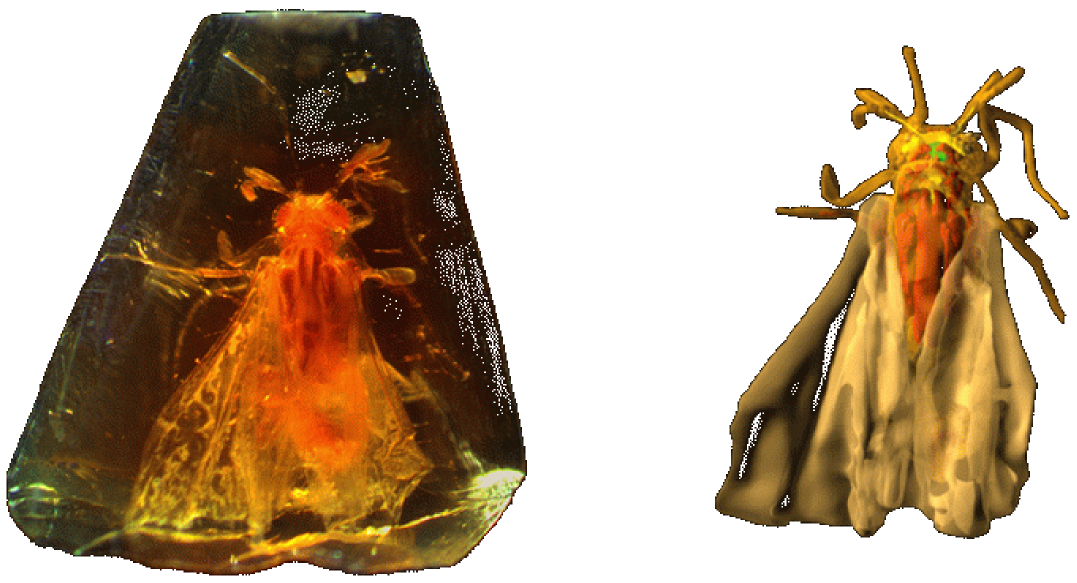


Figure 15: H.Pohl et al., Naturwissenschaften. 2010 Sep;97(9):855-9. doi: 10.1007/s00114-010-0703-x: *Reconstructing the anatomy of the 42-million-year-old fossil Mengea tertiaria (Insecta, Strepsiptera).*

The 3-D models are carefully segmented and annotated by the scientists, often a manual and rather tedious procedure. As such, 3D-models are fantastic materials for education and the general public. Common practice is however to publish the results in conventional print-journals showing selected 2D-projections of the full 3D-model, and occasionally add or rather hide the 3D-model in proprietary formats in supplements to the publication. Discovery of the materials and models becomes tremendously complicated for interested 3rd parties like students, educators or teachers. Likewise, quantitative comparative studies are severely hampered by the use of non-standard data formats and the lack of solid quality indicators. In brief: a substantial part of the scientists work creating a 3-dimensional model shamefully remains unpublished, undiscovered or unused.

The Science3D project aims to overcome some of the hurdles in publishing, presenting and promoting data and models from tomographic reconstructions. The project focusses primarily on biological specimen investigated by SrXµ-tomography at the initial stage, but is open to all imaging techniques and 3D-models regardless if it originates for example from x-ray tomography or electron microscopy and studies materials, biological specimen or any other 3-D object. The common denominator is however the open accessibility of all data relating to a scientific investigation and the proper annotation of the materials supporting data sharing and re-use.

To achieve this, the consent and support by the researchers performing the experiments is indispensable. The Helmholtz-Center Geesthacht (HZG) is operating the imaging beamlines at PETRA III and is supporting their users in all aspects of the data analysis, and is hence a perfect partner to promote the Science-3D project in the SrXμ-tomography. Consequently the entire Science-3D project has been designed as a joint project of the HZG and PaNdata ODI.

The Science-3D data catalog serves as a Virtual Laboratory demonstrator for tomography (VL3), but extends beyond the original scope of the virtual lab, and is operated as a sustainable platform for publishing, sharing and promoting research from imaging applications.

4.1 The Science 3D kick-off workshop

We have been in contact with several users of the tomography-community for quite a while to gather their requirements and concerns on the idea of an open access data catalogue. This dialog was quite fruitful in shaping the Science3D-portal. To increase the user base, we organized together with the HZG a Science3D workshop which brought together the user community, the Science3D team, the beamline scientists at PETRA III and a number of educators and teachers. The Science3D workshop served as a user-meeting as well as a project kick-off, covering different aspects of the data lifecycle from pre-experiment stages to the re-use after publication.

We specifically invited school teachers and teachers' educators to gather feedback from them on possible uses of Science3D materials in the classroom. Fortunately, six school teachers followed the invitation and joined the Science3D workshop on the first day, which was intended to provide talks in a scholarly manner. The speakers made accordingly quite an effort avoiding too much technical details and kept the presentations well understandable for non-experts.

The presentations covered a rather wide spectrum of topics around x-ray tomography, ranging from a basic introduction of the experimental and computational methods to unpublished results in evolutionary biology and nano-sciences. The presentations hence gave a very nice overview of the current status of the research at a non-expert-level. A particular emphasis was on the visualization of the 3D-models with free or affordable applications, a pre-requisite for the participation of students and schools.

The ability to use the 3D-models for rapid-prototyping (3D-print) is another particular appealing feature, since it permits to take real 3D-objects into the classroom. Models need however to be prepared in a specific manner, which isn't always easily achieved. To provide the tools and expertise to work with rapid prototyping of tomographic 3D-models we will organize another Science-3D hands-on workshop later on.

The science3D-workshop was held in Hamburg at DESY, June 02-04 2014 and was visited by roughly 50 participants (with some variations, since some sessions where dedicated to specific technical aspects). The slides are currently being prepared – avoiding premature release of unpublished data – and will be deposited on the meeting page under https://indico.desy.de/event/science3d.

4.2 Concept of the Science 3D open access catalogue

The concept of the Science3D catalog was kept as simple and as modular as possible. The proposed setup consists of a small number of modules serving distinct tasks (see Figure 16).

The frontend is based on the Drupal content management system. Drupal[4] is highly modular and reasonably flexible to be customized meeting the Science3D requirements. Drupal brings an easy way to add content, implement DOI resolution or populate taxonomies. Drupal also offers a straightforward implementation of different authentication modules, in particular for OpenID and Shibboleth/SAML.

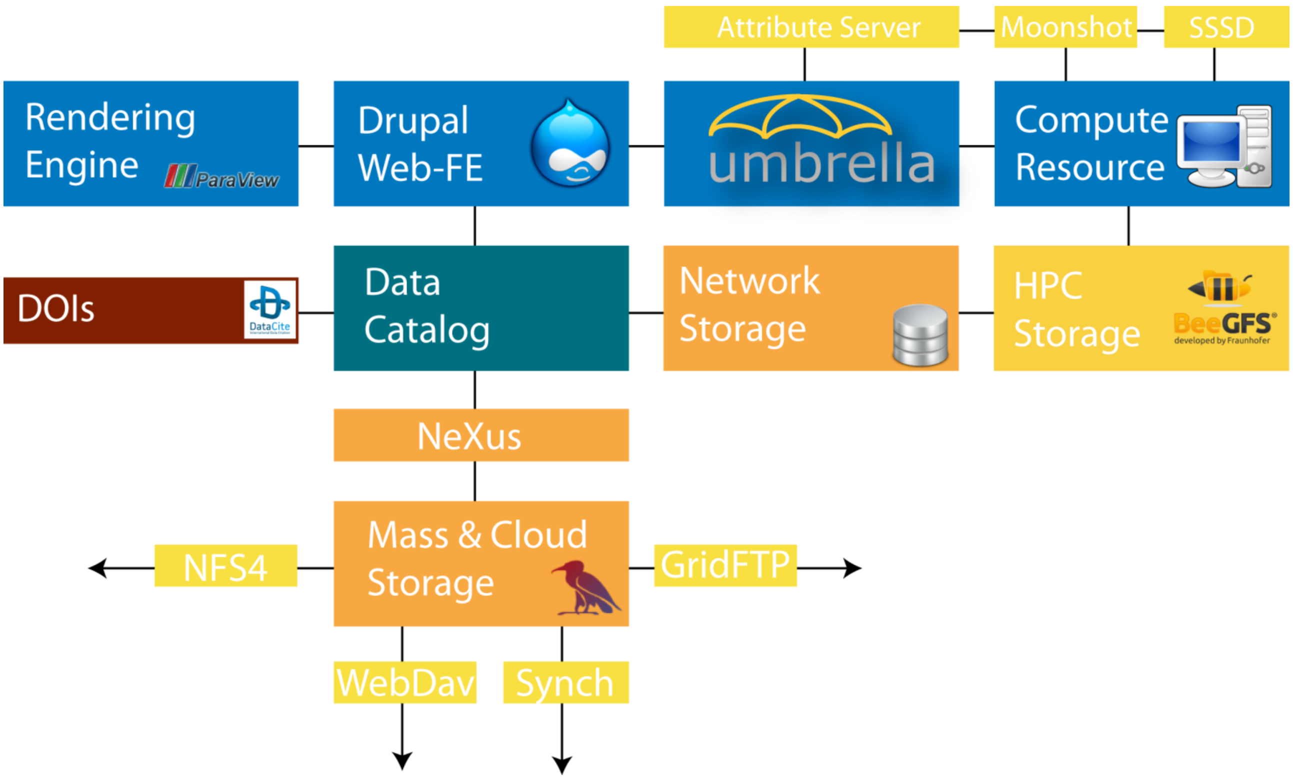


Figure 16: Schema of the proposed Science3D architecture

Science3D not only provides access to raw scientific data, but also to an arbitrary number of additional materials, in particular 3D-models for visualization and printing, publication and supplements, movies and other supporting media. The raw scientific data are stored on the internal dCache-System, which can however be replaced by any posix-.compliant storage element. dCache[5] hence serves as the primary mass storage. All other materials and media, mostly uploaded by the scientists themselves, are stored on a network-storage with snapshot and backup capabilities. Currently we are dealing exclusively with data which had resulted in a publication, and are hence legacy data in non-standard formats. For re-use and publication data need to be reformatted and repackaged, meta-data to be extracted from various text-files. The NeXus integration is progressing very well, which will permit automatic ingestion and accessibility without this procedure.

The dCache storage allows to access scientific data not only through the the WebFE but also via a number of different pathways. For external access the WebDav and GridFTP protocols are most convenient. For internal access for example from compute resources access is realized through the pNFS4.1 interface. Currently a prototype for an owncloud-dCache hybrid is being tested which

[4] http://drupal.com/

[5] http://www.dcache.org/

implements the owncloud[6] synchronization mechanism which very conveniently synchronizes data from/to local storage to the dCache cloud.

UmbrellaID will serve as the primary authenticator. The ability to integrate UmbrellaID into the WebFE has been successfully tested on the Science3D prototype system. On the production system, UmbrellaID is currently being disabled until Science3D- and moonshot-developments are sufficiently stable. It is intended to use moonshot as the interface between the UmbrellaID and the compute services for 3D-reconstruction and modelling. STFC, PSI together with NRENs Switch and Janet (as well as edugain and Geant through prototype projects), are deeply involved in the moonshot developments and implementation. DESY is an early prototype tester. Developments are progressing rapidly. The fixes required for openssh and various clients are however not yet integrated into the corresponding upstream packages. Therefore it is too cumbersome for science3D users to use the moonshot system, and the moonshot system in a too early stage to be part of the science3D production system. Access to compute resources is therefore currently granted on a conventional way (via krb5/ldap).

Science3D is serving as the landing page for DOI resolution. For that purpose, Science3D uses Drupals parametric re-write engine, which translates a specific DOI or rather URL (http://www.science3d.org/doi/<doi>) to the internal node, for both publications as well as scientific data. This permits to easily crosslink between publications and underlying scientific data. The DOI lookup is fully functional. Frequently, a publication refers to several datasets. To resolve this one-to-many relation, Science3D uses collections of datasets with a separate DOI, individual datasets are identified through taxonomy terms defining relation between datasets. Through taxonomies, it is also possible to identify all publications or datasets referencing a specific instrument or facility. A taxonomy for biological specimen (i.e. the entire phylogenetic tree) is currently not implemented, but would permit to also identify relation between datasets and publication based on the samples phylogeny.

The internal HPC cluster is available to all Science3D depositors. The resource is currently providing about 3500 cpu cores and 10 nodes equipped with single or dual GPUs, so it’s a comparatively small HPC resource. Connected to the compute nodes is – next to dCache and network storage – a fast cluster file system (BeeGFS). Remote access is possible through a very fast load-balanced graphical interface based on the commercial product StarNet FastX[7]. The system has proved to be fully sufficient to do remote graphical rendering. For rendering of complex 3D-models a single node might not also be sufficient. We intend to enhance the service by a distributed ParaView rendering engine, which is however not yet fully implemented.

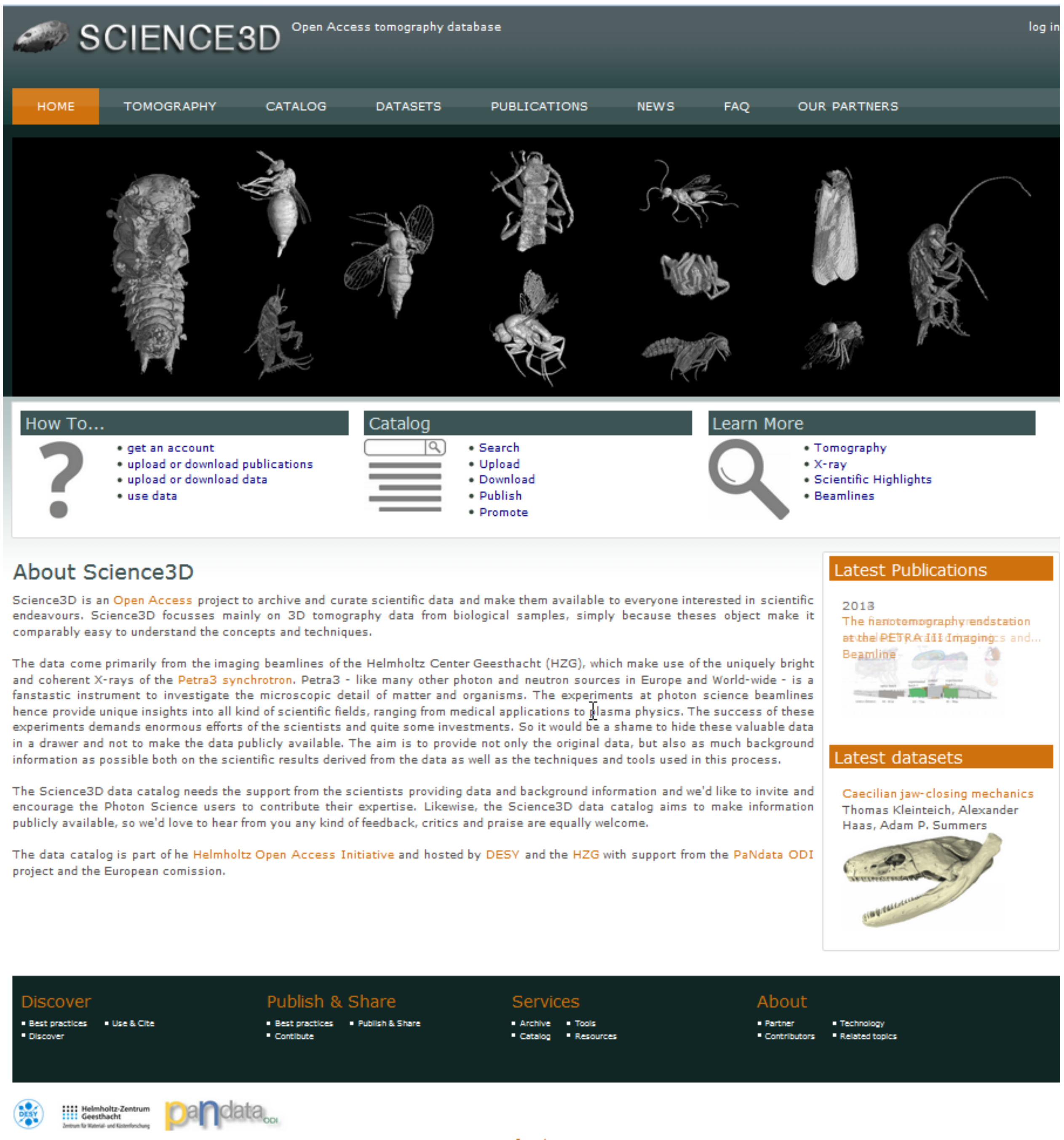
The HPC cluster provides a simple scheduler which guarantees exclusive use of nodes during a specified time. The ability to schedule access to compute resources well in advance to be able doing the computational tasks shortly after an experiment was a requirement by the user communities. The HPC cluster implements the UFO-Framework[8] developed by KIT, which permits a very fast, GPU-capable tomographic reconstruction. Work to also support other frameworks like the Eur.XFEL developed karabo and other imaging techniques like Electron Microscopy is underway.

Typical software like tomopy is also available on the HPC farm. A complete list of available software will be published on the Science3D web.

[6] https://owncloud.org/

[7] https://www.starnet.com/fastx/

[8] http://ufo.kit.edu/



How To...

- get an account
- upload or download publications
- upload or download data
- use data

Catalog

- Search
- Upload
- Download
- Publish
- Promote

Learn More

- Tomography
- X-ray
- Scientific Highlights
- Beamlines

About Science3D

Science3D is an Open Access project to archive and curate scientific data and make them available to everyone interested in scientific endeavours. Science3D focusses mainly on 3D tomography data from biological samples, simply because theses object make it comparably easy to understand the concepts and techniques.

The data come primarily from the imaging beamlines of the Helmholtz Center Geesthacht (HZG), which make use of the uniquely bright and coherent X-rays of the Petra3 synchrotron. Petra3 - like many other photon and neutron sources in Europe and World-wide - is a fanstastic instrument to investigate the microscopic detail of matter and organisms. The experiments at photon science beamlines hence provide unique insights into all kind of scientific fields, ranging from medical applications to plasma physics. The success of these experiments demands enormous efforts of the scientists and quite some investments. So it would be a shame to hide these valuable data in a drawer and not to make the data publicly available. The aim is to provide not only the original data, but also as much background information as possible both on the scientific results derived from the data as well as the techniques and tools used in this process.

The Science3D data catalog needs the support from the scientists providing data and background information and we'd like to invite and encourage the Photon Science users to contribute their expertise. Likewise, the Science3D data catalog aims to make information publicly available, so we'd love to hear from you any kind of feedback, critics and praise are equally welcome.

The data catalog is part of he Helmholtz Open Access Initiative and hosted by DESY and the HZG with support from the PaNdata ODI project and the European comission.

Latest Publications

2013
The nanotomography endstation at the PETRA III Imaging Beamline

Latest datasets

Caecilian jaw-closing mechanics
Thomas Kleinteich, Alexander Haas, Adam P. Summers

Figure 17: Snapshot of the Science3D web

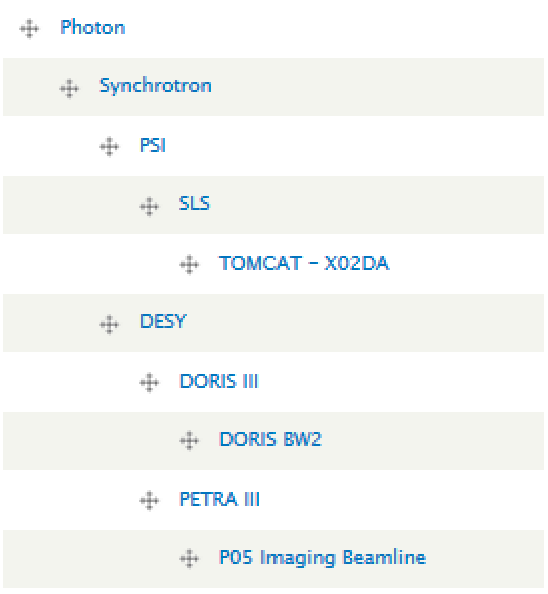
4.3 Current status of the Science 3D open access catalogue

Science3D will offer next to publications, materials and scientific data, a comprehensive overview of the experimental techniques and tools to process data or visualize 3D-models. The content base has been established, and content is continuously being added to the site, but is still far from being complete. In particular tutorials and documentation helping scientists and non-expert to visualize and interpret the 3D-models requires an in-depth knowledge of the techniques and methods. Participation of the scientists and beamline staff is hence essential. The Science3D kickoff meeting has certainly helped to initiate contributions from the user communities, which for example provided otherwise unpublished 3D-models and recipes.

Thanks to feedback from the user communities the metadata model is continuously being refined. Frequently, when new materials get uploaded, it's discovered that a certain tag or support for a format would be a useful addition.

4.3.1 Schema

While the metadata schema is still evolving the instrument taxonomy has been settled. It follows a simple scheme composed of the illuminating particle, type of source, RI operating the source, facility, instrument. The taxonomy tags permit a very easy discovery of publications and data utilizing a particular instrument. Currently lacking is however a taxonomy for the experimental techniques. Identifying the instrument is actually not sufficient to identify the experimental technique utilized. An instrument might well support different imaging techniques (for example crystallography, ptychography and tomography) or different variations of a particular imaging technique depending on the beam geometry chosen. Though it's obviously not a major task implementing the taxonomy for experimental techniques, it strongly depends on an appropriate controlled vocabulary. Work on the controlled vocabulary is however still in progress.



Photon
Synchrotron
PSI
SLS
TOMCAT – X02DA
DESY
DORIS III
DORIS BW2
PETRA III
P05 Imaging Beamline

Figure 18: Section of the instrument taxonomy

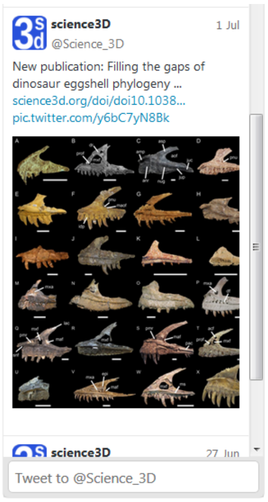
4.3.2 Catalog

The Science3D catalog has meanwhile been populated with a (limited) number of publications and datasets, additional materials like movies, 3D-models or blender-scenes. The catalog supports embedding and viewing 3D-models in the web-browser through javascript-libraries (see Figure 19). It actually works quite well since the browser is doing the rendering fully benefiting from the hardware acceleration of the host running the browser. The trade-off is however that the browser needs to load the entire 3D-model before it can start rendering the scene, which might become problematic on slow networks. The alternative is remote rendering where the browser just displays individual scenes, which might however become slow when displaying different scenes in short time (e.g rotating the 3D-model). The optimal balance between remote and local rendering still needs to be investigated. Another issue is the support for different file formats and model-associated materials. Many scientists favor the capabilities of 3D-PDFs, which support segmented 3D-models and different materials as well as preselected views. It's indeed a very convenient way of providing 3D-models to users, however the proprietary format and the lack of linux-support are troublesome. Providing substantial parts of the 3D-PDF functionality in the web-browser by suitable javascript libraries, rendering applications or plug-ins would therefore be desirable. We are currently investigating the option, but it's regardless of the choice of implementation a major task beyond the work program of this project.

4.3.3 Documentation & Dissemination

Documentation and tutorials are slowly being added to the Science3D-web. It depends to some extend on the volunteering efforts and expert knowledge of the user community as well as the beamline staff. The feedback and motivation is however very encouraging, so that we are confident that Science3D will become a very informative platform for the user communities as well as the general public.

The Science3D platform has been presented at various occasions, like workshops devoted to Open Access, webinars and science fares. The kick-off workshop has been promoted in DESY inform[9].


science3D 1 Jul
@Science_3D
New publication: Filling the gaps of dinosaur eggshell phylogeny ...
science3d.org/doi/doi10.1038...
pic.twitter.com/y6bC7yN8Bk

science3D 27 Jun

Tweet to @Science_3D

Figure 19: Science3D on twitter

4.3.4 Summary

The integration of most, but not all components has been demonstrated successfully.

- The Umbrella integration proved to be very simple on the web front end and has been demonstrated on a prototype system. It is embedded in the current production system and will be made the default authentication system at a later stage.
- The metadata scheme in Science3D contains a subset of the NeXus application definition. The NeXus schema contains all experimental details, which are not in its entirety relevant for the casual visitor and are hence omitted from the display in Science3D. A link to the full NeXus header can easily be embedded into the data page.
- NeXus and ICAT integration is still incomplete. We are currently dealing almost exclusively with legacy data in legacy formats. A user-friendly and platform independent tool to convert and annotate this kind of legacy data is in development. The need for such a tool arose from the scattered meta-data which require the data creator to complement the available information.
- A simple tool is available to convert legacy data (coming in a hand-woven format) into a browser-compliant image (tiff) format. Stack of images or entire datasets can be converted into movies or HDF5 formats for simple frame-by-frame inspection and in-depth analysis.
- DOI integration is complete. DOIs will be registered with DataCite. Contracts are in place.
- DOIs are automatically recognized by the web-FE and translated accordingly. That guarantees availability of the landing pages for DataCite references.
- Crosslinking between publications and dataset are implemented.
- Web-Visualization is partially supported; remote rendering via ParaView[10] is in progress.
- Access to HPC resources can be granted. A data processing pipeline using the UFO framework[11] is implemented on the compute resources. The UFO framework supports GPGPU hardware acceleration and can fully benefit from the mpi i/o capable cluster file

[9] http://issuu.com/zoufal/docs/desyinform_3_14_eng_web_hq_023a84051328c0/15?e=2874007/9092437

[10] http://www.paraview.org/

[11] http://ufo.kit.edu/

system on the compute resources. Likewise parallel processing with tomopy[12] is supported as well. Tomopy fully supports HDF5 (or NeXus). Tomopy is however currently not capable of multi-threaded i/o.

- Support for software frameworks and mpi i/o accelerated analysis will be further extended in the future, beyond the end of the project.

Science3D receives strong support from various parties and is guaranteed to continue after the end of the project.

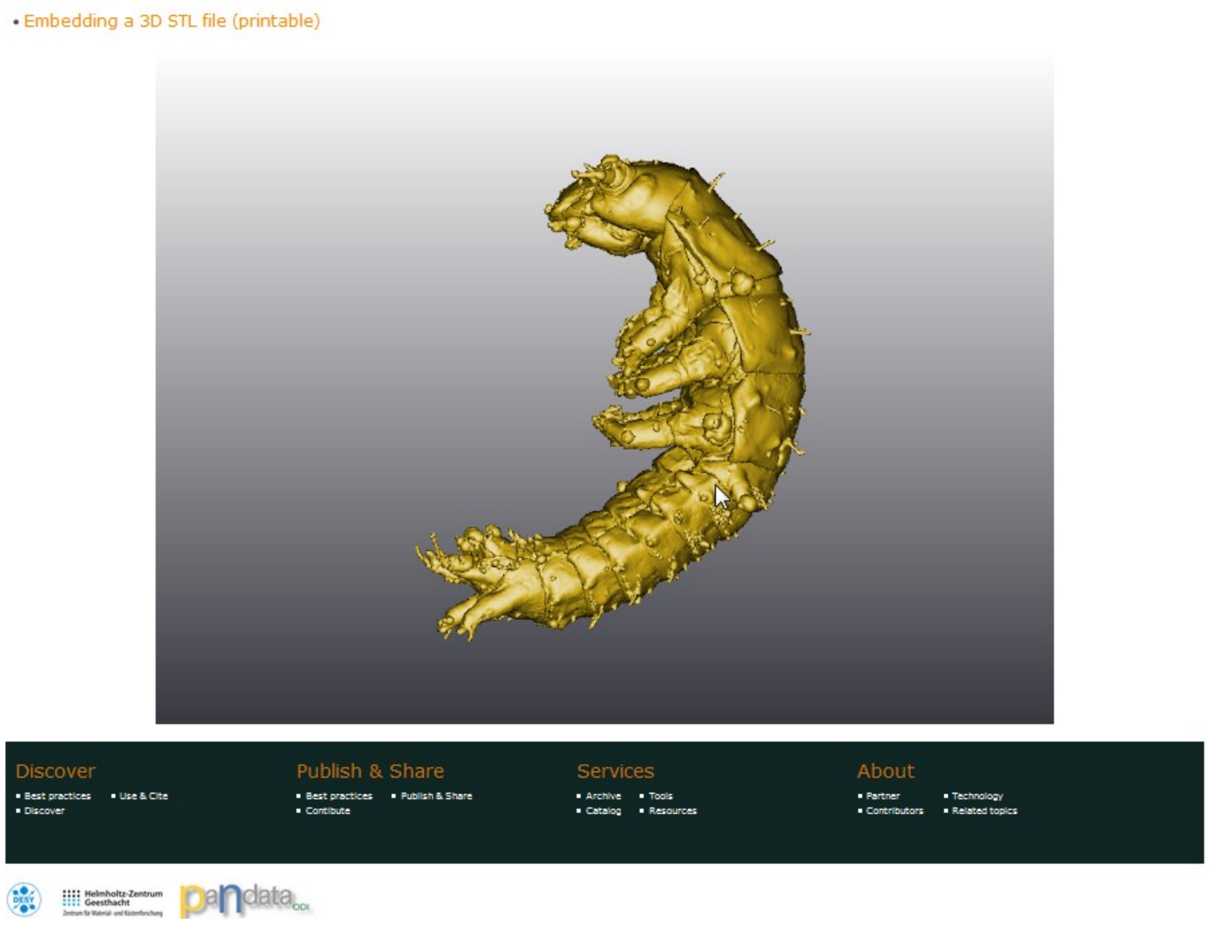

• Embedding a 3D STL file (printable)
Discover
• Best practices • Use & Cite
• Discover
Publish & Share
• Best practices • Publish & Share
• Contibute
Services
• Archive • Tools
• Catalog • Resources
About
• Partner • Technology
• Contributors • Related topics
Helmholtz-Zentrum Geesthacht
pandata ODI

Figure 20: Capture of an embedded (printable) 3D-model. Basic manipulation like rotate/translate/zoom/slab are supported.

[12] http://www.aps.anl.gov/tomopy/

5 NeXus Deployment

5.1 Introduction

Experimental data should be recorded in a way that the information necessary to understand the measurement is captured in the output files. This comprises the source parameters, the optical components of the beamline and all details of the sample environment. The biggest part of the data is generated by the readout of detectors, motion controllers and other electronics equipment. However, information describing the experimental setup and the precise experimental procedure is of equal importance. The output files should also contain details about the facility, the beamline, the beamtime, the scientists involved and information needed for provenance and preservation purposes. The PaN-data partners agreed to use NeXus/HDF5 as a common format.

This section concentrates on integration issues. The central part is the configuration of the NeXus files. This is a complex task especially for experimental stations with frequently changing setups. It is discussed how the configuration is prepared depending on the applied experimental technique and how the data acquisition process implements the NeXus configuration. Each facility stores metadata common to all experiments. These mandatory metadata are also discussed.

5.2 Status of the NeXus library

The foundation of all Nexus activities at DESY are two C++ libraries:

- *libpnicore* providing fixed size primitive data types and higher order data structures like multidimensional arrays
- *libpniio* implementing Nexus IO facilities

Both libraries are available from their Google Code project site[13].

5.2.1 Why implementing a new Nexus library?

There are two questions that may strike the reader at this place:

1. why developing a new Nexus library though the NIAC[14] already provides a C library[15] with bindings to many other programming languages?
2. and why doing it in C++ rather than in Java or Python?

The former one is easy to answer. The current Nexus C/C++ interface (NAPI) is outdated. It provides a very simple procedural interface with a file system like semantics with no higher level data structures. It would be rather painful to write more complex algorithms with this interface. In contrast to this the new interface comes with a whole set of features including

- All objects are first class objects which avoids accidental resource leaks (in particular when using exceptions).

[13] https://code.google.com/p/pni-libraries

[14] Nexus International Advisory Committee

[15] www.nexusformat.org

- Container types expose an STL compliant interface and thus can be used with the algorithms provided by the C++ standard library.
- The code is type safe in the sense that no pointers to void are used (for instance to read and write data).
- All objects are moveable, greatly enhancing the performance of the library.

The second question requires a bit more explanation. Over the last years Python and Java have gained ground in scientific computing. Python became nearly a lingua franca for interactive data analysis and projects like DAWN[16] have proven Java to be a suitable language for scientific applications. However, both languages share a common disadvantage: once code is written in either one of them, it can hardly be reused by software written in any other programming language (the DAWN project with its complex bridge between Java and CPython is a very good example for this issue). C++ provides all facilities a modern object oriented language should have and is thus superior over C when it comes to the design of large systems. However, it still produces native code and it is relatively easy to generate bindings from C++ to any other language including Java and Python. Besides its interoperability advantage, C++ has two additional benefits that should be mentioned here

1. the performance of C++ is superior in comparison with Java or Python (in particular when latency becomes important). The current development version of *libpniio* has a performance penalty of about 4% over plain HDF5 code while providing a full STL[17] compliant interface to Nexus data structures.
2. C++ requires relatively little infrastructure (currently the libraries depend only on the BOOST and HDF5 libraries). Therefore, C++ code should be easily portable to virtually any platform and/or architecture providing a sufficiently recent C++ compiler (C++11 support is required).

In addition to the C++ libraries a Python binding is provided to *libpniio*. At the current state the binding is rather low level. It is basically a one-to-one mapping of the C++ functionality to Python. The Python binding to *libpniio* is used in virtually all DESY infrastructure software related to Nexus.

5.2.2 Project status

At the time of writing these lines, the stable release for *libpnicore* and *libpniio* are 0.9.4 and 0.9.6 respectively. Both libraries are under constant development and are currently heading towards their 1.0.0 release. For this purpose both libraries experience a major code review which will

- further improve the performance
- reduce the amount of code that must be maintained
- complete the documentation
- simplify the interfaces (all container objects are now fully STL compliant)
- add a whole bunch of new algorithms making the life with Nexus easier.

The new algorithms include facilities for managing XML and Nexus, recursive searches for objects within the Nexus tree and improved performance for partial IO.

The libraries are currently in use by the following projects

[16] www.DAWNsci.org

[17] STL=Standard Template Library

Project Name	Description
PNI-Tools[18]	A set of command line tools to work with Nexus files (developed at DESY)
Fuse for Nexus[19]	A user space file system allowing to mount Nexus files (developed at KIT)
DESY Nexus tool-chain[20]	Tango servers and utility programs used for Nexus deployment at DESY (developed at DESY)

And the supported architectures and platforms are

Architecture	Platform
I386	Linux
X86_64	Linux
ARM-HF	Linux

5.2.3 Performance markers for *libpnicore*

libpnicore was mainly developed because C++ does not provide a reasonable multidimensional array type (like Fortran or the numpy arrays for Python). The library provides a compile-time configurable multidimensional array type via the mdarray template with the following features:

- Linear access via iterators to the data stored in the array. The order in which data items are accessed follows the C-order convention (last index varies fastest). The linear interface is fully STL compliant. A use could use code like this:

```
mdarray<…> array;

for(auto &element: array) element = …;
```

- Data access via a multidimensional index

```
mdarray<…> array;

for(i=0;i<nx;i++)
      for(j=0;j<ny;j++)
            array(i,j) = …;
```

- Implementation of basic arithmetic operators (+,-,/,*) are implemented as expression templates.

[18] https://code.google.com/p/pnitools

[19] https://code.google.com/p/fuse-for-nexus

[20] https://code.google.com/p/nexdatas

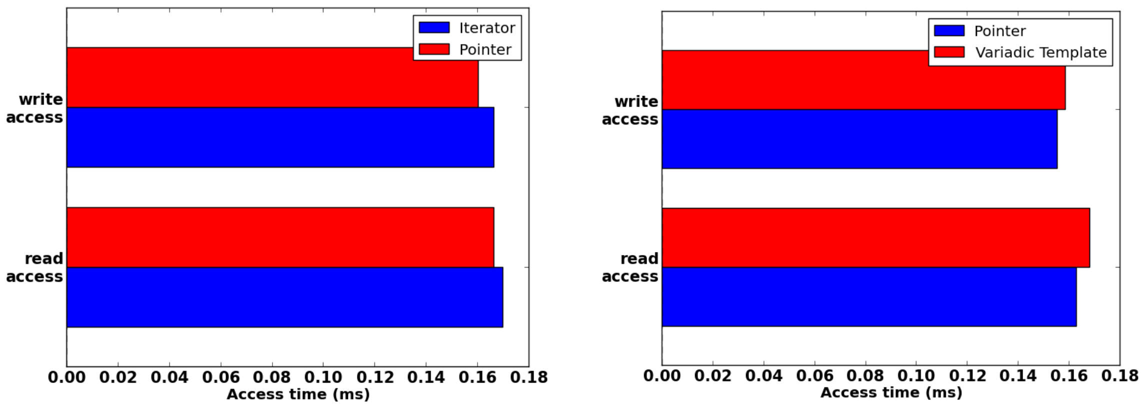
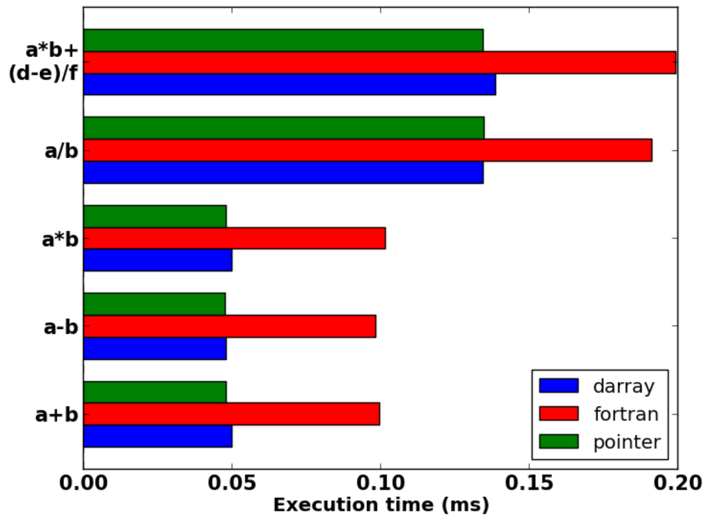


Figure 21: Data access to an instance of the mdarray template. The left panel shows the benchmark results for linear access via an iterator. On the right panel the benchmark results for data access via a multidimensional index is shown.

As shown in Figure 21 neither the interator- nor the index-interface introduce significant overhead in comparison to raw pointer access to the arrays elements. As shown in Figure 22 expression templates used to implement the arithmetic operators for the mdarray template show in fact the same performance as simple pointer loops. A surprising result was also that the resulting code even outperforms Fortran90. However, this could also be a problem with the used gfortran compil-er.



```
mdarray<…> a,b,c,d,e,f;
…
//expression template code
c = a*b+(d-e)/f;
```

```
float64 *a,*b,*c,*d,*e,*f;
…
//simple pointer loop
for(size_t i=0;i<n;++i)
      c[i]=a[i]*b[i]
           +(d[i]-e[i])/f[i];
```

Figure 22: The left panel shows the execution time for expression templates compared with pointer loops and Fortran90 code. On the right panel the essential code sequences for expression templates and pointer loops are shown.

5.2.4 Performance markers for *libpniio*

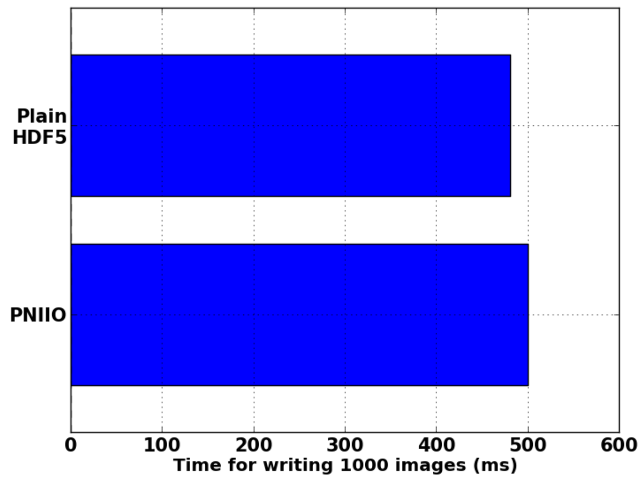


Figure 23: Write-performance of the libpniio interface vs. plain HDF5 code.

One of the design goals of *libpniio* was to keep its overhead over plain HDF5 small. The write benchmark between *libpniio* and plain HDF5 code shown in Figure 23 demonstrates that this design goal has been met. The libpniio code is only about 4% slower than the plain C HDF5 code. It should be mentioned that this performance penalty shows only when writing data to a RAM-disk as it was the case for the benchmark. When writing data to a common hard-drive the performance loss of *libpniio* over plain HDF5 virtually vanishes. It is important to note how drastically the number of lines of code is reduced by the *libpniio* Nexus interface as the following code snippet shows.

```
for(size_t n=0;n<nframes;++n)
{
      field.grow(0);
      field(n,slice(0,nx),slice(0,ny)).write(frame_data);
      file.flush();
}
```

This snippet is basically what the benchmark in Figure 23 measures.

5.3 External filters

With an increasing amount of data recorded during today's experiments data compression becomes a critical aspect. Using compression not only reduces the disk space consumed by a data file but also decreases the amount of data which has to be transferred via a network. Although HDF5 natively provides a couple of compression algorithms, custom algorithms may provide far better results not only in the compression rate which can be achieved but also in terms of performance. DECTRIS[21], a Swiss detector vendor, has taken this track and applies an LZ4 algorithm on the data produced by its new EIGER detectors. The algorithm is implemented in an FPGA and applied at real-time during data recording. To achieve this goal additional features had to be added to the HDF5 library by the HDF group. While DECTRIS funded the filter bypass functionality, re-

[21] https://www.dectris.com

quired to store pre-compressed data in an HDF5 dataset, DESY provided the money to implement a new external filter interface for HDF5 which will be described in this section.

The HDF5 library already provided a mechanism to load 3rd party compression algorithms. However, the original approach had one critical flaw: the new filter had to be registered with the library at runtime by the program which intended to use the custom code (see Figure 24).

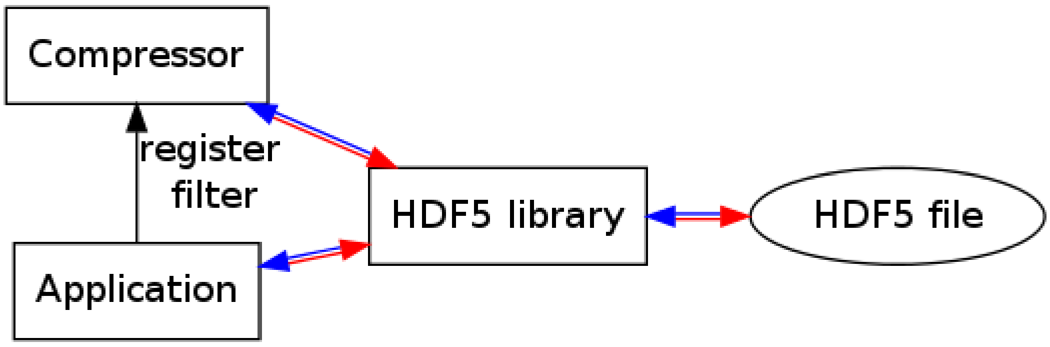


Figure 24: With the original external filter interface, the calling program had to register the custom filter with the HDF5 library.

When a new filter is added, the applications code needs to be modified and the program recompiled. Hence commercial closed source applications like Matlab[22] or IDL[23] could not use the external interface. As such a situation would be inacceptable for ours and DECTRIS's users a new solution was implemented. The new external interface does not require any modifications of the program using the HDF5 library. Instead, the library searches for custom filter modules in one or more configurable search paths and registers the available filters by itself as shown in Figure 25.

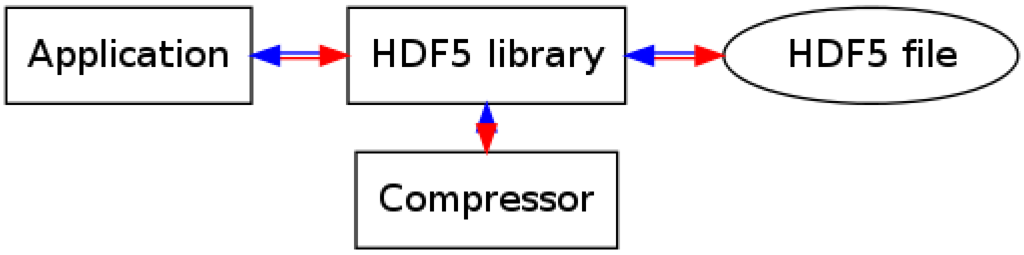


Figure 25: The new external filter interface of HDF5 requires no modification of the calling program. Instead it automatically configures external filters by itself.

5.3.1 First implementation of an external filters infrastructure

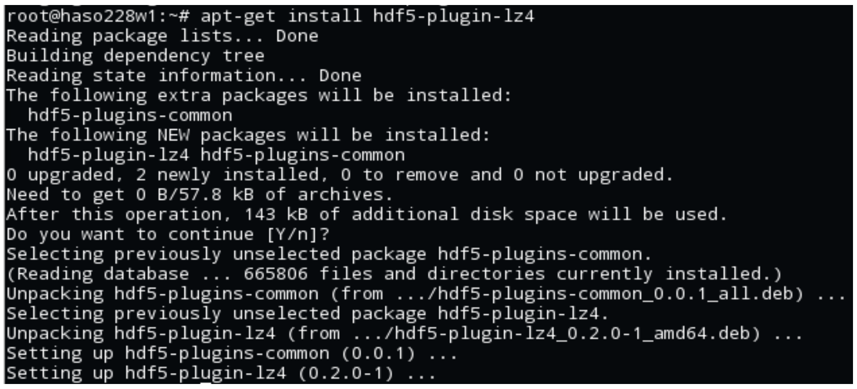
Such an approach clearly needs some support from system integrators. At DESY, this infrastructure has been implemented for Debian GNU/Linux. Two packages are provided

- *hdf5-plugins-common* including components required by all external filter plug-ins
- *hdf5-plugin-lz4* providing the LZ4 compression algorithm for DECTRISs EIGER detector.

[22] http://www.mathworks.com

[23] http://www.exelisvis.com/ProductsServices/IDL.aspx

As shown in Figure 26 the *hdf5-plugins-common* package does not require explicit installation, it is rather pulled in as a dependency for the *hdf5-plugin-lz4* package. Users, thus, only need to know which package to install for a particular filter – required packages are installed automatically.

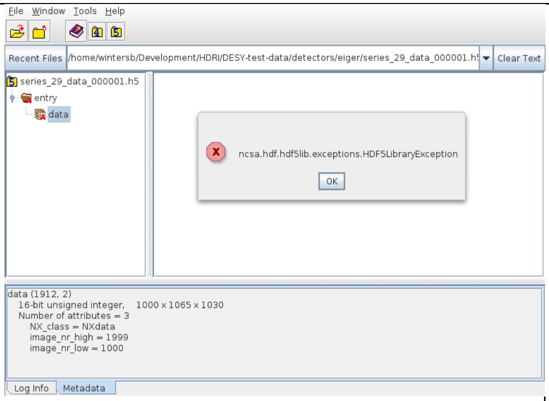


```
root@haso228w1:~# apt-get install hdf5-plugin-lz4
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  hdf5-plugins-common
The following NEW packages will be installed:
  hdf5-plugin-lz4 hdf5-plugins-common
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/57.8 kB of archives.
After this operation, 143 kB of additional disk space will be used.
Do you want to continue [Y/n]?
Selecting previously unselected package hdf5-plugins-common.
(Reading database ... 665806 files and directories currently installed.)
Unpacking hdf5-plugins-common (from .../hdf5-plugins-common_0.0.1_all.deb) ...
Selecting previously unselected package hdf5-plugin-lz4.
Unpacking hdf5-plugin-lz4 (from .../hdf5-plugin-lz4_0.2.0-1_amd64.deb) ...
Setting up hdf5-plugins-common (0.0.1) ...
Setting up hdf5-plugin-lz4 (0.2.0-1) ...
```

Figure 26: Installation of the infrastructure packages for HDF5s external filter interface on Debian GNU/Linux.

5.3.2 External filters and user applications

We tested the functionality of the external filter setup with several applications, including IDL as representative of commercial applications.

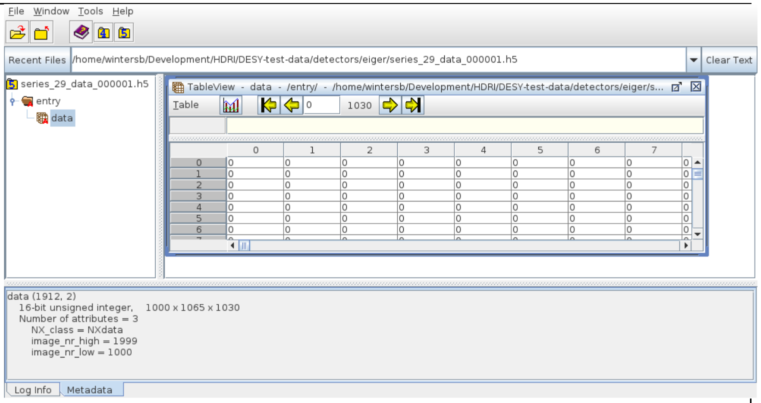


/home/wintersb/Development/HDRI/DESY-test-data/detectors/eiger/series_29_data_000001.h5

ncsa.hdf.hdf5lib.exceptions.HDF5LibraryException

```
data (1912, 2)
   16-bit unsigned integer,    1000 x 1065 x 1030
   Number of attributes = 3
      NX_class = NXdata
      image_nr_high = 1999
      image_nr_low = 1000
```

Figure 27: HDFView throws an exception when trying to access EIGER data without the filter package installed.



/home/wintersb/Development/HDRI/DESY-test-data/detectors/eiger/series_29_data_000001.h5

TableView - data - /entry/ - /home/wintersb/Development/HDRI/DESY-test-data/detectors/eiger/s...

0 1030

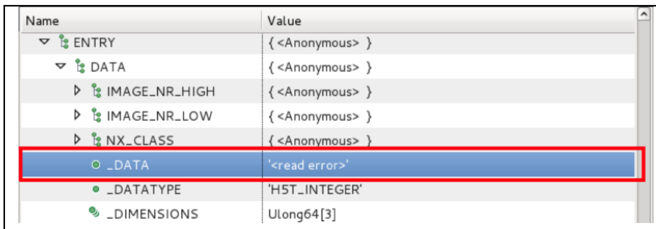
```
     0    1    2    3    4    5    6    7
0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0
```

```
data (1912, 2)
   16-bit unsigned integer,    1000 x 1065 x 1030
   Number of attributes = 3
      NX_class = NXdata
      image_nr_high = 1999
      image_nr_low = 1000
```

Figure 28: Once the LZ4 filter package is installed HDFView can access EIGER data like it would have been compressed with a native HDF5 filter.

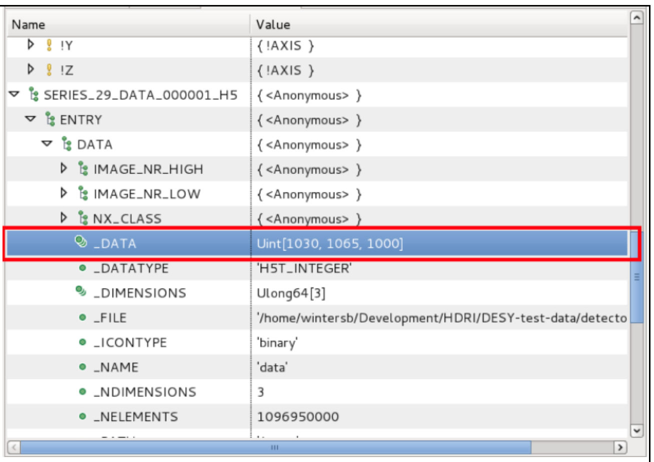
HDFView is a very common application distributed by the HDF group. This Java program is typically used to have a quick look into an HDF5 file. Although its capabilities are limited from the point of data visualization and analysis it quickly unravels the structure of a file. In Figure 27 we tried to access data stored by an EIGER detector with HDFView without having the appropriate filter plug-in installed. Consequently the application throws an exception. Once the LZ4 package is installed, HDFView can access the data like it would have been compressed using one of HDF5s native algorithms (see Figure 28).

As it is shown in the next example our setup does not only work for open source applications. IDL is a quite popular commercial software package for data analysis (in particular for image data). It provides native support for HDF5 (a general requirement that every software which should read the data has to satisfy). When we try to read EIGER data without having the LZ4 package installed IDL responds with a "read error" message as shown in Figure 29. Unlike HDFView, IDL cannot even read the metadata of the dataset without the plug-in. However, once the plug-in package is copied to the system IDL is happily accessing EIGER data as shown in Figure 30.



Name	Value
ENTRY	{ <Anonymous> }
DATA	{ <Anonymous> }
IMAGE_NR_HIGH	{ <Anonymous> }
IMAGE_NR_LOW	{ <Anonymous> }
NX_CLASS	{ <Anonymous> }
_DATA	'<read error>'
_DATATYPE	'H5T_INTEGER'
_DIMENSIONS	Ulong64[3]

Figure 29: IDL complains with a 'read error' message that it cannot access the data stored in the HDF5 dataset.



Name	Value
!Y	{ !AXIS }
!Z	{ !AXIS }
SERIES_29_DATA_000001_H5	{ <Anonymous> }
ENTRY	{ <Anonymous> }
DATA	{ <Anonymous> }
IMAGE_NR_HIGH	{ <Anonymous> }
IMAGE_NR_LOW	{ <Anonymous> }
NX_CLASS	{ <Anonymous> }
_DATA	Uint[1030, 1065, 1000]
_DATATYPE	'H5T_INTEGER'
_DIMENSIONS	Ulong64[3]
_FILE	'/home/wintersb/Development/HDRI/DESY-test-data/detecto
_ICONTYPE	'binary'
_NAME	'data'
_NDIMENSIONS	3
_NELEMENTS	1096950000

Figure 30: After installation of the LZ4 filter package IDL can access the EIGER data.

5.3.3 External filters with Python

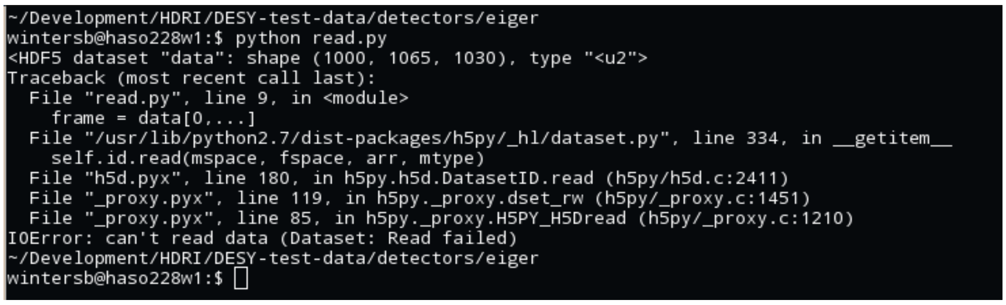
As we have already seen in the previous section the HDF5 external filter plug-in is entirely transparent to the user. Thus, it can be used with virtually every software linked against a sufficiently recent version of HDF5 (>1.8.11). In this last use case a Python script should access EIGER data using the h5py Python binding to the HDF5 library.

Consider for this case the following script code

```
#!/usr/bin/env python
import h5py

f = h5py.File("series_29_data_000001.h5","r")
data = f["/entry/data"]
print data
frame = data[0,...]
```

Without the LZ4 plug-in installed the script will throw an exception as shown in Figure 31.



```
~/Development/HDRI/DESY-test-data/detectors/eiger
wintersb@haso228w1:$ python read.py
<HDF5 dataset "data": shape (1000, 1065, 1030), type "<u2">
Traceback (most recent call last):
  File "read.py", line 9, in <module>
    frame = data[0,...]
  File "/usr/lib/python2.7/dist-packages/h5py/_hl/dataset.py", line 334, in __getitem__
    self.id.read(mspace, fspace, arr, mtype)
  File "h5d.pyx", line 180, in h5py.h5d.DatasetID.read (h5py/h5d.c:2411)
  File "_proxy.pyx", line 119, in h5py._proxy.dset_rw (h5py/_proxy.c:1451)
  File "_proxy.pyx", line 85, in h5py._proxy.H5PY_H5Dread (h5py/_proxy.c:1210)
IOError: can't read data (Dataset: Read failed)
~/Development/HDRI/DESY-test-data/detectors/eiger
wintersb@haso228w1:$ 
```

Figure 31: The Python script will throw an exception when attempting to access EIGER data without the LZ4 plug-in installed.

After the installation of the plug-in the script runs without errors. From these examples one can also learn that many bindings to HDF5, in particular the Python bindings, provide additional compression filters. Until now, data stored in files produced by those bindings was inaccessible by commercial applications using binding-specific filters. The new external filter interface provides a simple means to make this data available to those applications. Even open source applications can benefit from the new filter interface. It reduces the efforts for package maintainers as a package does not need to be recompiled when a new filter becomes available. Once a package for the filter plug-in is available the data can be accessed without modifications of the original source code.

5.4 NeXus Integration at DESY

5.4.1 Overview

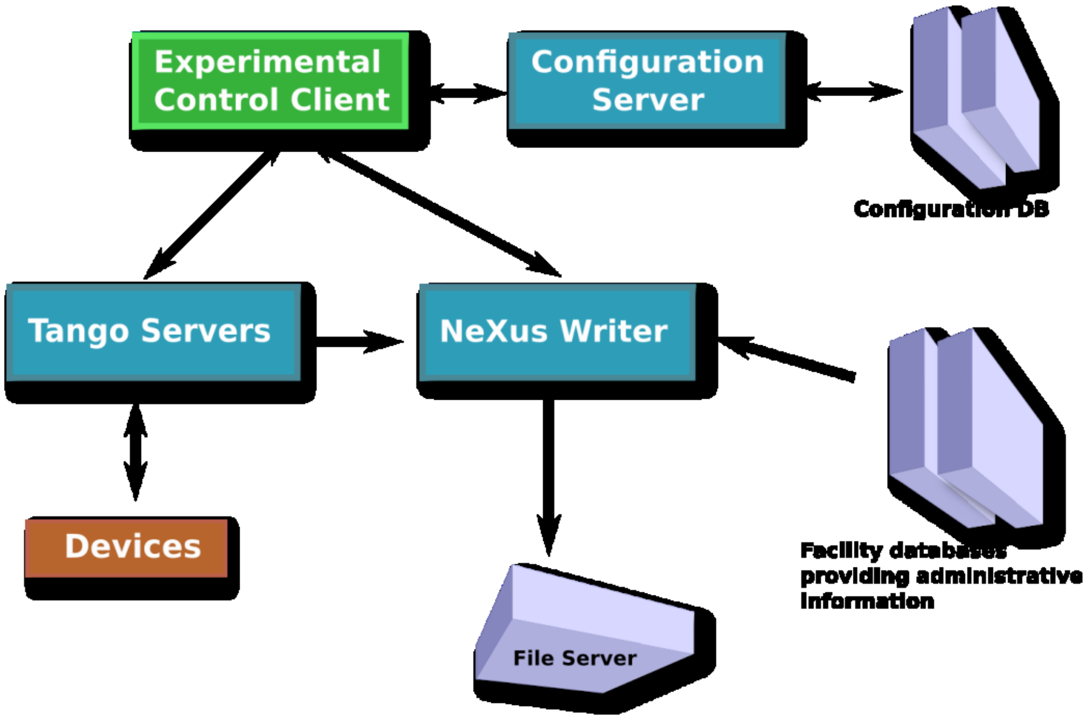

Experimental Control Client
Configuration Server
Configuration DB
Tango Servers
NeXus Writer
Devices
File Server
Facility databases providing administrative information

Figure 32: Data acquisition framework at DESY

The data acquisition framework consists of several applications communicating through standardized interfaces, Figure 32. The following list briefly explains these applications in order to introduce notions mentioned below in this section.

- The experiment control client (ECC) is an application used by researches to conduct measurements. It has graphical and command line interfaces. Scripts may also act as ECCs.
- Tango[24] is the communication layer for all PETRA III experiments. Therefore all devices are exported via Tango servers. Their interface consists of properties, commands and attributes. Attributes are of special interest because they provide data being read by the NeXus Writer, e.g. motor positions, counter readings or MCAs.
- The NeXus Writer is a Tango Server which opens NeXus files, creates the internal structure and fills it with data coming from the ECC, other Tango servers, database queries or Python scripts. The NeXus Writer is controlled by the ECC.

[24] www.tango-controls.org

- The Configuration Server is a Tango server managing the information necessary to build the structure of the NeXus files. It has access to a configuration database.
- The facility database provides administrative information stored in the NeXus files.

5.4.2 The configuration procedure

The NeXus configuration is a composition of components and data sources. Components are objects constructed of NeXus base classes representing simple devices (e.g. counter, MCA), composite devices (e.g. molecular beam epitaxy chambers mounted in a beam path) or structures only holding metadata (e.g. mandatory metadata). The available data sources are listed below. The concept is inspired by these correspondences linking scientific notions to technical terms:

- measurement ↔ file,
- experimental setup ↔ NeXus configuration,
- device ↔ component,
- device attribute (e.g. counter reading, motor position) ↔ data source

The central point is the relation between devices and components. Devices are selected or de-selected depending on the experimental technique applied. Likewise components are inserted into a configuration or deleted from it. They consist of the following items:

- Name: a string identifying the component.
- Data sources: specify where the data to be stored in NeXus fields have to be fetched from. The available data source types are:
    - Tango attributes,
    - database queries,
    - data supplied by the experiment control client (ECC) and
    - output of python scripts.
- Strategies: indicate at which point of the measurement the data have to be captured. Four strategy types are implemented: INIT, STEP, FINAL and POSTRUN. A NeXus field or attribute is marked with the POSTRUN tag, if the information is inserted by a subsequent data processing step.
- NeXus path: specifies the absolute position of the data inside the file. It is expressed in terms of NeXus groups.

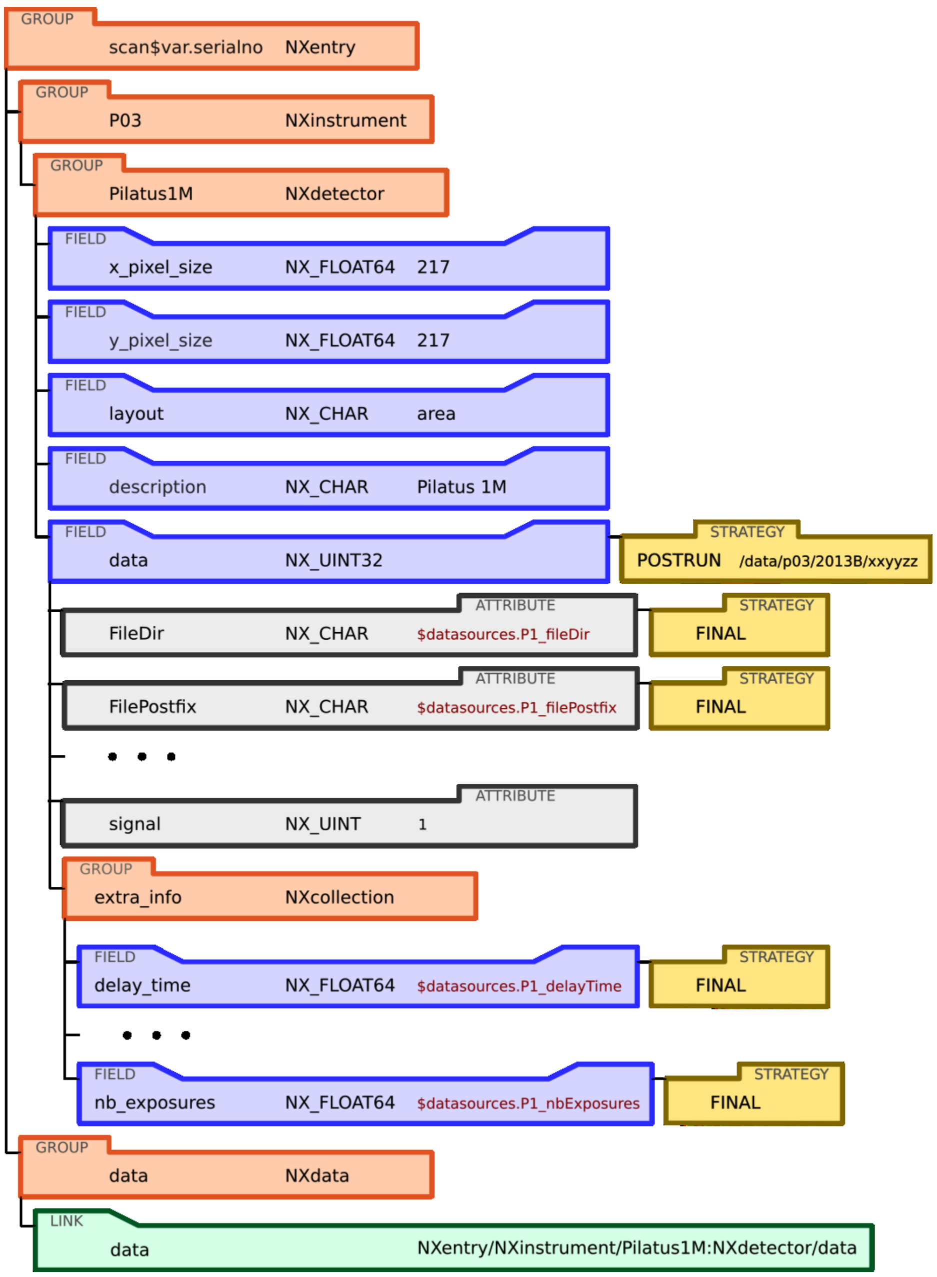

GROUP
scan$var.serialno NXentry
GROUP
P03 NXinstrument
GROUP
Pilatus1M NXdetector
FIELD
x_pixel_size NX_FLOAT64 217
FIELD
y_pixel_size NX_FLOAT64 217
FIELD
layout NX_CHAR area
FIELD
description NX_CHAR Pilatus 1M
FIELD
data NX_UINT32
STRATEGY
POSTRUN /data/p03/2013B/xxyyzz
ATTRIBUTE
FileDir NX_CHAR $datasources.P1_fileDir
STRATEGY
FINAL
ATTRIBUTE
FilePostfix NX_CHAR $datasources.P1_filePostfix
STRATEGY
FINAL
ATTRIBUTE
signal NX_UINT 1
GROUP
extra_info NXcollection
FIELD
delay_time NX_FLOAT64 $datasources.P1_delayTime
STRATEGY
FINAL
FIELD
nb_exposures NX_FLOAT64 $datasources.P1_nbExposures
STRATEGY
FINAL
GROUP
data NXdata
LINK
data NXentry/NXinstrument/Pilatus1M:NXdetector/data

Figure 33: Pilatus1M component

Figure 33 presents a concrete example of a component, a Pilatus1M detector. It is composed of groups (red), fields (blue), attributes (grey) and links (green). Groups contain groups, fields and links. They generate the hierarchical file structure. Groups have names, associated attributes and types like NXentry, NXinstrument, NXdata or NXdetector. Fields contain data with their attributes: name, data type, shape and unit. Attributes are descriptive information for groups and fields. Links refer to fields. They are generally used for the NXdata group identifying the most important data fields.

A device is identified by a hierarchy of NeXus groups. The pathname is a concatenation of the group names. If NeXus files contain several scans, they have multiple entries in the NXentry level. The NXinstrument group collects all devices belonging to an experiment. Therefore, NeXus files have only one NXinstrument entry per NXentry. There are beamlines with several experiment hutches. They are distinguished by different values of the name field of the NXinstrument group.

Examples for NeXus fields can be found in Figure 33. Most of them are already provided by the NXdetector class. Those fields not contained in the base class are stored in the NXcollection group. One of the NXdetector fields specifies the pixel size in x-direction. It has the NX_FLOAT64 type. Since this field contains static information, it has no data source assigned. In contrast, we see that the data field has a tag named FileDir, a string of the type NX_CHAR. The value of the string is a reference to the data source $datasources.P1M_fileDir. It is marked with the FINAL strategy tag. Therefore FileDir is resolved when the measurement ends.

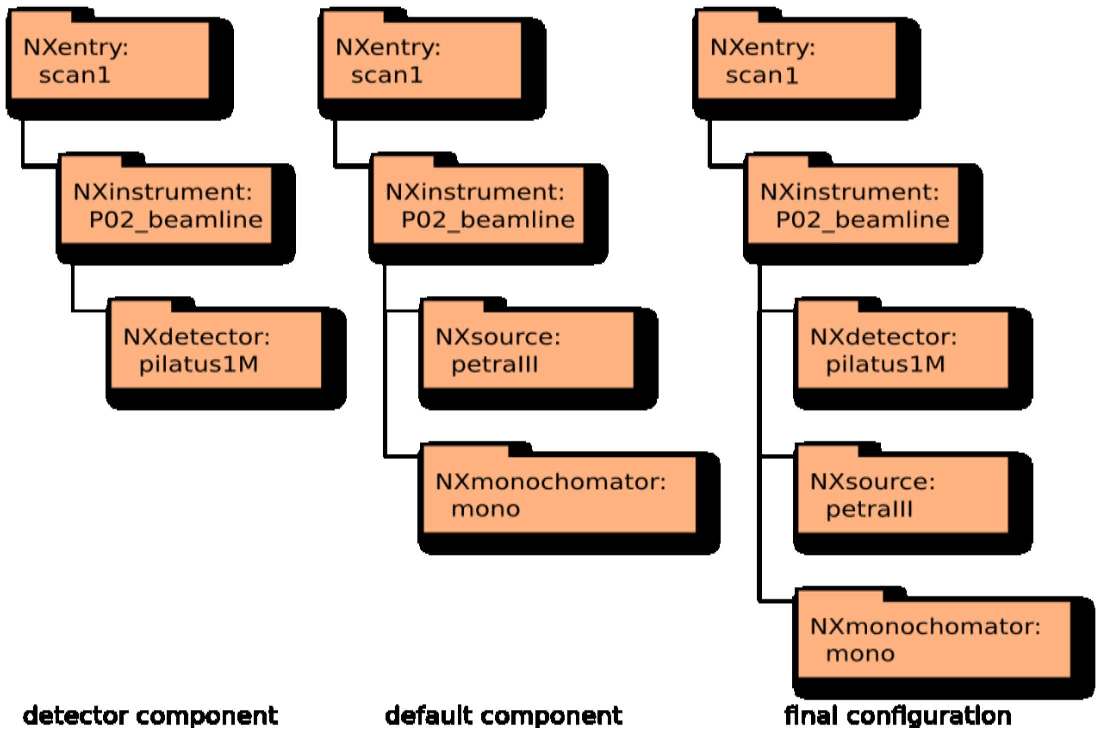
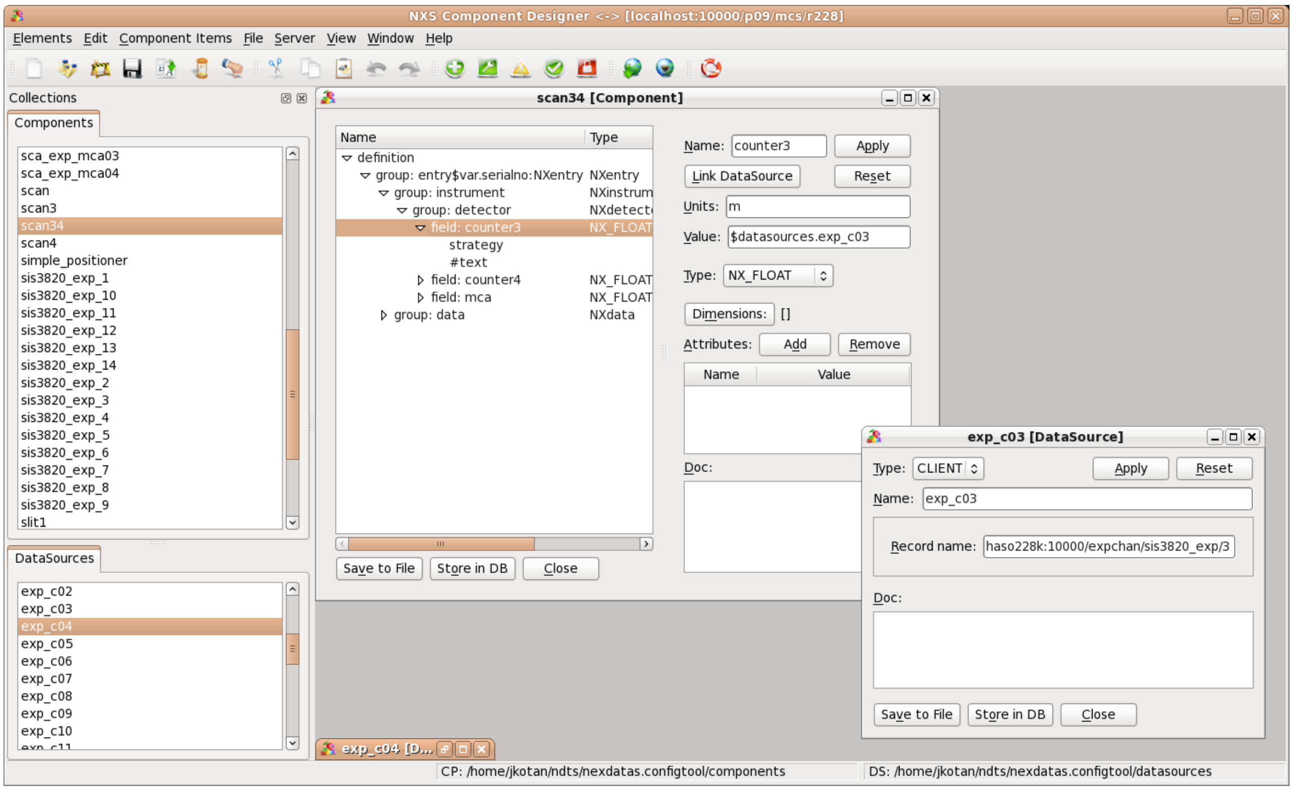


Figure 34: Merging components

Since components contain absolute path names, they can be merged to obtain the complete configuration. Figure 34 exemplifies the process with two components, a detector component representing a Pilatus camera and the default component containing the beamline description. The components are merged according to their group types and names. In practice, NeXus files typically consist of 20 to 40 components.

When components are created, the data source fields are initially occupied by placeholders. The process of assigning actual references to the data sources can be considered as an instantiation of a component. The aim is to create generic components that can be used at various places.



Components: sca_exp_mca03, sca_exp_mca04, scan, scan3, scan34, scan4, simple_positioner, sis3820_exp_1, sis3820_exp_10, sis3820_exp_11, sis3820_exp_12, sis3820_exp_13, sis3820_exp_14, sis3820_exp_2, sis3820_exp_3, sis3820_exp_4, sis3820_exp_5, sis3820_exp_6, sis3820_exp_7, sis3820_exp_8, sis3820_exp_9, slit1

DataSources: exp_c02, exp_c03, exp_c04, exp_c05, exp_c06, exp_c07, exp_c08, exp_c09, exp_c10

Name	Type
definition	
group: entry$var.serialno:NXentry	NXentry
group: instrument	NXinstrum
group: detector	NXdetecto
field: counter3	NX_FLOAT
strategy	
#text	
field: counter4	NX_FLOAT
field: mca	NX_FLOAT
group: data	NXdata

Name: counter3
Units: m
Value: $datasources.exp_c03
Type: NX_FLOAT
Dimensions: []

Type: CLIENT
Name: exp_c03
Record name: haso228k:10000/expchan/sis3820_exp/3

CP: /home/jkotan/ndts/nexdatas.configtool/components DS: /home/jkotan/ndts/nexdatas.configtool/datasources

Figure 35: Component designer

Since the preparation of components and data sources is a rather tedious task, the Component Designer (Figure 35), has been developed. It creates new components and data sources, edits existing ones or links data sources to components. The Component Designer (CD) uses a data base to store components and data sources.

The following procedure outlines the steps to create all required components for a particular experiment:

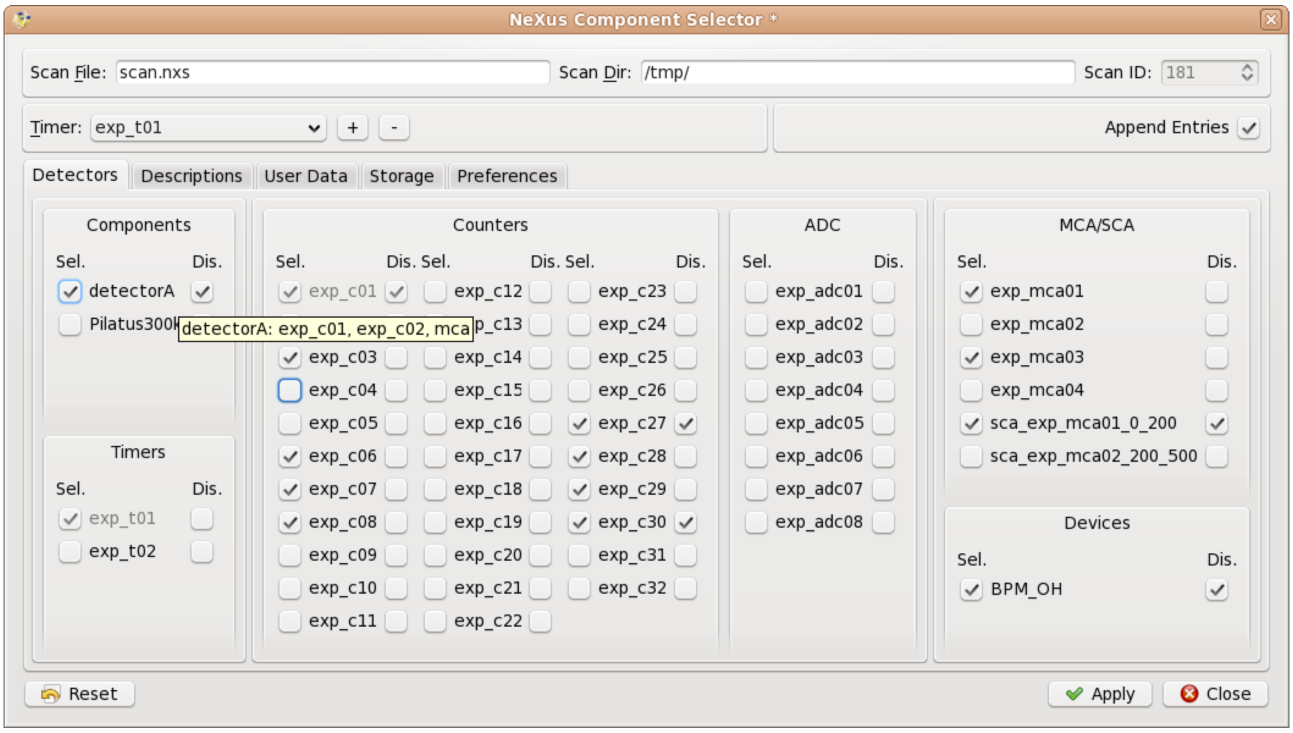
1. Creation of the data sources: data sources being related to Tango[25] attributes can be generated by automated procedures. The others have to be created with the CD.
2. Re-use of existing components: if a device is already in use at another beamline the corresponding component can be imported and the local data sources have to be supplied.
3. Creation of new components: if a device is used for the first time, the component has to be created with the CD and the data sources have to be supplied.
4. Storage of the newly created components and data sources.

Creating data sources and components requires a reasonable amount of expertise. Hence this task it typically performed by beamline staff or members of the experiment control group.

5.4.3 Component selection

So far, technical details concerning the NeXus configuration have been explained. This section is focussed on the integration into the data acquisition system. Every measurement requires components to be selected. The Component Selector (CS) is a graphical user interface serving this purpose. Figure 36 displays the Detectors tab of this tool including the available components and data sources.

[25] www.tango-controls.org


NeXus Component Selector *

Scan File: scan.nxs Scan Dir: /tmp/ Scan ID: 181

Timer: exp_t01 + - Append Entries ✓

Detectors | Descriptions | User Data | Storage | Preferences

Components

Sel.		Dis.
✓	detectorA	✓
	Pilatus300k	

detectorA: exp_c01, exp_c02, mca

Timers

Sel.		Dis.
✓	exp_t01	
	exp_t02	

Counters

Sel.		Dis.	Sel.		Dis.	Sel.		Dis.
✓	exp_c01	✓		exp_c12			exp_c23	
				exp_c13			exp_c24	
✓	exp_c03			exp_c14			exp_c25	
	exp_c04			exp_c15			exp_c26	
	exp_c05			exp_c16		✓	exp_c27	✓
✓	exp_c06			exp_c17		✓	exp_c28	
✓	exp_c07			exp_c18		✓	exp_c29	
✓	exp_c08			exp_c19		✓	exp_c30	✓
	exp_c09			exp_c20			exp_c31	
	exp_c10			exp_c21			exp_c32	
	exp_c11			exp_c22				

ADC

Sel.		Dis.
	exp_adc01	
	exp_adc02	
	exp_adc03	
	exp_adc04	
	exp_adc05	
	exp_adc06	
	exp_adc07	
	exp_adc08	

MCA/SCA

Sel.		Dis.
✓	exp_mca01	
	exp_mca02	
✓	exp_mca03	
	exp_mca04	
✓	sca_exp_mca01_0_200	✓
	sca_exp_mca02_200_500	

Devices

Sel.		Dis.
✓	BPM_OH	✓

Reset Apply Close

Figure 36: Component selection: detectors

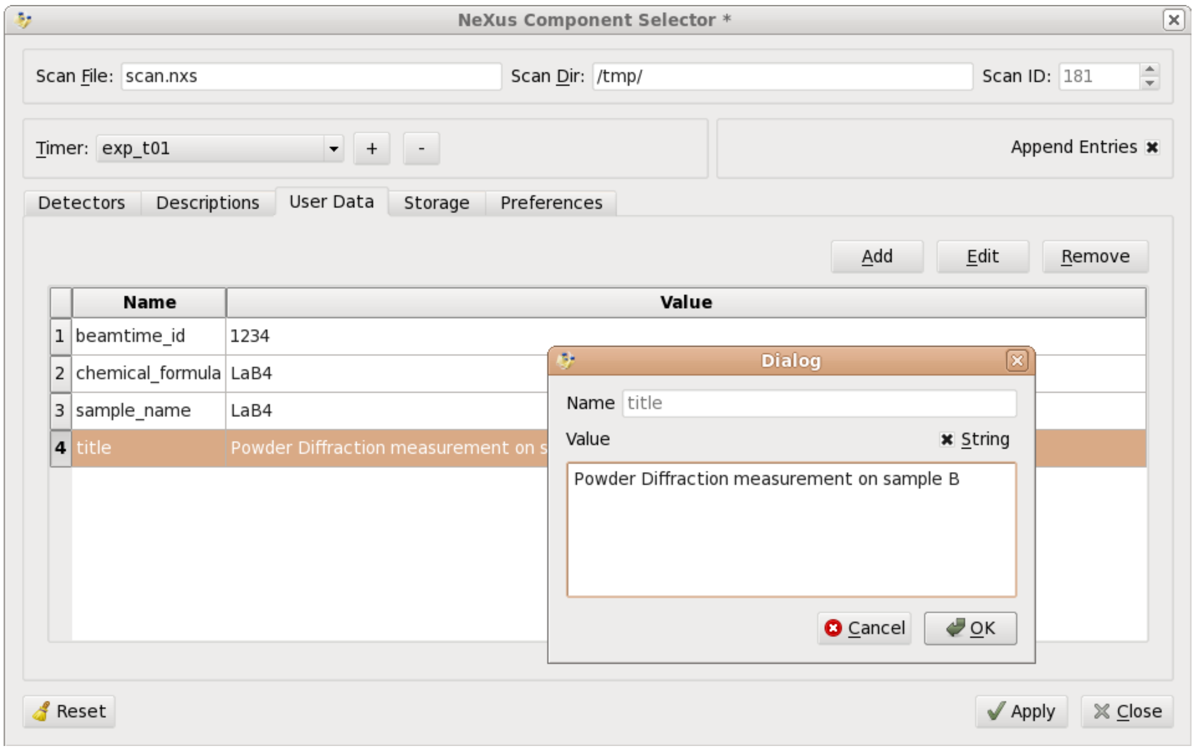
Composite components with multiple data sources appear in the Components frame. Simple components are distributed in the frames labelled Counters, ADC, MCA, etc. Components are selected by activating the Sel. checkbox. If the Dis. checkbox is enabled, the output of the device is displayed during the scan for monitoring purposes.

Simple components may be part of composite components. As a consequence, selecting a composite component may implicitly select one or more simple components. This dependency has to be visible for the users. Therefore, simple components being implicitly selected are deactivated and their font colour changes to grey. The user may also move the mouse over a composite component to inspect the contents.

Ideally all devices are contained in components ensuring that they have sensible NeXus paths and meaningful metadata associated with them. In practice this is not always possible. Consider a counter module with 32 channels. Some of them are permanently connected to specific detectors. It is an easy task to create components for these inputs. However, during the course of a beamtime, it may happen that a researcher needs to record some other signal. Depending on the circumstances it may be impossible to create a new component immediately but the new signal has to be recorded. In order to handle this situation, dynamical components have been introduced. They are automatically created whenever a selected device is not covered by a component.

After devices have been selected the state of the Detectors tab can be saved in a profile. It is possible to create several profiles. Profiles can be loaded to restore a particular device selection. This way, the researcher can easily switch from one data acquisition setup to another.

The Descriptions of the CS displays components containing metadata only. They are divided into two groups: the beamline-specific components and the discipline-specific components. The beamline group describes the source device and the facility. The Discipline group contains information about the spatial arrangement of the experimental setup, mainly motor positions.



Scan File: scan.nxs
Scan Dir: /tmp/
Scan ID: 181
Timer: exp_t01
Append Entries ✖

	Name	Value
1	beamtime_id	1234
2	chemical_formula	LaB4
3	sample_name	LaB4
4	title	Powder Diffraction measurement on s

Name: title
Value ✖ String
Powder Diffraction measurement on sample B

Figure 37: Component selector: NeXus user data

It has already been mentioned that ECC can serve as a data source. The data are transferred as json-encoded strings consisting of keyword/value pairs. The values are mostly generated by automated procedures. However, some of the client data have to be provided by the user. Figure 37 shows the CS tab allowing the researcher to supply this information. Typical examples for user-supplied metadata are title, sample name and user comment.

In order to allow for the component selection to be carried out from the command line, several Python scripts have been developed serving this purpose.

5.4.4 Mandatory metadata

Certain metadata contained in NeXus files are common to all DESY experiments, the mandatory metadata (MMD). The size of the MMD is small. But they are sufficient for a rough identification of the file contents. Besides, MMD are used by the ingestion procedure inserting files into the metadata catalogue.

NeXus field or attribute	ICAT field	Remark
/NXentry/experiment_identifier	Investigation	a unique beamtime IT
/NXentry/title	Investigation.Summary	description of the experiment
/NXentry/NXinstrument/name	Instrument.Fullname	name of the beamline
/NXentry/NXinstrument/name@short_name	Instrument.Name	short name of the beamline
/NXentry/NXinstrument/NXsource/name	Facility.Fullname	accelerator name
/NXentry/NXinstrument/NXsource/name@short_name	Facility.Name	short name for the accelerator
/NXentry/NXsample/name	Sample.Name	name of the sample
/NXentry/NXsample/chemical_formula	Sampletype.Molecularformula	the chemical formula of the sample
/NXentry/start_time	Instrument.Start_Date	start time of the scan

/NXentry/end_time	Instrument.End_Date	end time of the scan

Table 5: Mandatory metadata at DESY

Table 5 contains a list of the DESY MMD and it shows how the NeXus items are mapped to ICAT fields.

- The /NXentry/experiment_identifier identifies the beamtime. Several NeXus files can have the same beamtimeId.
- /NXentry/title is supplied by the user. It is scan-specific.
- /NXentry/NXinstrument/name is the full beamline name, e.g.: HighRes Diffraction
- /NXentry/NXinstrument/name@short_name the name of the beamline, e.g. P08
- /NXentry/NXsample/name is the sample name, it is scan specific.
- /NXentry/NXsample/chemical_formula the chemical formula, it is scan specific, e.g. LaB4
- NXentry/start_time, /NXentry/end_time start and end of a scan, expressed in ISO format, e.g. 2013-06-10T14:30:16.238875+0200.

5.4.5 The data acquisition process

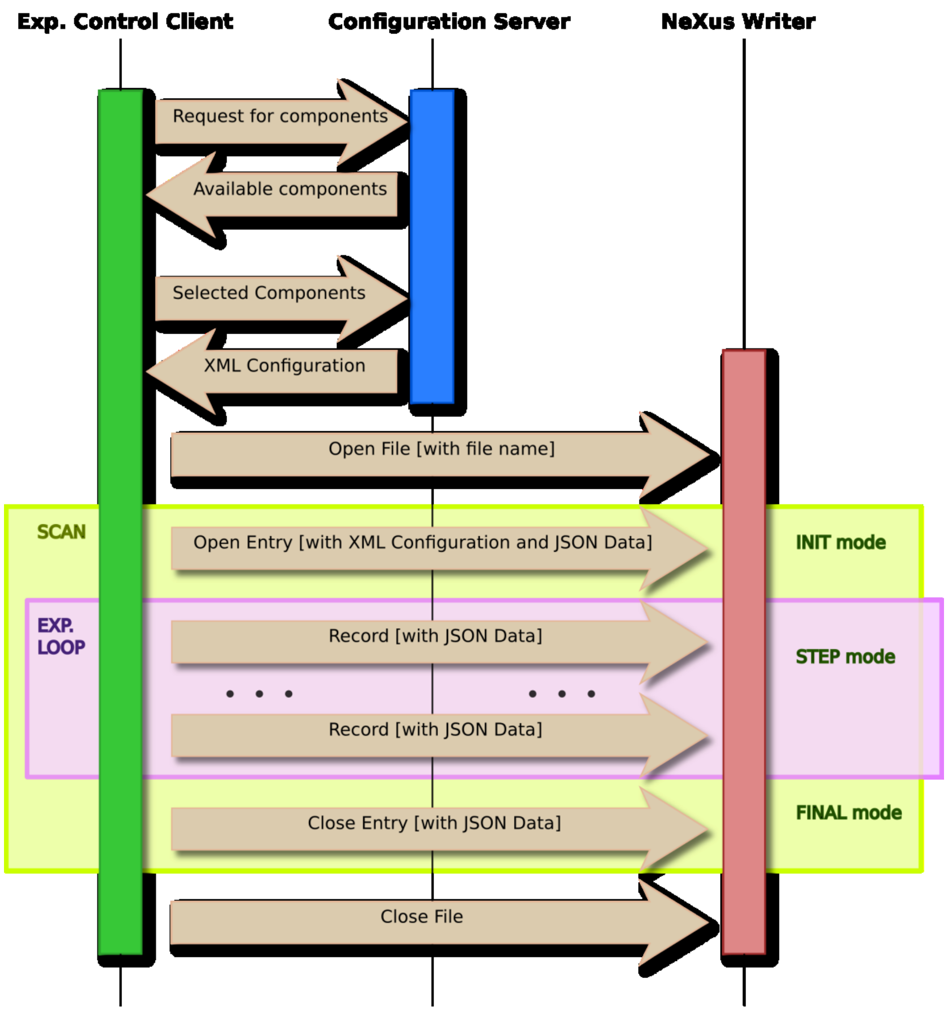


Figure 38: DESY data acquisition process

This section explains how the NeXus file production is integrated into the data acquisition system at DESY. Figure 38 displays a sequence diagram of the process. The ECC prompts the Configuration Server for a list of available components. Depending on the experimental technique the researcher selects components. The ECC sends the selected components to the Configuration Server where they are merged into the configuration, an XML string. The ECC reads this string and passes it to the NeXus Writer. The NeXus Writer parses the configuration and creates a NeXus file with the required hierarchical structure. For each data source an object is created carrying the information about where to fetch the data from and where to store them. These data objects are maintained in lists. There is one list for INIT, STEP and FINAL strategies.

After the NeXus file has been prepared and the lists of data objects have been produced, the measurement starts. At this stage the NeXus writer appends an NXentry group as a container for all data being captured during a scan and it iterates through the list of INIT data objects invoking their function members to transfer the data from the sources to the file. Thereafter the experimental loop is executed involving the execution of the STEP list objects. A scan ends with the FINAL phase closing the NXentry group. INIT and FINAL data are read once. STEP data are read for each step of a scanning procedure.

Data collected during the INIT phase typically consist of information describing the instrument and the facility. User metadata and the initial state of the beamline devices, motor positions, are also included. The STEP phase generates most of the raw data. An example for FINAL data is the end time of the scan.

In addition there is the POSTRUN strategy for data to be supplied in a post processing step. This feature has been introduced for 2D detectors creating data which, for performance reasons, are not immediately inserted into the NeXus files. Camera servers store their image frames temporarily on ramdisks, local disks or on network file systems. After a measurement is finished the data collection process opens the NeXus file, searches for POSTRUN tags and inserts the images into the file. This way, the NeXus file contains all experimental data belonging to a measurement.

5.4.6 Generalizing the DESY NeXus implementation

At DESY, the NeXus file production is based on Tango, the control system for the PETRA III experiment. The consequence is that the NeXus Writer and the Configuration server are Tango servers and data sources may refer to Tango server attributes. Since this dependency limits the utilization of the described framework to Tango institutes, Python modules have been developed that make the functionality generally available. It is possible to run the NeXus Writer and the Configuration Server independent of Tango.

5.5 NeXus Integration at ALBA

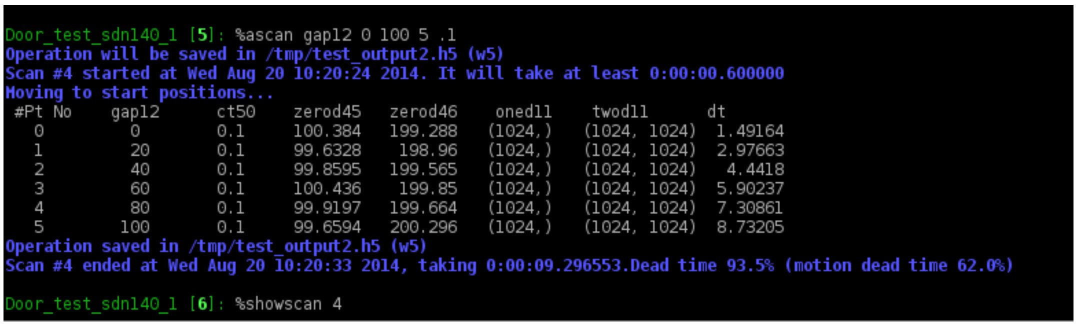
5.5.1 Overview

At ALBA data acquisition is performed within the Sardana[26] framework. This program package establishes an abstraction layer for hardware access, the Device Pool. Pool controllers expose standardized interfaces for device classes such as Moveables or Experimental Channels. In practice there is a hierarchy of specializations of these interfaces. Sardana has a mechanism for synchronizing actions on Pool objects, e.g. for movements. Measurements are executed by means of macros. These are Python scripts implementing a special interface. The current NeXus integration at ALBA is done in the context of macros performing scans.

Users typically generate data by running scan macros from the command line interface Spock or

[26] http://sardana-controls.org

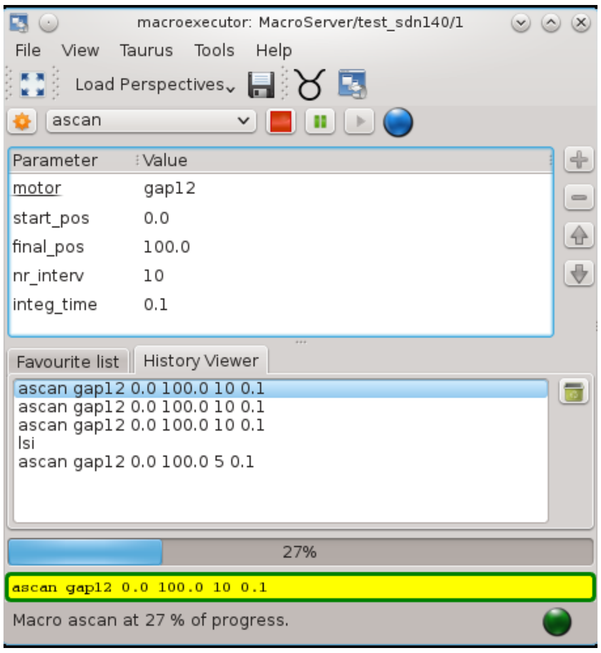
from GUIs built with Taurus[27] widgets, e.g. the Macroexecutor (Figure 40), Macrosequencer, Macrobutton. Figure 39 is an example for a scan started in a Spock session.



```
Door_test_sdn140_1 [5]: %ascan gap12 0 100 5 .1
Operation will be saved in /tmp/test_output2.h5 (w5)
Scan #4 started at Wed Aug 20 10:20:24 2014. It will take at least 0:00:00.600000
Moving to start positions...
 #Pt No    gap12      ct50    zerod45   zerod46    oned11     twod11       dt
   0         0         0.1    100.384   199.288   (1024,)   (1024, 1024)  1.49164
   1         20        0.1    99.6328    198.96   (1024,)   (1024, 1024)  2.97663
   2         40        0.1    99.8595   199.565   (1024,)   (1024, 1024)   4.4418
   3         60        0.1    100.436    199.85   (1024,)   (1024, 1024)  5.90237
   4         80        0.1    99.9197   199.664   (1024,)   (1024, 1024)  7.30861
   5        100        0.1    99.6594   200.296   (1024,)   (1024, 1024)  8.73205
Operation saved in /tmp/test_output2.h5 (w5)
Scan #4 ended at Wed Aug 20 10:20:33 2014, taking 0:00:09.296553.Dead time 93.5% (motion dead time 62.0%)

Door_test_sdn140_1 [6]: %showscan 4
```

Figure 39: Sardana CLI Spock


macroexecutor: MacroServer/test_sdn140/1
File View Taurus Tools Help
Load Perspectives
ascan
Favourite list
History Viewer
27%

Parameter	Value
motor	gap12
start_pos	0.0
final_pos	100.0
nr_interv	10
integ_time	0.1

```
ascan gap12 0.0 100.0 10 0.1
ascan gap12 0.0 100.0 10 0.1
ascan gap12 0.0 100.0 10 0.1
lsi
ascan gap12 0.0 100.0 5 0.1
```

```
ascan gap12 0.0 100.0 10 0.1
```

Macro ascan at 27 % of progress.

Figure 40: Sardana GUI Macroexecutor

The scan macros produce data which are passed, together with metadata from other sources, to agents called recorders for data storage, display or processing. For example, the OutputRecorder displays the measured data in a terminal window and the JSONRecorder exports data as json-encoded strings via Tango attributes. NeXus files are created by the NeXusRecorders, Figure 41.

[27] http://taurus-scada.org

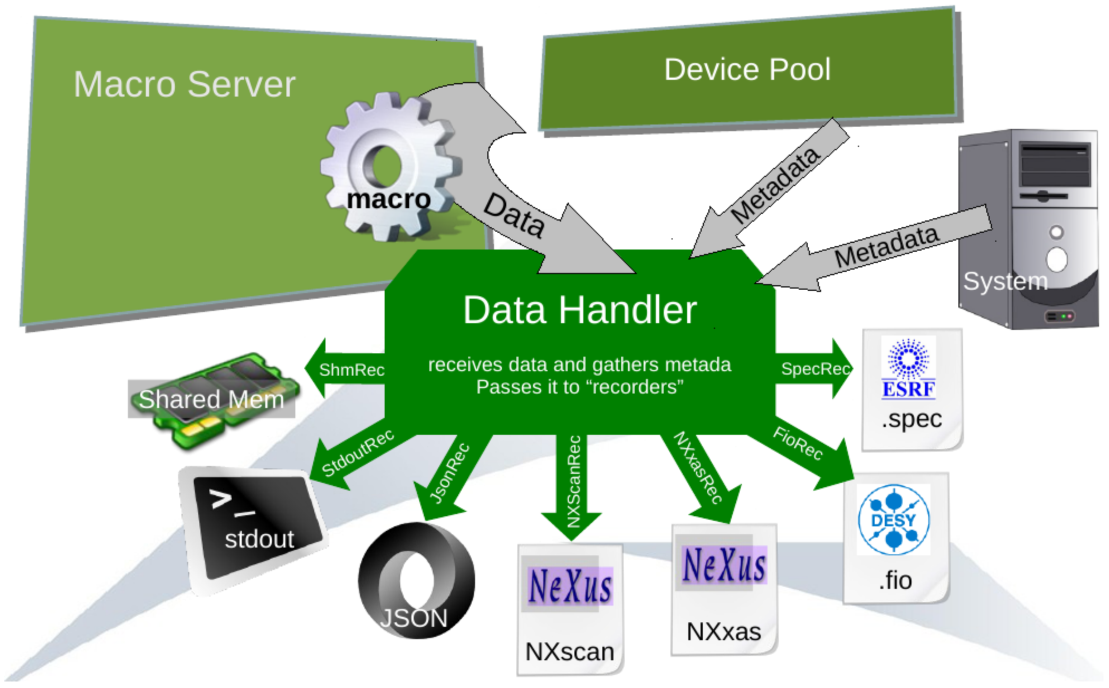


Figure 41: Sardana data acquisition. NeXus files are created by NeXusRecorders

5.5.2 Application definitions

NeXusRecorders produce files that comply with the desired application definition. Currently, only the generic scan application definition (Nxscan) is being used in production, and therefore all scans use the "NXscan_FileRecorder" for storing their data.

In order to support other application definitions within the standard scan framework, specific NeXusRecorders have to be created and the scan macro has to select them explicitly. This mechanism is already available, but, at the time of writing, it has not being used in production. A proof-of-concept consisting of a macro ("xas_acq") and a recorder ("NXxas_FileRecorder") was implemented for storing data from X-ray absorption experiments in a NXxas-compliant file.

For the future, there are plans for creating a base NeXus recorder capable of loading a NXDL application definition in order to simplify the creation of new application definition specific recorders.

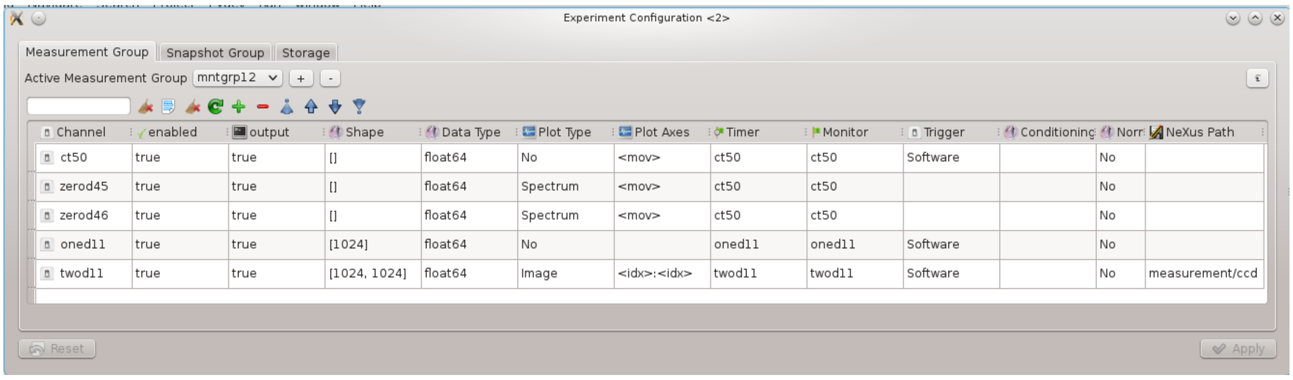
5.5.3 Data and metadata sources selection

Some experiments are fairly standard and therefore, specialized macros/recorders can be created which access a pre-defined set of control system elements/variables. But the dynamic nature of some other experiments that take place at ALBA requires that the person responsible for the experiment is able to configure at least the following aspects:

- which set of elements from the control system are involved in the experiment
- how those elements are accessed as well as their synchronization mechanisms
- the path in which the data from each given element is to be stored within a NeXus file
- which are the standard plots for this experiment (necessary, among other things, for creating NXdata groups within the NeXus files)

Sardana provides an API for accessing and modifying these configuration details. The "Experiment Configuration" widget (Figure 42) makes use of this API to give the user control over most of these settings. Figure 43 shows the TaurusNeXusBrowser widget displaying the contents of a

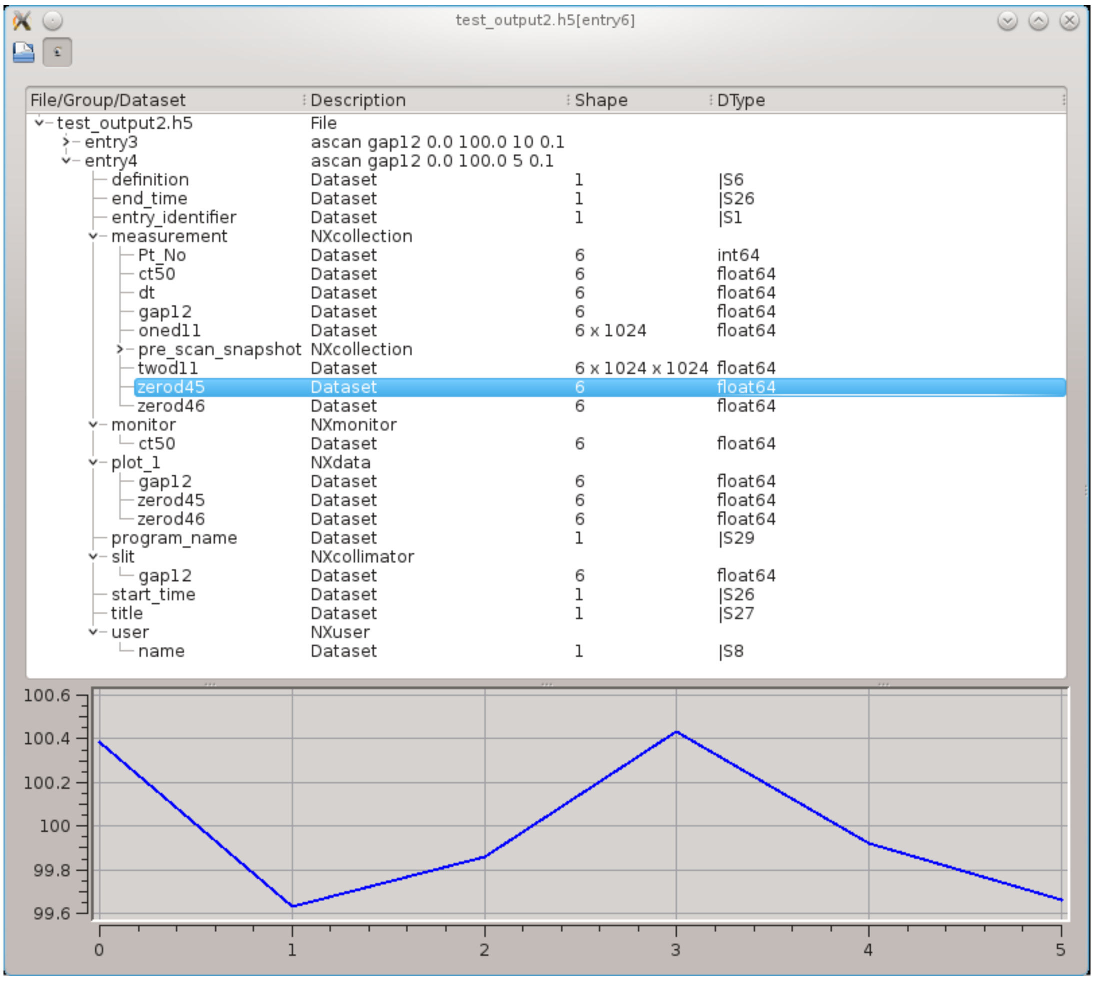
NeXus file.



Channel	enabled	output	Shape	Data Type	Plot Type	Plot Axes	Timer	Monitor	Trigger	Conditioning	Norr	NeXus Path
ct50	true	true	[]	float64	No	<mov>	ct50	ct50	Software		No	
zerod45	true	true	[]	float64	Spectrum	<mov>	ct50	ct50			No	
zerod46	true	true	[]	float64	Spectrum	<mov>	ct50	ct50			No	
oned11	true	true	[1024]	float64	No		oned11	oned11	Software		No	
twod11	true	true	[1024, 1024]	float64	Image	<idx>:<idx>	twod11	twod11	Software		No	measurement/ccd

Figure 42: Sardana: experiment configuration widget

Apart from the user-customizable configurations already described, Sardana provides the possibility of defining virtual "Instruments" which are collections of control system elements that represent the physical or idealized instruments from the point of view of the scientist, e.g., a monochromator or a sample holder. The concept of a Sardana instrument is very close to that of a NeXus instrument, and they are used as the default information for automatically populating the internal path tree of the NeXus files, e.g. for creating NXinstrument and NXsample groups.



File/Group/Dataset	Description	Shape	DType
test_output2.h5	File		
entry3	ascan gap12 0.0 100.0 10 0.1		
entry4	ascan gap12 0.0 100.0 5 0.1		
definition	Dataset	1	|S6
end_time	Dataset	1	|S26
entry_identifier	Dataset	1	|S1
measurement	NXcollection		
Pt_No	Dataset	6	int64
ct50	Dataset	6	float64
dt	Dataset	6	float64
gap12	Dataset	6	float64
oned11	Dataset	6 x 1024	float64
pre_scan_snapshot	NXcollection		
twod11	Dataset	6 x 1024 x 1024	float64
zerod45	Dataset	6	float64
zerod46	Dataset	6	float64
monitor	NXmonitor		
ct50	Dataset	6	float64
plot_1	NXdata		
gap12	Dataset	6	float64
zerod45	Dataset	6	float64
zerod46	Dataset	6	float64
program_name	Dataset	1	|S29
slit	NXcollimator		
gap12	Dataset	6	float64
start_time	Dataset	1	|S26
title	Dataset	1	|S27
user	NXuser		
name	Dataset	1	|S8

Figure 43: Sardana TaurusNexusBrowser (reusing several PyMca widgets)

Finally, Sardana also provides an API allowing macros to pass custom data to the recorders using a simple key-data interface. This mechanism is generic to all recorders (not just NeXus file recorders) but it also supports the transfer of recorder-specific information, such as the NeXus path. This way, a macro can build arbitrary NeXus tree structures that comply with a given application definition.

Regarding the metadata sources, the Experiment Configuration and Instrument APIs described above already constitute important sources of metadata for the creation of NeXus files, but not all of the required metadata can be obtained from them, e.g. information regarding the user responsible for the experiment or information about the beamline.

The scan framework in Sardana does its best to gather some of these missing bits of metadata, e.g. the date and time, the name of the user running the macro, etc., from whatever generic sources are available (mostly by introspection), but still some of the metadata are only available from sources that are not generic among the different institutions using Sardana.

As of today, ALBA relies only on the generic and universal metadata sources and therefore some interesting metadata (e.g. the proposal ID associated to the experiment or the beamline description) is missing in the generated NeXus files. This constitutes the weakest point of the current integration of NeXus in ALBA. One possibility for overcoming this limitation would be the generalization of DESY's nexus implementation and its integration into Sardana. Another solution would be to provide some query plug-ins API within Sardana so that each institution could provide glue code to allow the scan framework to access specific sources of data and metadata

5.6 NeXus Integration at DLS

5.6.1 Overview

At Diamond Light Source writing of NeXus files is primarily orchestrated by the data acquisition system GDA[28]. Notable data sources that feed into GDA are:

- Databases (Operating System, ICAT and ISPyB)
- EPICS[29] (motion, sample environment, detectors, etc.)
- User input (scripted or interactive)
- Other non-EPICS devices

GDA gathers all the information and aggregates the data into a NeXus file, with the exception of (most) EPICS area detectors. These medium to high data rate devices are configured to write their own data files. That allows a more efficient streaming of data to disk. The files written by EPICS area detector at Diamond are HDF5 files with a minimal NeXus structure that makes it possible to be referenced from the main GDA NeXus file in a way that is transparent to any user or analysis program.

When a user launches GDA, the correct visit in ICAT is picked for the user. That determines the proposal metadata to be included and also the top level directory for data files.

The main paradigm that GDA works on for recording data is a scan. Scans can run over a single point (snapshot mode), in 1D, 2D or higher dimensions. NeXus file writing works in pretty much the same way for point (step) scans and for trajectory (fly) scans, but for simplicity here we only cover the point scan procedure.

5.6.2 Device selection

[28] http://www.opengda.org

[29] http://www.aps.anl.gov/epics/

As mentioned earlier GDA records data in a scan. GDA provides a number of ways in which scans can be constructed: Interactively on a command line, via scripts, from a GUI or otherwise programmatically. All scans internally have a list of devices to operate on (with a list of target positions) and a list of devices or detectors to read back at each point. In addition to those explicitly listed devices and detectors the beamline may have additional devices configured to be recorded. For that there are three options:

1. Record at every point for any scan
2. Record a single value for any scan
3. Record a single value as a dependency on a other device included in the scan

Devices at most record at every point, there is no additional single value written to the file if more than one rule applies. The last option is evaluated recursively for example in order to build up a chain motions.

5.6.3 Configuration

NeXus files written by GDA contain a single NXentry, holding exactly one scan. Without special configuration detectors write into an NXdetector node under the name by which they are known in the GDA namespace. Detectors with extended metadata (the norm rather than the exception) provide their partial NeXus tree with is merged into the file.
Other devices in the generic case are placed into groups of their name into NXinstrument, linking the explicitly scanned ones as axes into NXdata. It is worth noting that a GDA device (or detector) aka scannable can be a compounded device with an in principle unlimited number of output parameters. Since most of these devices serve a special purpose on the beamline, like defining sample temperature or photon energy, these devices can be configured to place their data into the valid location in the NeXus file. In some cases the data also needs to be converted into NeXus-acceptable representation. That could involve a simple numerical unit conversion or a more complicated transformation. Both location and conversion are configured manually by the Diamond data acquisition group.
Currently none of the beamlines has a fully complete description of the beamline state in NeXus, only the more prominent items are present to cover the needs of data analysis. This amounts to about 50 to 70 datasets on average.

5.6.4 Default metadata

The amount of metadata that is put in depends on the beamline. As a bare minimum GDA attempts to place the following in the file:

- **`/NXentry/experiment_identifier`** the visit identifier in our proposal system
- **`/NXentry/entry_identifier`** the scan number (aimed to be unique per beamline)
- **`/NXentry/NXuser/username`** the FedID of the user running the experiment
- **`/NXentry/program_name`** identifying GDA and the version number
- **`/NXentry/scan_command`** the command line used to start the scan
- **`/NXentry/NXinstrument/name`** the beamline name
- **`/NXentry/NXsource/name`** the facility name

Anything on top of that requires either user cooperation (in case of title or sample name etc) or sensible local configuration (for example for insertion device parameters). The idea here is to only record sensible information and not to fill in meaningless defaults.

5.5.5 Data Acquisition Process

GDA moves motors, triggers detectors and reads them out, or instructs them to write separate data files. After first the data point of the scan is taken a file is created and the minimal default and configured structure is written as well as data from first point and links to detector files are made. For every further point in the scan the data is added and flushed. After the last scan point the file is closed. Some beamlines have special methods to supply post processing data at that step.

5.6.6 Application definitions

Most files do not claim to follow any application definition (some could). For tomography the conforming part is in a subentry.

5.7 NeXus Integration at HZB

5.7.1 Overview

At HZB, NeXus files are created at the neutron diffractometer E2, E4 and E6. The following we focus on the Neutron Flat-Cone-Diffractometer E2 (BER II). Figure 44 gives an overview of the instrument control and data acquisition processes. Like at most other instruments of the research reactor BER II CARESS controls the whole setup. This program package was developed at HZB. It stores data in a binary file format. In addition the ROOT[30] file format is used for frame data.

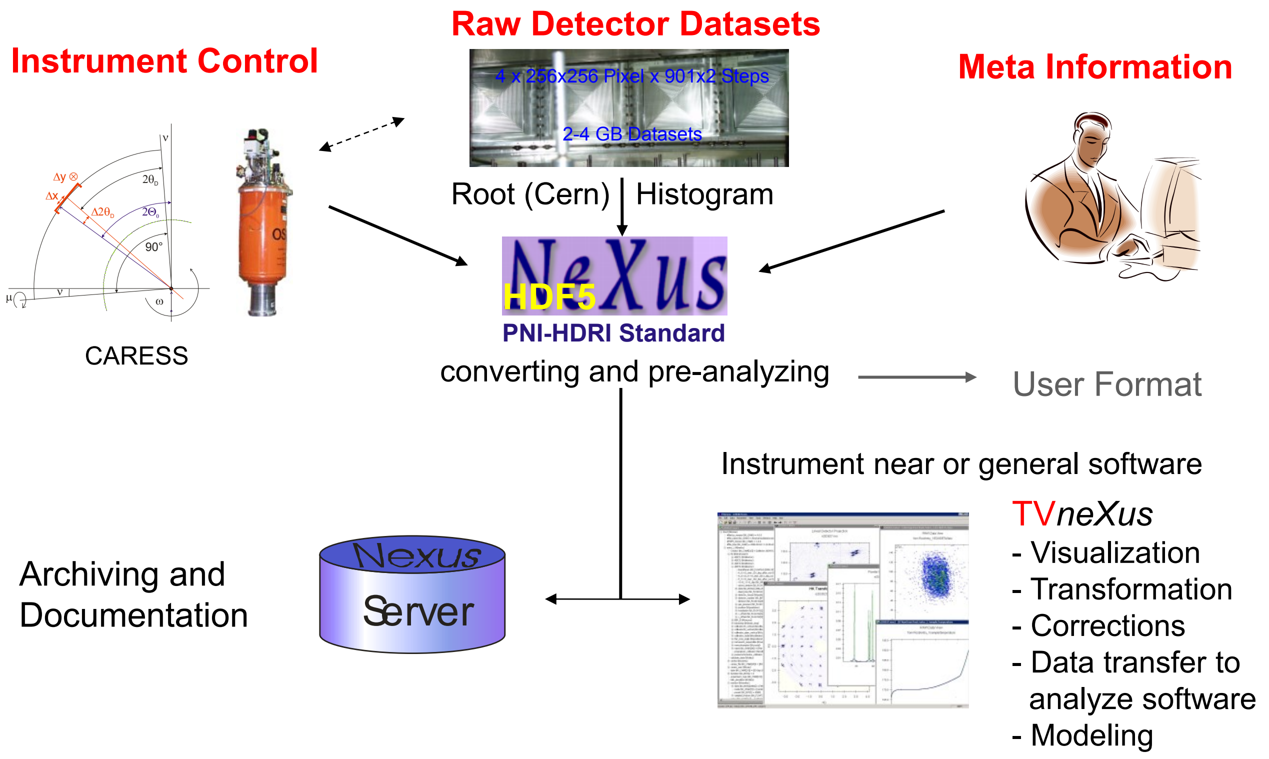


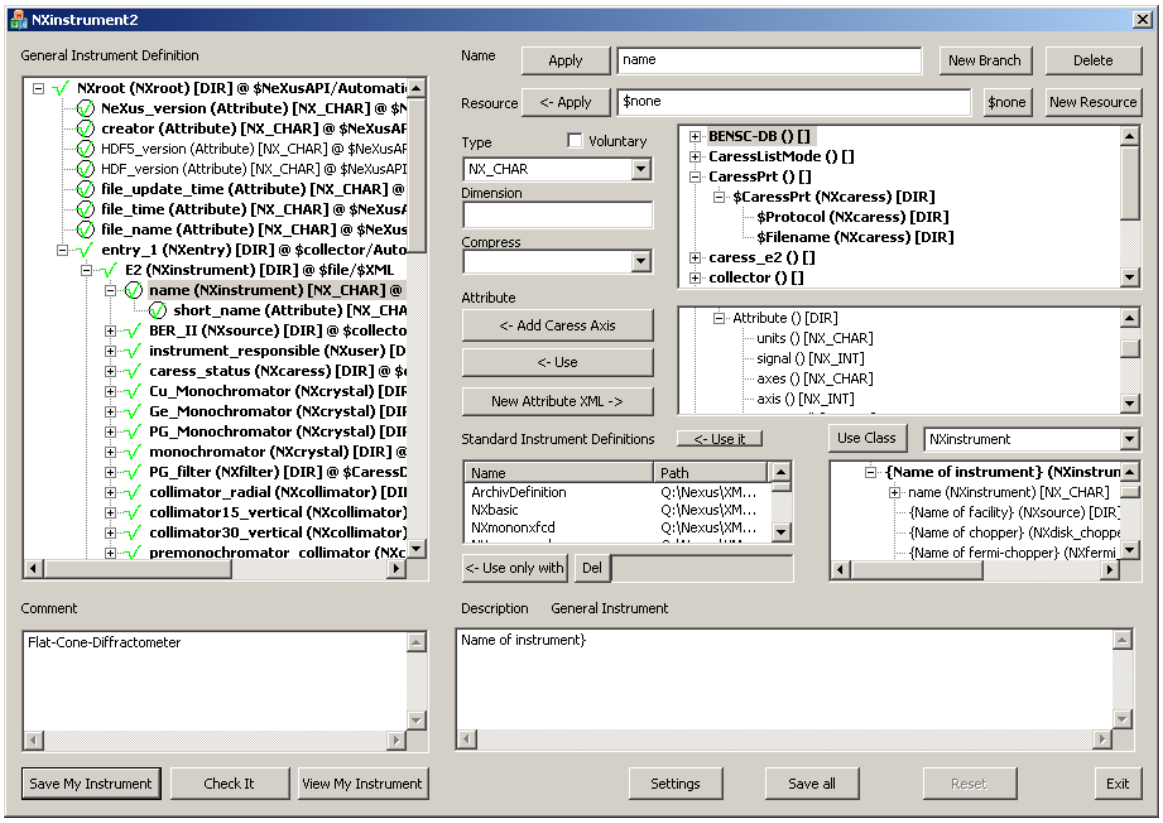
Figure 44: HZB Data from different sources are collected in NeXus files

NeXus files are created by the Data Collector. This application merges data coming from various sources, including CARESS and ROOT files. The Data Collector processes also log files, calibration data and information from the user and sample databases. A configuration tool, Figure 45, has been developed linking data sources and their internal names to NeXus file entities.

[30] root.cern.ch

Depending on the experimental technique applied the Data Collector transforms the raw data into powder diffractograms or into 3-dimensional arrays representing the intensity distribution in reciprocal space. These derived data are inserted into the original NeXus files where they are available for near real time analysis. The aforementioned transformations can be interactively parameterized with TVneXus, a program developed at HZB which also has modelling capabilities.

In addition, HZB provides users with several analysis software packages which are partly NeXus compliant, e.g. MatLab[31], Mathematica[32], Fullprof[33] and Lamp. .



```
NXroot (NXroot) [DIR] @ $NeXusAPI/Automati
  NeXus_version (Attribute) [NX_CHAR] @ $N
  creator (Attribute) [NX_CHAR] @ $NeXusAF
  HDF5_version (Attribute) [NX_CHAR] @ $NeXusAF
  HDF_version (Attribute) [NX_CHAR] @ $NeXusAPI
  file_update_time (Attribute) [NX_CHAR] @
  file_time (Attribute) [NX_CHAR] @ $NeXus/
  file_name (Attribute) [NX_CHAR] @ $NeXus
  entry_1 (NXentry) [DIR] @ $collector/Auto
    E2 (NXinstrument) [DIR] @ $file/$XML
      name (NXinstrument) [NX_CHAR] @
        short_name (Attribute) [NX_CHA
      BER_II (NXsource) [DIR] @ $collecto
      instrument_responsible (NXuser) [D
      caress_status (NXcaress) [DIR] @ $
      Cu_Monochromator (NXcrystal) [DIF
      Ge_Monochromator (NXcrystal) [DIF
      PG_Monochromator (NXcrystal) [DIF
      monochromator (NXcrystal) [DIR] @
      PG_filter (NXfilter) [DIR] @ $CaressD
      collimator_radial (NXcollimator) [DII
      collimator15_vertical (NXcollimator)
      collimator30_vertical (NXcollimator)
      premonochromator_collimator (NXc
```

Name: name
Resource: $none — $none
Type: NX_CHAR

```
BENSC-DB () []
CaressListMode () []
CaressPrt () []
  $CaressPrt (NXcaress) [DIR]
    $Protocol (NXcaress) [DIR]
    $Filename (NXcaress) [DIR]
caress_e2 () []
collector () []
```

```
Attribute () [DIR]
  units () [NX_CHAR]
  signal () [NX_INT]
  axes () [NX_CHAR]
  axis () [NX_INT]
```

Use Class: NXinstrument

Name	Path
ArchivDefinition	Q:\Nexus\XM...
NXbasic	Q:\Nexus\XM...
NXmononxfcd	Q:\Nexus\XM...

```
{Name of instrument} (NXinstrun
  name (NXinstrument) [NX_CHAR]
  {Name of facility} (NXsource) [DIR]
  {Name of chopper} (NXdisk_choppe
  {Name of fermi-chopper} (NXfermi
```

Comment: Flat-Cone-Diffractometer

Description: {Name of instrument}

Figure 45: HZB NeXus configuration tool

5.7.2 Mandatory metadata

The ingestion process uses metadata from the NeXus files and from GATE, the HZB user portal.

NeXus field or attribute	ICAT field	Remark
/NXentry/experiment_identifier	Investigation.name	Proposal number
/NXentry/title	Investigation.title	description of the experiment
/NXentry/NXinstrument/name	Instrument.fullName	name of the instrument
/NXentry/NXinstrument/NXsource/name	Facility.Fullname	Either BESSY II or BER II
/NXentry/NXuser/name	User.fullName	Name of the user
/NXentry/NXsample/name	Sample.name	name of the sample
/NXentry/NXsample/chemical_formula	Sampletype.Molecularform	the chemical formula of the

[31] www.mathworks.de

[32] www.wolfram.com/mathematica

[33] www.ccp14.ac.uk

	ula	sample
/NXentry/run_number	Dataset.name	A name identifying the measurement
/NXentry/start_time	Dataset.startDate	start time of the measure-ment
/NXentry/end_time	Dataset.endDate	end time of the measure-ment

Table 6: Mandatory metadata at HZB

Table 6 contains a list of the metadata and it shows how the NeXus items are mapped to ICAT fields. Note that each measurement corresponds to an entry into the metadata catalogue.

- /NXentry/experiment_identifier identifies the proposal and thus the beamtime. Several NeXus files may belong to the same beamtime. Imported from GATE.
- /NXentry/title the title of the proposal as imported from GATE.
- /NXentry/NXinstrument/name the full name of the instrument, imported from GATE.
- /NXentry/NXuser/name name of the user responsible for the measurement, mported from GATE.
- /NXentry/NXinstrument/NXsource/name either BESSY II or BER II, derived from the instrument.
- /NXentry/NXsample/name the sample name. Must be registered into ICAT either before or during the measurement.
- /NXentry/NXsample/chemical_formula the chemical formula of the sample.
- /NXentry/run_number a unique name identifying the individual measurement within the investigation. Must be attributed during the measurement and will be imported into ICAT during ingestion of the NeXus file.
- /NXentry/start_time start time of the measurement. Must be recorded during the measurement and will be imported into ICAT during ingestion of the NeXus file.
- /NXentry/end_time end time of the measurement. Must be recorded during the measurement and will be imported into ICAT during ingestion of the NeXus file.

5.7.3 Generalizing the HZB NeXus implementation

The HZB has a great experience with the conversion and consolidation of different existing data formats into NeXus files. For the beam lines at BESSY II the conversion of SPEC files to NeXus is tested. The next upgrade of the instrument control software at BESSY will include a deployment of a generalized NeXus framework.

5.8 NeXus Integration at ILL

At the Institut Laue-Langevin a third (11) of the public instrument suite is producing experimental data in NeXus file format, the others are either using domain specific format, usually binary (for instance .lst transform during reduction process in .root for the nuclear physic) either the legacy ASCII ILL format. The ILL is engaged in this standardization program since 2008, which may take two more years to be completed. There is no specific technical difficulty but time is necessary in order to convince stakeholders and allow a smooth transition.

5.8.1 Configuration

The set-up is based on dedicated XML configuration files located on the Nomad server computer. Nomad is the instrument control software developed by the ILL. This is a client server application and the nexus configuration is on the server side. It is configured per instrument by the set-up procedure of the instrument. It is not modifiable by the users. This is a dynamic process: NX components appear in the file only if the corresponding devices are in use during the experiment.

Documentation is available for the users on a public web site (http://forge.epn-campus.eu/projects/nomad/wiki/Nexus_Documentation) on a per instrument basis. The history of change is preserved on a subversion repository, allowing users to browse by date of the experiment.

5.8.2 Device selection

The general structure is defined in two files per instrument containing the static and dynamic properties of those devices selected for a specific measurement. The sample description and administrative information are automatically retrieved from the proposal system once the users have identified themselves to the instrument control software.

5.8.3 Mandatory metadata

/NXentry/experiment_title < Proposal title >
/NXentry/instrument_name < ILL name of the instrument >
/NXentry/reactor_power < the power of the reactor expressed in MW >
/NXentry/run_number < ILL internal run number >
/NXentry/acquisition_mode < type of experiment: time of flight, kinetic, … >
/NXentry/duration < time of measurement in second >
/NXentry/start_time < start time of the measure >
/NXentry/end_time < end time of the measure >
/NXentry/NXuser/name < Name of the user sitting in front of the instrument control >
/NXentry/NXuser/namelocalcontact < Name of the local contact >
/NXentry/NXuser/proposal < ILL proposal identification number >
/NXentry/NXsample/sample_id < Id of the sample register in the proposal system >
/NXentry/NXdata/data < link to the data of the main NXdetector of the instrument >

The full updated list of entries is available per instrument, as a matter of example the current IN4 tof description is downloadable on the following link:
http://forge.epncampus.eu/projects/nomad/repository/raw/ill/Tools/Nexus2Doc/trunk/doc_generated/doc_IN4_TOF.html

5.9 NeXus Integration at ISIS

5.9.1 Overview

ISIS writes a mix of Nexus and raw (binary) files across its suite of instruments. All instruments share a common control/DAE software suit, and are capable of writing nexus files, raw files or both in parallel. During 2014 the majority of instruments wrote nexus files in parallel with raw files.

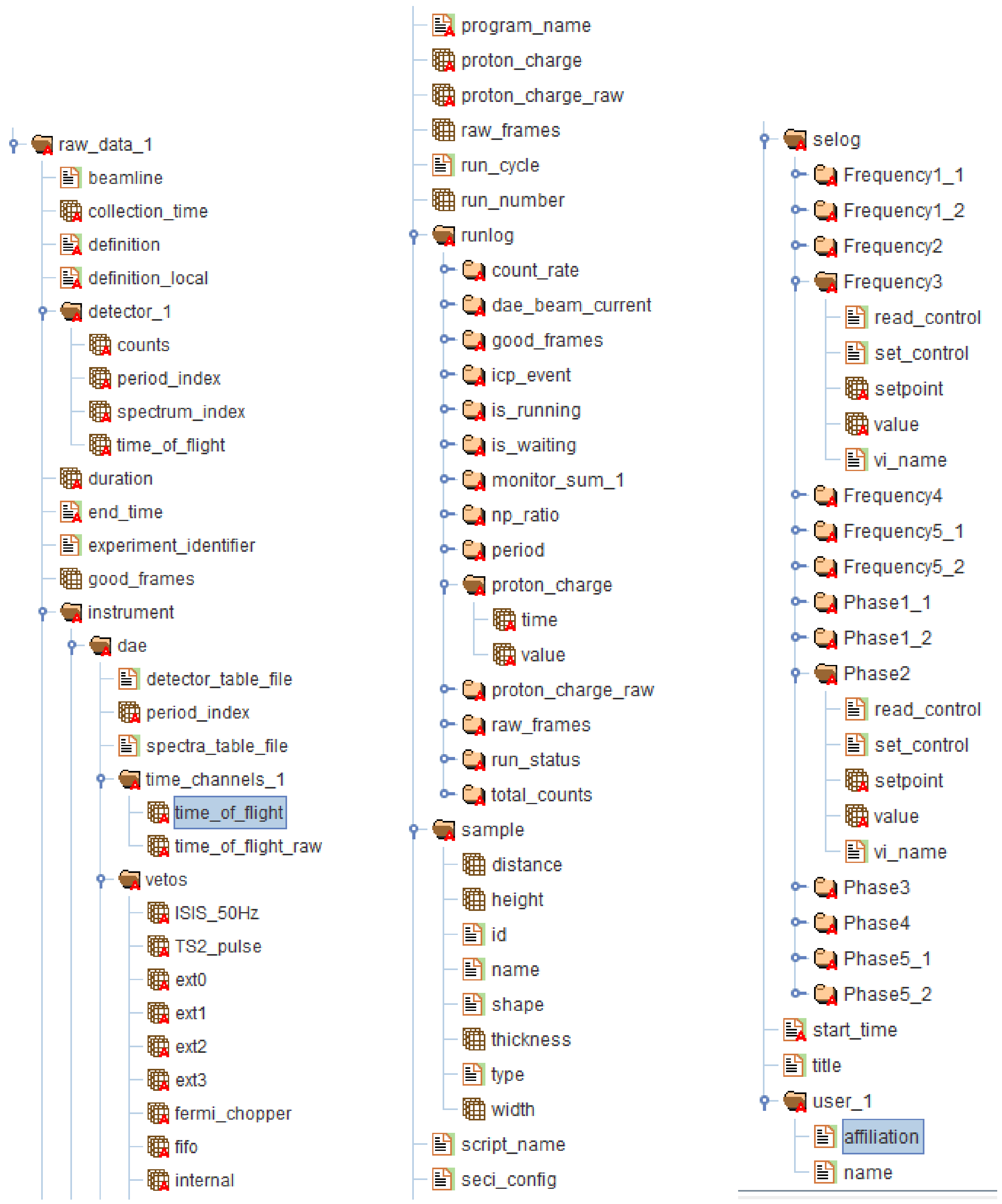
5.9.2 NeXus structure

All nexus files at ISIS are fully self-contained – they do not contain pointers to any external data files. The standard structure is illustrated below (Figure 46). The main detector data (neutron

events) is in the ‘detector_1’ area. Neutron monitors are stored separately in ‘monitor_n’ areas. The nexus file also records the relevant vetos for use in data reduction.
The metadata are also in the nexus file, in a number of places for example – beamline stores the instrument name, title is a user controlled field to describe the data collection run experiment identified the proposal number and the nx:sample structure is used for sample data. Sample environment data/metadata is stored in the selog area. This is a dynamic part of the file, depending on what pieces of sample environment are in use during each experiment. Finally the user area stores the users name and affiliation.

5.9.3 Mandatory metadata

As much as possible of the metadata in the nexus file is collected automatically from the instrument control software: title, sample details, sample environment values, experiment identifier etc. However, the accuracy of this is largely dependent on the diligence of the users in entering meaningful and correct values.



```
raw_data_1
    beamline
    collection_time
    definition
    definition_local
    detector_1
        counts
        period_index
        spectrum_index
        time_of_flight
    duration
    end_time
    experiment_identifier
    good_frames
    instrument
        dae
            detector_table_file
            period_index
            spectra_table_file
            time_channels_1
                time_of_flight
                time_of_flight_raw
            vetos
                ISIS_50Hz
                TS2_pulse
                ext0
                ext1
                ext2
                ext3
                fermi_chopper
                fifo
                internal
    program_name
    proton_charge
    proton_charge_raw
    raw_frames
    run_cycle
    run_number
    runlog
        count_rate
        dae_beam_current
        good_frames
        icp_event
        is_running
        is_waiting
        monitor_sum_1
        np_ratio
        period
        proton_charge
            time
            value
        proton_charge_raw
        raw_frames
        run_status
        total_counts
    sample
        distance
        height
        id
        name
        shape
        thickness
        type
        width
    script_name
    seci_config
    selog
        Frequency1_1
        Frequency1_2
        Frequency2
        Frequency3
            read_control
            set_control
            setpoint
            value
            vi_name
        Frequency4
        Frequency5_1
        Frequency5_2
        Phase1_1
        Phase1_2
        Phase2
            read_control
            set_control
            setpoint
            value
            vi_name
        Phase3
        Phase4
        Phase5_1
        Phase5_2
    start_time
    title
    user_1
        affiliation
        name
```

Figure 46: ISIS NeXus structure

5.10 NeXus Integration at PSI-SINQ

5.10.1 SINQ Overview

SINQ at PSI is writing NeXus data files since it started operation in 1997, first in HDF-4 format, now in HDF-5. Before discussing the implementation of NeXus at PSI some general information about SINQ, its instruments and the data acquisition system used at SINQ is required.

To start with SINQ: SINQ is a medium flux neutron source. Medium flux means that SINQ instruments produce much data by 1997 standards but little by todays (2014) standards. With about 9 month operation per year SINQ, SINQ instruments generate ~70GB data per year with imaging beam lines excluded.

SINQ instruments are of medium complexity. The most complex instrument has about 60 motors and 2 detector arrays. Most SINQ instruments are considerably less complex. There are few configuration changes: most instruments are operated in one of 2-3 possible configurations. The only set of devices which changes frequently is the sample environment. The electronics at SINQ instruments is highly standardized across instruments. Most of the electronics is still 1990 standards. Thus SINQ operates many devices which communicate through RS-232. Such devices are connected to the network via terminal servers. Neutron data acquisition on position sensitive detectors happens in histogramming mode. This means that neutron events detected in the detector are binned into multidimensional arrays right in the frontend electronics. There is a program in place to replace SINQ instrument electronics by a more modern second generation electronics. This new electronics generally communicates via TCP/IP. The second generation DAQ will also support event mode neutron data acquisition. But progress on the upgrade paths is slow due to insufficient resources.

The low data rates and the modest complexity of instruments lets SINQ get away with a less complex data acquisition system. The SINQ data acquisition software is called SICS for: SINQ Instrument Control System. SICS is a client server system. This means that for running SICS at minimum two programs are required:

- A SICS clients which implements the user interface to the data acquisition system
- A SICS server which does all the work: talking to devices, running measurement procedures, writing data files and more. The SICS server includes Tcl as server side scripting language.

SICS clients and the SICS server communicate with each other through a simple ASCII protocol through the TCP/IP network.

Sample environment devices are managed via another server, the SeaServer. This is just another instance of a SICS server; specially configured to operate sample environment.

Due to organizational reasons all of the above is true for the neutron scattering instrumentation. Neutron imaging at SINQ choose a different path and work with LabView, tiff images and little helper programs.

5.10.2 5.10.2 Writing NeXus files at SINQ

All NeXus files at SINQ are written by the SICS server. Writing NeXus files at SINQ is controlled by a combination of scripting and dictionaries. Scripts are executed within the SICS server and are written in the server side scripting language Tcl.

Whenever the SICS server needs to write data, when storing a scan point or a complete data acquisition run, an instrument specific Tcl script is executed. This Tcl script then uses SICS server built in primitives, implemented in ANSII-C, to manage NeXus files and write device data efficiently to NeXus files. These internal primitives are known under the name of nxscript. Nxscript uses an externalized description of the structure of the NeXus files for controlling the placement of data items in the NeXus file. This externalized description is called NXdict. NXdict is a dictionary which maps an alias, a shortcut, to a definition string which describes the data item in the NeXus file. Nxscript takes these definition strings and generates all necessary structural elements from them before writing the actual data.

Thus adding another data item to a NeXus file at a given instruments involves just two steps:

1. Add a line to the NXdict dictionary
2. Add a suitable line to the instrument data file writing script

Configuration is managed via a configuration program, written in server side Tcl, which is executed by the SICS server at startup. This Tcl configuration program initializes the available devices at the instruments, loads additional scripting programs, configures the command set and the scripts needed for data file writing. Different instrument configurations are accounted for by flags in the configuration script.

All this is quite simple and works well. SINQ has written > 2.5 Million valid NeXus files with this system by the time of writing (July 2014).

5.10.3 SINQ NeXus standard compliance

SINQ was one of the first facilities to adopt NeXus. Thus it is no surprise that the NeXus files at SINQ do not conform to the newest state of the NeXus standard anymore. SINQ plans to make all old files application definition conforming once the application definitions have gained sufficient community approval. It is expected that this will require little more than running a small instrument specific program against the NeXus files which will:

- Create links to existing fields at the required places
- Add some additional field properties
- Add or modify very few data fields

5.10.4 SINQ auxiliary NeXus handling software

As all SINQ data collected since 1997 still amount to less than one terrabyte, PSI keeps all SINQ data files on a shared AFS file system. Once files have been written they are read only. PSI keeps another copy of SINQ data files in a tape archive.

An ingestor exists which indexes new NeXus files into an Oracle database for file searching. This ingestor is driven by a nightly cron job. This database system will be phased out in 2014 and replaced by an ICAT catalog.

In addition there exist instrument specific tools for performing primary data reduction and data analysis on NeXus files. The 2.5 Million SINQ NeXus files have been analyzed and results have been published. Thus NeXus works for PSI.

5.11 Summary

This is the list of general conclusions concerning the NeXus deployment at various PaN-data institutes::

- As far as the NeXus integration is concerned the neutron sources are still ahead of the x-ray facilities. However, the latter ones are catching up.
- NeXus configurations based on the device selection are integrated into the local experiment control software. Metadata are gathered from various sources including the digital user offices.
- Application definitions are respected by a small subset of the experiment stations. Nonetheless, concepts for the integration of application definitions exist almost everywhere.
- The concept of mandatory metadata (MMD) is generally acknowledged. Surprisingly the institutes selected more or less the same MMD items independently of each other. The MMD are the basis for the metadata catalogue ingestion procedure. They seem to be sufficient for file discoveries.

6 Comparison of Data Processing Frameworks

6.1 Introduction

This section compares data processing frameworks developed by members of the PaN-data community to evaluate and exploit the synergy potential in this area. It is intended to make partners aware of the ongoing activities and to discuss the applicability of existing solutions. The aim is to foster the re-use of code, to avoid parallel software developments and to promote collaborations among the participating facilities and other institutes.

Most of the experiments carried out at photon and neutron sources pose common computing challenges:

- Visualization applies to raw and derived data, for online and offline purposes. Data to be displayed may originate directly from detectors or it can be read from files.
- Monitoring is important to ensure the quality of the data while the measurement is running.
- Interactive analysis tools allow the scientists to conduct some very basic data processing steps, like normalizations, background subtraction, peak fitting, integration, etc.
- Scripting refers to procedures being executed for online, offline and near real time purposes. The main scripting language within the PaN-data community is Python.
- Batch processing is an extension of script execution. Batch jobs run on central IT installations, controlled by a load balancing system, possibly on dedicated hardware. Batch jobs have access to large file servers.
- Workflow systems allow scientists to construct applications by specifying sequences of data processing steps. At best, the definition of the sequences is supported by a graphical user interface.
- Web portals facilitate the discovery, inspection and download of data. An interface to a workflow system would allow users to select data, choose an application, supply parameters, initiate the workflow execution and monitor the results.

There are strong correlations between the above mentioned tasks:

- Visualization is needed for monitoring, interactive work, scripting and inspecting batch job results or workflow outputs.
- Scripts are used in interactive sessions, for batch processing and for workflows.
- Workflows can be executed online and offline, possibly controlled by a portal. For suitable measurements workflows can also be used for near real-time processing.

Taking these considerations into account it appears that an integrated, modular framework is an excellent solution for the computing tasks mentioned. It reduces the programming effort and eases the burden of the scientists because they only have to familiarize themselves with one system. Good candidates are the open-source projects DAWN[34], DPDAK[35] and Mantid[36]. The remainder of this section contains a comparison of these applications

The next section introduces the three tools presenting their most important ingredients. Esp. DAWN and Mantid offer a large variety of features. It is beyond the scope of this report to discuss

[34] www.DAWNsci.org

[35] dpdak.desy.de

[36] www.mantidproject.org

all of them in detail. Instead we focus on how the programs can be applied to our use cases: an online application for powder diffraction measurements (VL1) and an analysis of small angle scattering data (VL2).

6.2 Description of investigated tools

6.2.1 DAWN

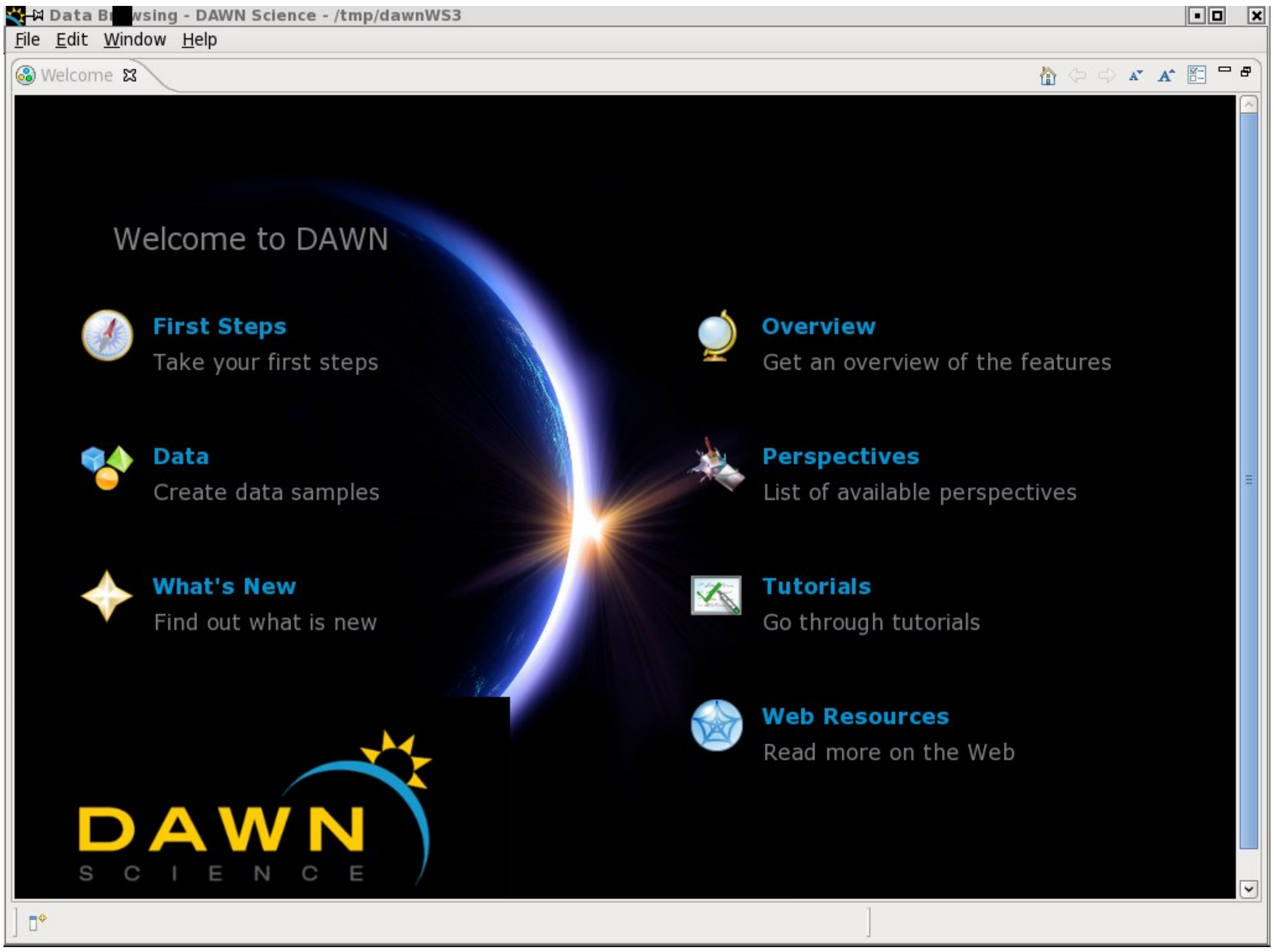


Figure 47: The DAWN welcome screen.

The **D**ata **A**nalysis **W**orkbe**N**ch (DAWN)[37] is an Eclipse based tool box for scientific data analysis, written in Java. DAWN is developed by and for the synchrotron community, but has strong overlap with other communities like neutron scattering, photon science, etc. DAWN implements support for visualisation of data in 1D, 2D and 3D, for Python script development, debugging and execution, and for workflows for analysing scientific data calling Python and binary codes.

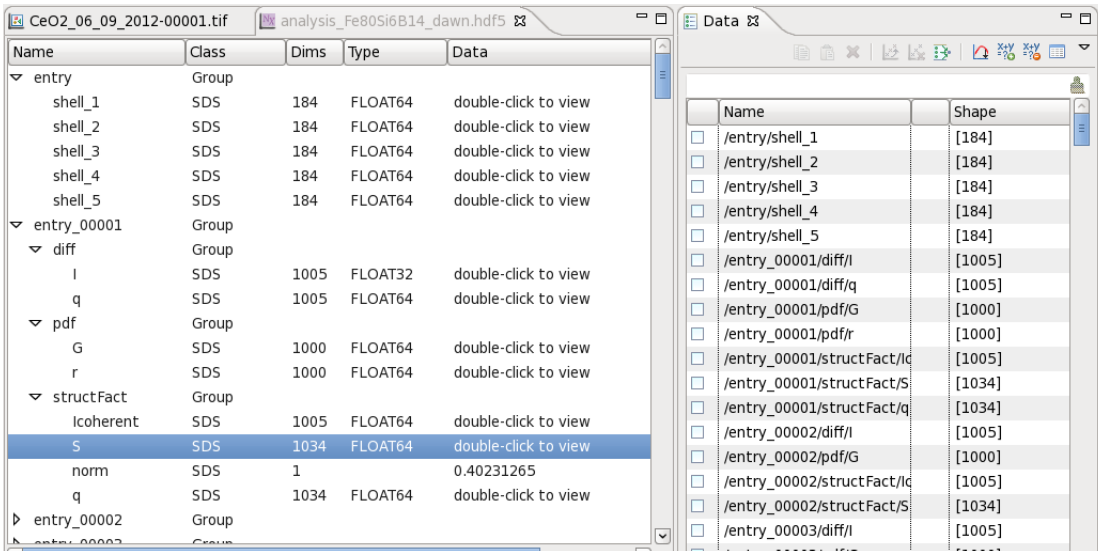
The main contributions to DAWN come from Diamond and ESRF, but development is open to external collaboration. Regular developer meetings ensure an exchange of ideas and the discussion

[37] http://www.DAWNsci.org

of future developments. The core of the source code is available at GitHub[38] and mainly under the Eclipse Public License. The DAWN website provides ready-to-use versions for Linux and Windows, 32 and 64 bit, including a Java run time environment. A MacOS version is also available, but it is not officially supported. Starting DAWN the first time, a welcome screen appears presenting links to various documentation resources, Figure 47.

The graphical user interface is based on Eclipses Perspectives and Views. Each view is a tab in a sub-window of the DAWN frame, providing space for a plot, the content of a file, a Python console, a file browser, a tool, etc. A Perspective manages Views belonging to a specific task, e.g. data browsing or Python scripting. The concept is very flexible. Perspectives can be customized by adding, removing or arranging Views. They can be saved and restored. Data and scripts are organized in Projects which are stored in workspaces. A workspace is a directory in the user's home directory.

DAWN is documented on web pages and by mini-tutorials called “Cheat sheets”. The latter ones guide users step by step through certain tasks. Comments and questions can be sent to the development team via a mechanism which also records the feedback information in a bug tracking system[39]. Issues can be brought to the attention of the DAWN community by means of the DAWN



Name	Class	Dims	Type	Data
entry	Group			
shell_1	SDS	184	FLOAT64	double-click to view
shell_2	SDS	184	FLOAT64	double-click to view
shell_3	SDS	184	FLOAT64	double-click to view
shell_4	SDS	184	FLOAT64	double-click to view
shell_5	SDS	184	FLOAT64	double-click to view
entry_00001	Group			
diff	Group			
I	SDS	1005	FLOAT32	double-click to view
q	SDS	1005	FLOAT64	double-click to view
pdf	Group			
G	SDS	1000	FLOAT64	double-click to view
r	SDS	1000	FLOAT64	double-click to view
structFact	Group			
Icoherent	SDS	1005	FLOAT64	double-click to view
S	SDS	1034	FLOAT64	double-click to view
norm	SDS	1		0.40231265
q	SDS	1034	FLOAT64	double-click to view
entry_00002	Group			

	Name		Shape
☐	/entry/shell_1		[184]
☐	/entry/shell_2		[184]
☐	/entry/shell_3		[184]
☐	/entry/shell_4		[184]
☐	/entry/shell_5		[184]
☐	/entry_00001/diff/I		[1005]
☐	/entry_00001/diff/q		[1005]
☐	/entry_00001/pdf/G		[1000]
☐	/entry_00001/pdf/r		[1000]
☐	/entry_00001/structFact/Ic		[1005]
☐	/entry_00001/structFact/S		[1034]
☐	/entry_00001/structFact/q		[1034]
☐	/entry_00002/diff/I		[1005]
☐	/entry_00002/pdf/G		[1000]
☐	/entry_00002/structFact/Ic		[1005]
☐	/entry_00002/structFact/S		[1034]
☐	/entry_00003/diff/I		[1005]

Figure 48: DAWN browsing a HDF5 file

mailing list.

DAWN has plug-ins for many data formats, like various ASCII formats, HDF5, NeXus, tiff, edf and numpy. Figure 48 shows the HDF5 viewer, a very convenient application for the inspection for all kind of HDF5 files. DAWN’s interface to the ICAT metadata catalogue can be useful as long as the data volumes to be investigated are not too large.

DAWN comprises powerful, interactive visualisation tools:

- The DataBrowsing and DExplore perspectives display one- and two-dimensional slices of multi-dimensional datasets. In addition, also results of expressions can be presented. These are user defined functions of datasets and other expressions. Slices to be displayed

[38] https://github.com/DAWNScience

[39] http://jira.diamond.ac.uk/browse/DAWNSCI

are explicitly specified or selected by a slider in which case the graphical output is instantaneously updated when the slider is moved.

- Detailed investigations of the displayed data are done with zoom tools.
- Several curves can be overlaid, optionally with two different scales in vertical direction.
- The Trace perspective is specialised to work with 1D data imported from several files. The colours and styles of all curves are individually adjusted.
- Two dimensional data are presented as images, surfaces in 3D, or a contour plots. The colour mapping is very flexible with a suitable default algorithm to skip outliers.
- Special Hyper3D views are provided to investigate regions of 3D datasets.

Various tools interact with the displayed data. The most prominent features are:

- Function derivatives can be computed.
- The peak fitting tool finds the position of multiple peaks in a defined region, supporting common peak shapes and assuming a constant background around each peak.
- Polynomials and more complex functions can be fitted to 1D data.
- Profiling tools create projections from ROIs like lines, boxes, sectors, etc. The line profile extracts the intensities along a line. The box profile integrates along one dimension of a box and displays the intensities along the other. The radial (azimuthal) profile uses a sector, integrates in azimuthal (radial) direction and displays the intensities as a function of the radius (azimuthal angle). Masks can be specified that blind-out regions from the input data before the projection is carried out, see Figure 49.
- Some perspectives support specific science analyses like powder diffraction calibration and its application to image data, ARPES data reduction, or alignment and analysis of PEEM images.

Many of these tools can operate on projections of multi-dimensional datasets. For example, starting with a 3D dataset, like a stack of images, a line profile can be extracted from any 2-dimensional sub-space. The result can be saved in an HDF5 file. This feature is called Data Reduction. It is invoked from the from the Data Browsing perspective.

Figure 49 illustrates how some of the aforementioned tools are applied to a powder diffraction image. The left side of the figure contains the original image and a rectangular mask covering the beam stop. In addition an intensity mask has been defined specifying lower and upper intensity limits. All blinded-out pixels are displayed in light-green. The rectangular mask is located at the centre of the image. Pixels that do not pass the intensity cuts are scattered across the entire image, concentrated in circles. The image also contains the position of a line. The intensity distribution along the line is displayed on the right side of Figure 49.

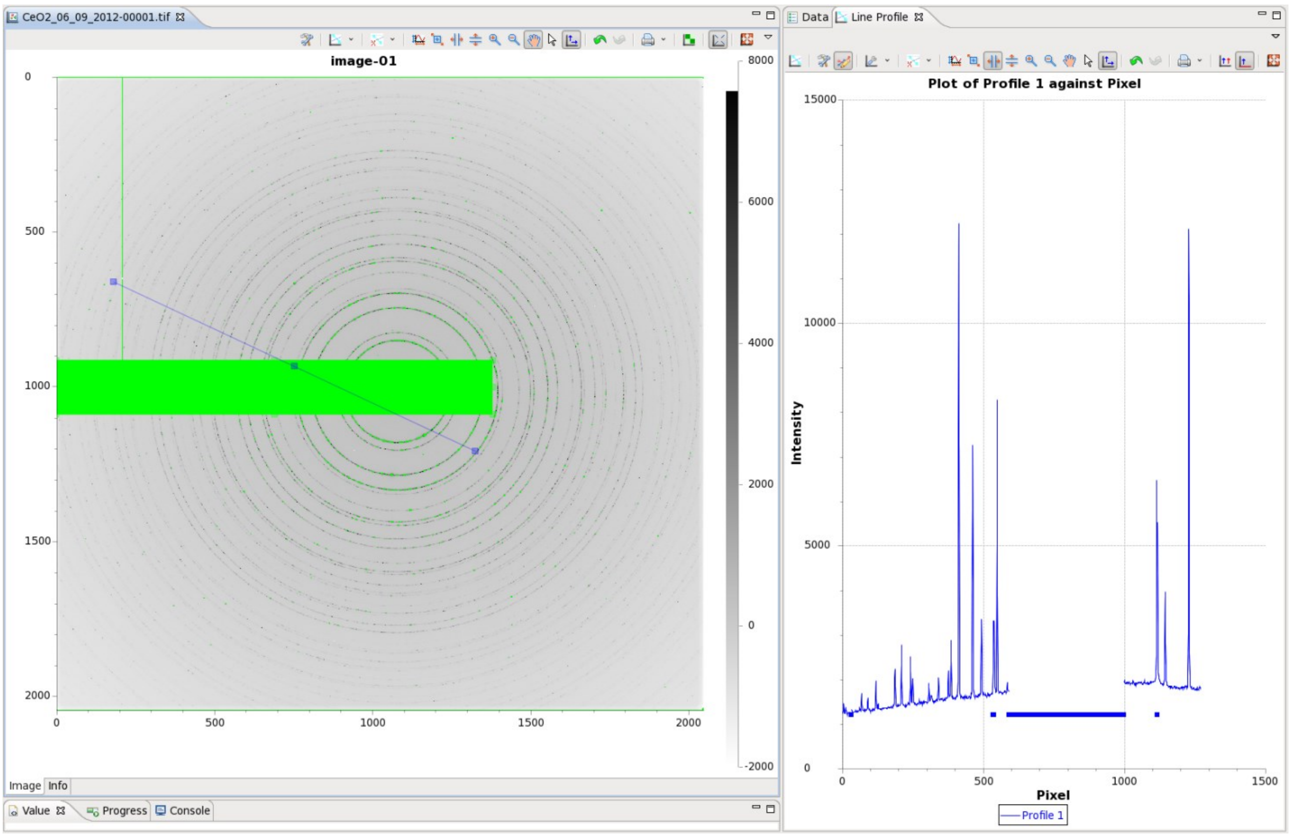


Figure 49: Line profile after application of an image mask.

In addition to the interactive tools, DAWN facilitates data processing and visualisation by means of Python APIs and a workflow engine. Two Python interfaces are supported: CPython and Jython, integrating Python via C and Java. Scientists choose the interface depending on the programming task. If scientific computing is the key aspect, CPython is the best choice because many packages have been developed for this platform. On the other hand, if a script needs to access DAWN's Java packages, Jython is better suited. However, DAWN's Python module `scisoftpy` provides unified CPython/Jython access to many features available in the CPython package NumPy[40], along with I/O, plotting and fitting[41] routines. To ease the development and debugging of Python scripts and modules, PyDev[42], an integrated development environment (IDE) for Python within Eclipse, is integrated in DAWN, providing an IPython command line as well.

DAWN has an integrated workflow system. The basic idea of workflows is to graphically combine actors to assemble data processing chains. Several kinds of actors are available: to request user input via a dialogue window, to open files or to monitor directories, to control the execution flow, to perform calculations, to execute user supplied Python code, to display or to review plots or to export data to files, etc. The graphical representation of workflows facilitates the understanding of the processing steps and the dataflow through complex procedures. Parallelization is implemented by means of threads. Workflows can also run in batch mode without the DAWN GUI. Work is ongoing to port this feature to cluster systems.

[40] http://www.numpy.org

[41] Fitting via `scisoftpy` is currently not fully supported in CPython.

[42] http://pydev.org

Since DAWN is based on the Eclipse IDE, the functionality can be extended in a standardized way by means of java plug-ins. This way, an experienced user integrates additional features like specific file loaders, fit functions, workflow actors, conversion algorithms, tools acting on data, colour mappings, and perspectives. The newly created plug-ins can be shared with others.

6.2.2 DPDAK

The **D**irectly **P**rogrammable **D**ata **A**nalysis **K**it (DPDAK)[43] is a Python framework for analysing large sequences of 2D scattering data. It is developed in a cooperation between DESY and the MPI of Colloids and Interfaces. The source code is available under the GNU GPLv3 via a subversion repository[44]. The Python sources are directly executable on Linux, provided the package dependencies are satisfied. They are listed in the installation instructions on the DPDAK home page. The web page provides a Windows version that contains all the dependencies.

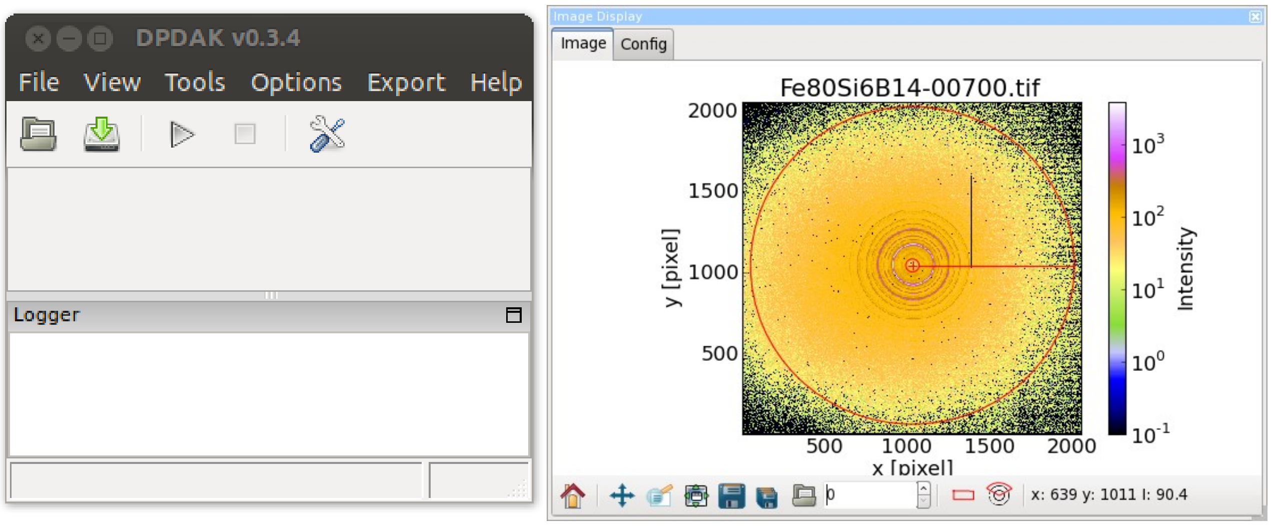


Figure 50: DPDAK main window and image display

The DPDAK design is influenced by the workflow concept. Every application is a sequence of computing tasks that are carried out by plug-ins. Three types of plug-ins exist: data processing, display tools and data I/O. The interface of processing plug-in is defined by their input, output and configurable parameters. Supported data types are scalars, one dimensional arrays, strings and file names or directory paths. Starting from the main window (Figure 50, left) a data processing chain can be graphically configured by selecting plug-ins, configuring inputs and outputs, and specifying parameters. Data that are created by plug-ins are transferred to the next plug-ins and are in parallel stored in an in-memory database from where they can be fetched for visualization purposes or data export. DPDAK has graphical user interfaces to display and analyse data interactively.

The DPDAK distribution comprises several standard plug-ins which are documented on the home page:

- The DirectoryImageLogger senses new files in a directory and returns their names.
- The ImageLogger plug-in returns sequences or sub-sequences (e.g. every 10th files) of file names.

[43] https://dpdak.desy.de

[44] https://svnsrv.desy.de/viewvc/dpdak

- There are plug-ins for calculating sums or averages, intensity corrections, image subtractions and image integrations.
- A ROI plug-in calculates its minimum, maximum, and average intensity as well as the sum of intensities.
- The powder plug-in integrates diffraction images radially or azimuthally using the Fit2D program or the `pyFAI` package[45].
- A plug-in supports variants of horizontal or vertical line integration.
- A peak fit plug-in allows fitting of curve data, possibly restricted to a sub-range of the curve. The fit function is configurable. The fit results can graphically be compared with the data.
- NeXus/HDF5 I/O is currently not supported. It is planned to write a suitable plug-in.
- A calibration tool for powder diffraction (sample – detector distance, wavelength, detector orientation) has been added recently.

New plug-ins can easily be integrated – their Python code is made available by copying it to a specific directory.

DPDAK offers a visualization tool for 2D data (Figure 50, right). The internal data structure can be filled with images being read from files or by combining 1D arrays that are extracted from multiple images. Another plug-in produces graphical output from 1D data. Curves can be plotted against the pixel index, a frame number or an arbitrary sequence of numbers. An overlay feature is implemented.

6.2.3 Mantid

The Mantid project created a framework for high-performance computing and visualisation of scientific data. Mantid manipulates and analyses Neutron and Muon scattering data, but could be applied to many other techniques[46]. The development is driven by ISIS in UK and Oak Ridge National Laboratory in the USA. Contributions come from, among others, the Institut Laue-Langevin in France and the PSI in Switzerland. The framework is written in C, C++ and Python. The source code is hosted on GitHub[47]. Mantid is an open source project under the GNU GPLv3 license. Various Windows, Linux and Mac platforms are supported and each release is citable with a digital object identifier.

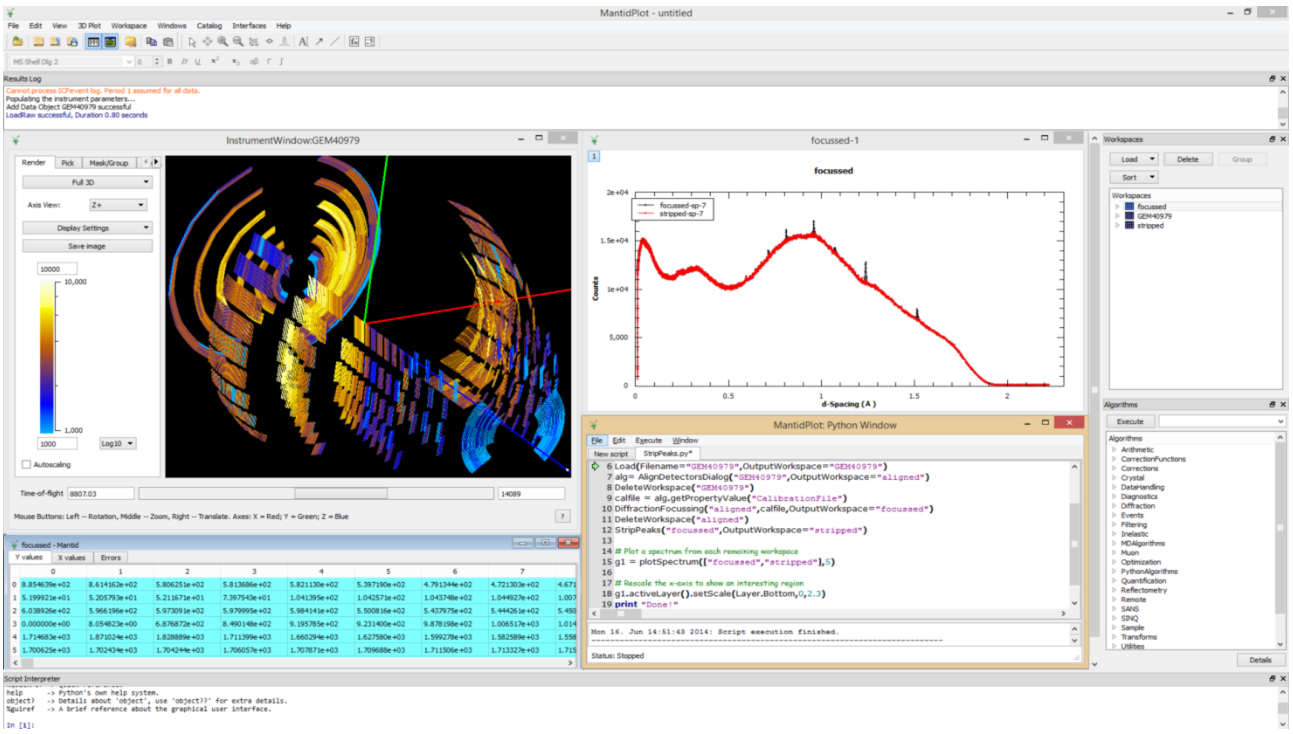
The GUI MantidPlot is based on the core Mantid functionality and QtiPlot[48]. A MantidPlot session can be stored as a “project” to resume the current work later. In the following, Mantid and MantidPlot will not be distinguished. Both will be referred to as Mantid. The Mantid homepage provides extensive documentation, including instructive material of training courses that are regularly offered e.g. at ISIS.

[45] https://forge.epn-campus.eu/projects/azimuthal

[46] http://www.mantidproject.org

[47] https://github.com/mantidproject

[48] http://www.qtiplot.com



```
Cannot process ICPevent log. Period 1 assumed for all data.
Populating the instrument parameters...
Add Data Object GEM40979 successful
LoadRaw successful, Duration 0.80 seconds
```

Time-of-flight 8807.03 14089

Mouse Buttons: Left -- Rotation, Middle -- Zoom, Right -- Translate. Axes: X = Red; Y = Green; Z = Blue

	0	1	2	3	4	5	6	7	
0	8.854639e+02	8.614162e+02	5.806251e+02	5.813686e+02	5.821130e+02	5.397190e+02	4.791344e+02	4.721303e+02	4.671
1	5.199921e+01	5.205793e+01	5.211671e+01	7.397543e+01	1.041395e+02	1.042571e+02	1.043748e+02	1.044927e+02	1.007
2	6.038926e+02	5.966196e+02	5.973091e+02	5.979995e+02	5.984141e+02	5.500816e+02	5.437975e+02	5.444261e+02	5.450
3	0.000000e+00	8.054823e+00	6.876872e+02	8.490148e+02	9.195785e+02	9.231400e+02	9.878198e+02	1.006517e+03	1.014
4	1.714683e+03	1.871024e+03	1.828889e+03	1.711399e+03	1.660294e+03	1.627580e+03	1.599278e+03	1.582589e+03	1.558
5	1.700625e+03	1.702434e+03	1.704244e+03	1.706057e+03	1.707871e+03	1.709688e+03	1.711506e+03	1.713327e+03	1.715

```python
Load(Filename="GEM40979",OutputWorkspace="GEM40979")
alg= AlignDetectorsDialog("GEM40979",OutputWorkspace="aligned")
DeleteWorkspace("GEM40979")
calfile = alg.getPropertyValue("CalibrationFile")
DiffractionFocussing("aligned",calfile,OutputWorkspace="focussed")
DeleteWorkspace("aligned")
StripPeaks("focussed",OutputWorkspace="stripped")

# Plot a spectrum from each remaining workspace
g1 = plotSpectrum(["focussed","stripped"],5)

# Rescale the x-axis to show an interesting region
g1.activeLayer().setScale(Layer.Bottom,0,2.3)
print "Done!"
```

```
Mon 16. Jun 14:51:43 2014: Script execution finished.
--------------------------------------------------------------------------------
```

Status: Stopped

```
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Figure 51: Mantid display

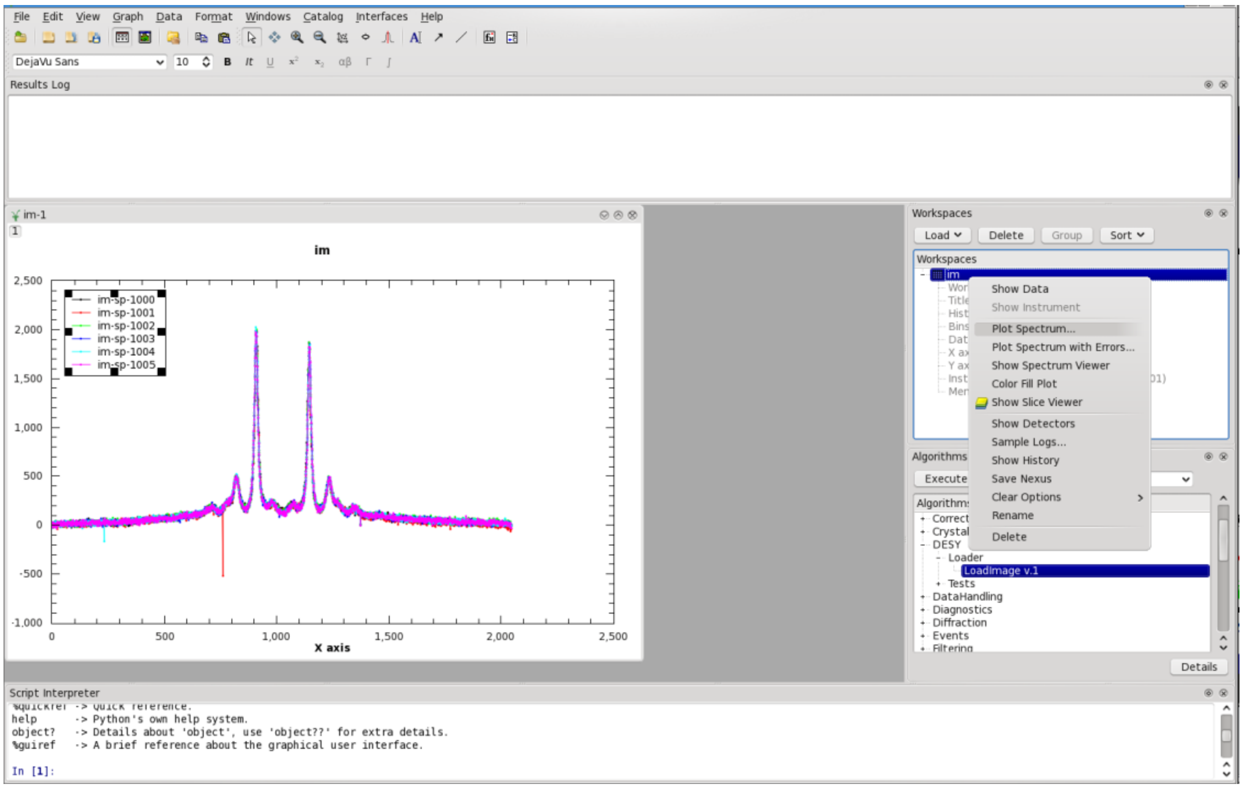
It is an essential concept of Mantid to internally organize data as workspaces. They have different flavours.

- A Workspace2D contains one or more lists of (X, Y, E) triples. (X, Y, E) stands for position, value (e.g. counter reading) and error. If a list is interpreted as a histogram, the X components are understood as bin boundaries. Otherwise lists represent spectra, including errors.
- An EventWorkspace looks like a Workspace2D, but each histogram can be arbitrarily rebinned since individual events are stored underneath.
- A MDHistoWorkspace is a multi-dimensional structure storing signal and error as a function of up to nine coordinates.
- A MDEventWorkspace is similar to MDHistoWorkspace storing individual events as the EventWorkspace.
- A TableWorkspace is a table holding data in rows of columns of particular data types (text, integer, floating point, etc.).

Further workspace types exist as variants of the previously mentioned ones. They can also hold metadata like units of axes and instrument information.

Mantid creates 2D or 3D views of neutron events including the detector positions (Figure 51) provided that XML-formatted experiment descriptions have been prepared. They are available for all experiment stations at ISIS and SNS.

Several input formats are supported. However, there is the restriction that the contents of the files have to be converted to workspaces. An interface to the ICAT metadata catalogue is included. Further information about this feature can be found in section 3.4.



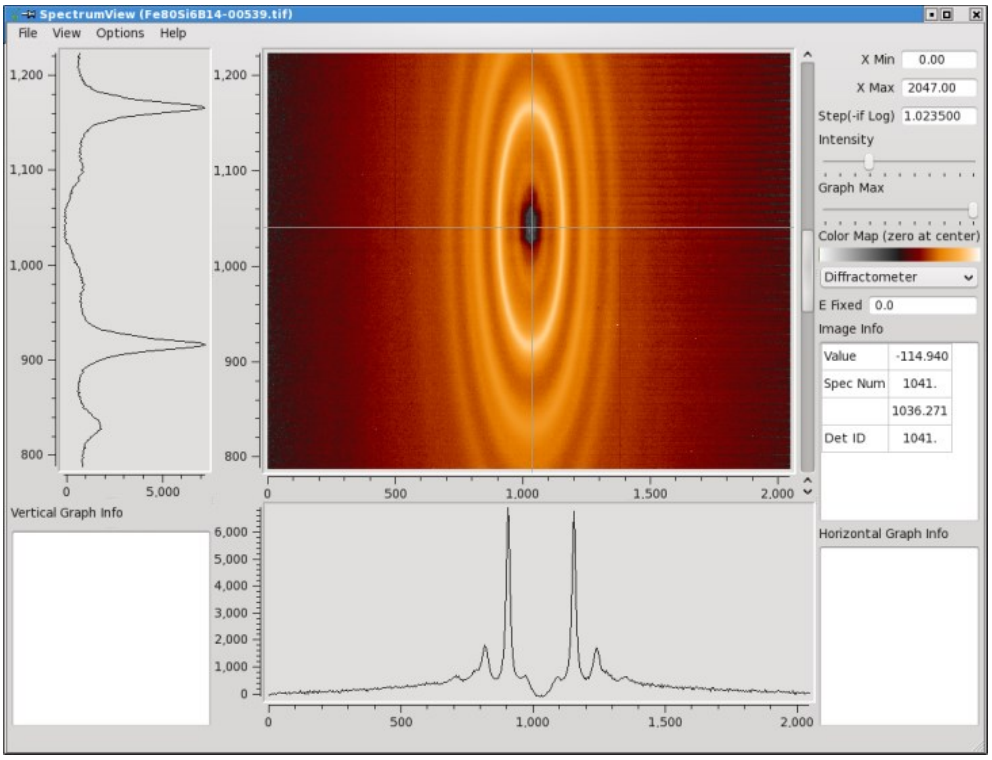
```
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Figure 52: Mantid plotting five consecutive rows

Mantid comprises powerful tools to visualise and investigate workspaces, e.g.:

- (X,Y,E) triples contained in Workspace2D or EventWorkspace can be presented as tables.
- Multiple histograms can be overlaid, as shown in Figure 52. Graphical attributes are changed with the help of context sensitive menus or by mouse clicks.
- An advanced feature allows to insert an axis break and to choose linear and logarithmic scale independently before and after the break. Furthermore it is possible to overlay curves with different scales on either or both of the x and y direction, placing additional axes at the top or left of the plot.
- The SpectrumView displays images, see Figure 53, together with the contents of horizontal and vertical lines which have been selected by a mouse-click into the image.
- The SliceViewer presents 2D slices of MDWorkspaces as images with a large choice of colour maps.
- A LineViewer is integrated to show the distribution along one selected dimension.

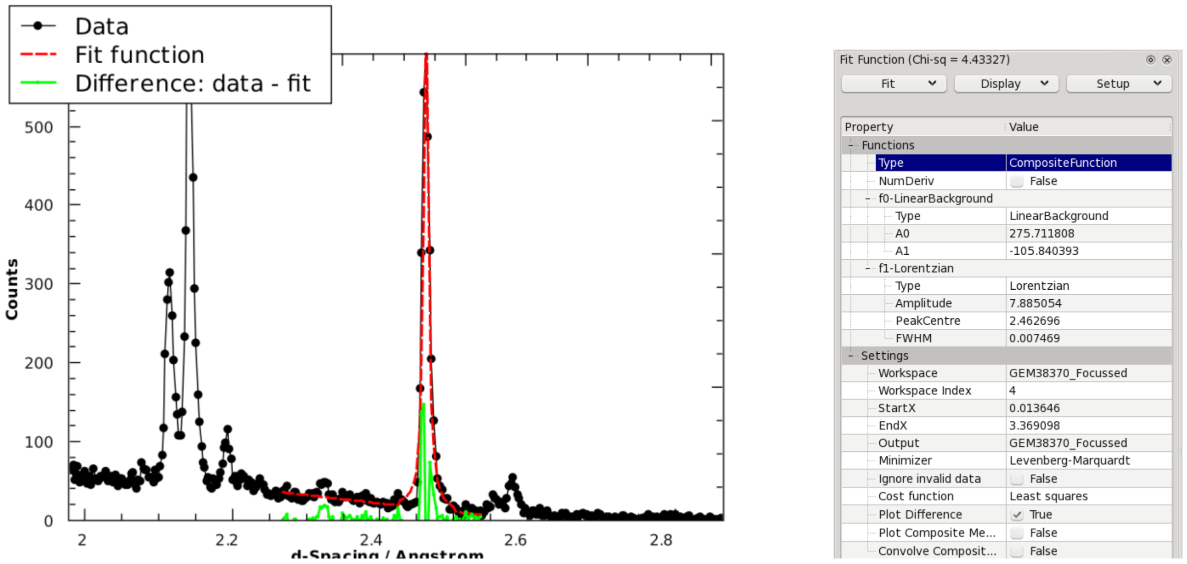


X Min	0.00
X Max	2047.00
Step(-if Log)	1.023500
Color Map (zero at center)	Diffractometer
E Fixed	0.0

Value	-114.940
Spec Num	1041.
	1036.271
Det ID	1041.

Figure 53: Mantid SpectrumView

Interactive function fitting is one of Mantids highlights, see Figure 54. Fit functions are selected from a long list of peak, background and functions for other shapes. The fit range can be graphically specified. The data and the fit function are simultaneously displayed allowing for a convenient specification of the start values of the fit parameters. It is possible to fix parameters, set upper and/or lower bounds or define constraints between parameters. The fitted parameter values and their errors are stored in a special workspace, the normalised covariance matrix in another one. By default, also the fitted function and the distribution of the residuals are overlaid with the data.



Property	Value
Functions	
Type	CompositeFunction
NumDeriv	False
f0-LinearBackground	
Type	LinearBackground
A0	275.711808
A1	-105.840393
f1-Lorentzian	
Type	Lorentzian
Amplitude	7.885054
PeakCentre	2.462696
FWHM	0.007469
Settings	
Workspace	GEM38370_Focussed
Workspace Index	4
StartX	0.013646
EndX	3.369098
Output	GEM38370_Focussed
Minimizer	Levenberg-Marquardt
Ignore invalid data	False
Cost function	Least squares
Plot Difference	✓ True
Plot Composite Me...	False
Convolve Composit...	False

Figure 54: Function fitted interactively to curve data (left) and the fit options dialogue (right).

Data processing in Mantid is organised by means of algorithms acting on workspaces. They can be invoked interactively or through an API. Algorithms may prompt input from the user by launching dialogues. Users typically supply names of workspaces, scalars, arrays, strings or names of output files. The dialogues display a short description of the algorithm and tooltips for each parameter. Further information can be retrieved from web pages which are linked to help buttons. A progress bar informs the user about the status of the calculations. There is also the option to cancel data processing. Workspaces keep track of their history. Each executed algorithm is recorded together with a complete list of parameters supplied.

Whenever algorithms are repeatedly executed in a certain sequence, they can be grouped into so-called interfaces customizing Mantid to meet discipline-specific requirements. A considerable number of interfaces are pre-installed. They have the additional advantage that the user communicates with only one dialogue.

Mantid has a full Python integration. It offers an IPython command line, a simple Python editor and APIs to the algorithms and graphics libraries. Users may extend the Mantid functionality by writing Python classes which inherit from algorithm or fit function base classes.

6.3 Monitoring diffraction measurements

Description of the science case

A disordered or amorphous material is investigated by means of the reduced pair distribution function (PDF). The PDF is a function of the atomic distance r and represents variations of the atomic density relative to the mean density. During the experiment, the environment conditions of the sample are changed, in this case the sample is heated up at a constant rate. The analysis procedure described in the following is based on the article "*In situ XRD studies of nanocrystallization of Fe-based metallic glass: a comparative study of reciprocal and direct space methods*"[49]:

In an in-situ experiment, the sample is placed into a monochromatic X-ray beam (wavelength $\lambda = 0.20712$ A) and the scattered photons are detected in a 2D area detector that is read out at a frequency of 0.1 Hz. As sketched in Figure 55, the following series of analysis steps need to be

[49] J. Bednarcik, S. Michalik, V. Kolesar, U. Rütt, H. Franz, *Phys. Chem. Chem. Phys.*, 2013, **15**, 8470-8479.

performed to calculate the reduced PDF from the image and to monitor its properties:

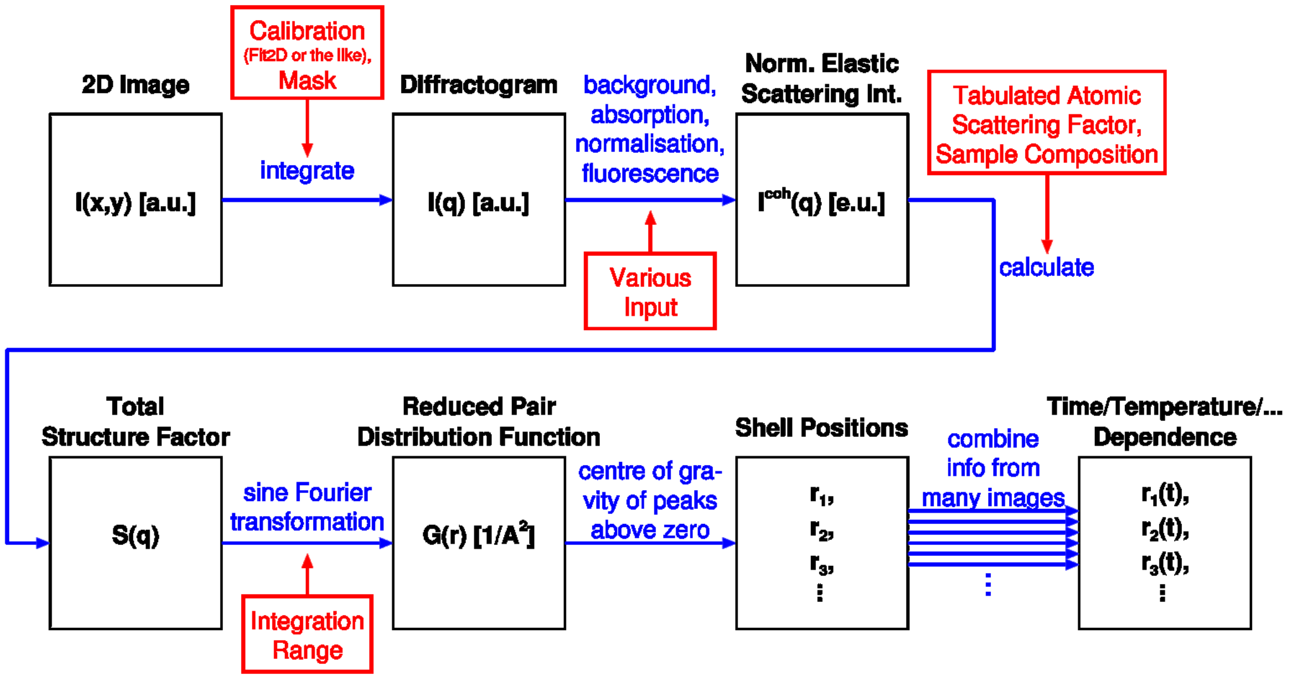


Figure 55: Analysis flow for monitoring the evolution of properties of the reduced PDF of an amorphous material. Black boxes represent data, blue text describes the data processing step, and red boxes mention further input needed.

- The raw images contain intensity values as a function of the pixel index.
- Using geometry information and the photon wavelength, the images are integrated to a diffractogram, i.e. an intensity distribution as a function of the momentum transfer q in reciprocal space. Detector position and orientation are obtained by a powder diffraction calibration procedure using e.g. the Fit2D program[50] or the pyFAI scripts[51]. The integration properly takes into account a mask of unusable image pixels.
- A background image containing the scattering pattern of the complete experimental setup excluding the data sample, and the effect of a fluorescence pattern are subtracted and an absorption correction is applied before the intensity distribution is normalised to obtain the coherent scattering intensity $I_{eu}^{coh}(q)$ in electron units.
- Using the atomic composition of the sample and the tabulated atomic scattering factor[52], the total structure factor $S(q)$ is derived.
- The reduced PDF $G(r)$ is then obtained by a sine Fourier transformation (using a finite integration range) that converts from reciprocal to real space. The $G(r)$ distribution oscillates around zero with an amplitude that gets smaller for larger atomic distances r.
- The average distances, aka shells, of the first, second, third, etc. nearest neighbours can be approximated by the centres of gravity of the peaks above zero.

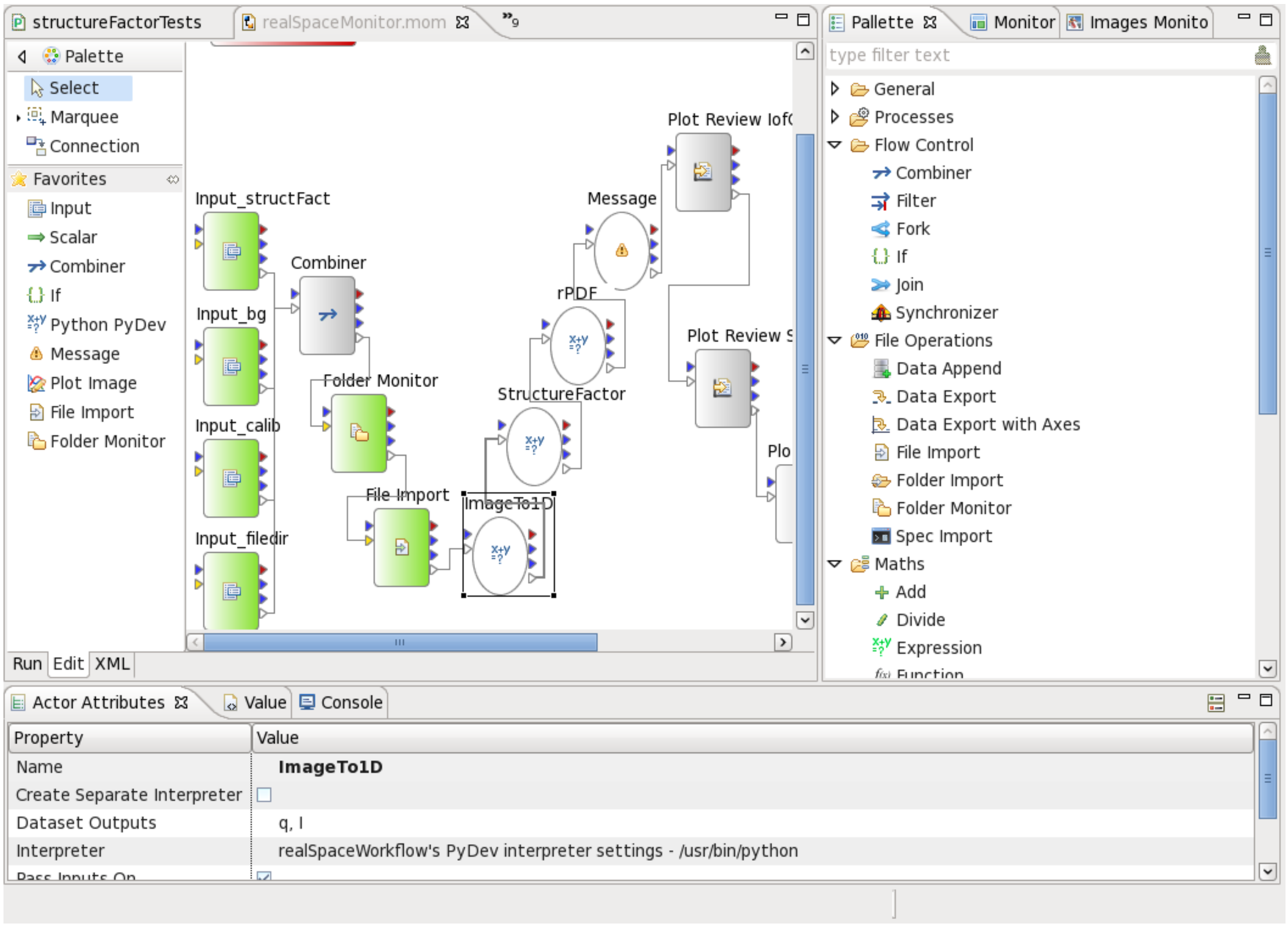
[50] A. Hammersley, S. Svensson, M. Hanfland, A. Fitch, and D. Häusermann, *High Pressure Res.*, 1996, **14**, 235.

[51] J. Kieffer, D. Karkoulis, *J. Phys.: Conf. Series*, 2013, **425** 20, 202012.

[52] D. Waasmaier, A. Kirfel, *Acta Cryst.*, 1995, **A51**, 416-413.

- The shell positions are then monitored with time (and thus with temperature) for each incoming picture.

6.3.1 Diffraction monitoring using DAWN

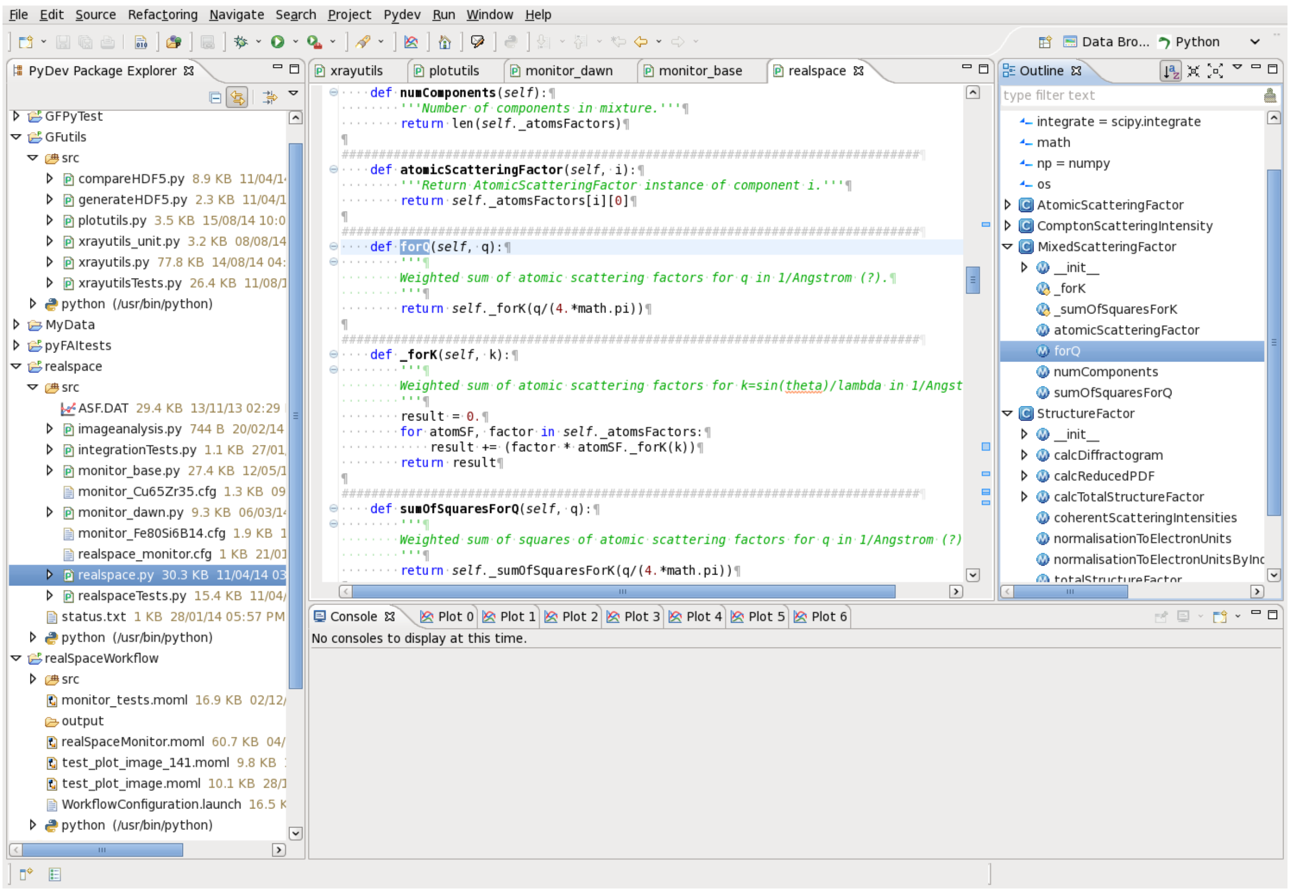


Property	Value
Name	**ImageTo1D**
Create Separate Interpreter	☐
Dataset Outputs	q, I
Interpreter	realSpaceWorkflow's PyDev interpreter settings - /usr/bin/python
Pass Inputs On	☑

Figure 56: DAWN used to develop a workflow for monitoring diffraction data taking

The first attempt to implement a monitoring procedure for diffraction measurements has been to utilise DAWN's workflow feature. “Actors” for several needed analysis steps already exist, e.g. actors to ask the user for input, to give messages, to monitor a directory for new files, to open files and to extract datasets, to store results in an output file, simple maths on dataset, etc. Complex mathematical calculations can be done by so called PyDev actors that run any Python code from DAWN's workspace, e.g. to compute diffractograms by image integrations or to calculate structure factors and reduced PDFs. Figure 56 shows how DAWN looks like when developing such a workflow.

It turns out that the above mentioned monitoring workflow cannot be done with the set of actors provided by DAWN. For example, a conceptual problem is to have input actors called only once while the actor to monitor a data directory should provide an image path whenever a new file appears. Also, the actor to plot a distribution to be monitored does not work without further integration of the specific workflow into DAWN by some Java code. Therefore, the monitoring task is implemented using Python scripts. One main advantage of using Python is that any code used for calculations can be re-used in other programs that understand Python.



```
    def numComponents(self):
        '''Number of components in mixture.'''
        return len(self._atomsFactors)

################################################################################
    def atomicScatteringFactor(self, i):
        '''Return AtomicScatteringFactor instance of component i.'''
        return self._atomsFactors[i][0]

################################################################################
    def forQ(self, q):
        '''
        Weighted sum of atomic scattering factors for q in 1/Angstrom (?).
        '''
        return self._forK(q/(4.*math.pi))

################################################################################
    def _forK(self, k):
        '''
        Weighted sum of atomic scattering factors for k=sin(theta)/lambda in 1/Angst
        '''
        result = 0.
        for atomSF, factor in self._atomsFactors:
            result += (factor * atomSF._forK(k))
        return result

################################################################################
    def sumOfSquaresForQ(self, q):
        '''
        Weighted sum of squares of atomic scattering factors for q in 1/Angstrom (?)
        '''
        return self._sumOfSquaresForK(q/(4.*math.pi))
```

```
No consoles to display at this time.
```

Figure 57: Navigating through a large Python module using the ‘Outline’ view on the right.

A new PyDev project, which organizes the Python files, is created from the File menu via ‘File → New → PyDev Module’. DAWN offers a choice of templates for files containing a class, a main program, etc. In our example we create a ‘realspace’ project containing a ‘realspace.py’ module based on the ‘Class’ template. Notice that for larger programs the ‘Outline’ view is very helpful to navigate inside files. It can be found in the right in Figure 57. The Python console output, which is displayed at the bottom of the screen, supports debugging. If a script terminates with an exception, the console contains links to the relevant lines in the source code. Furthermore, the PyDev console allows for interactive syntax checks and it offers other debug tools that cannot be covered here.

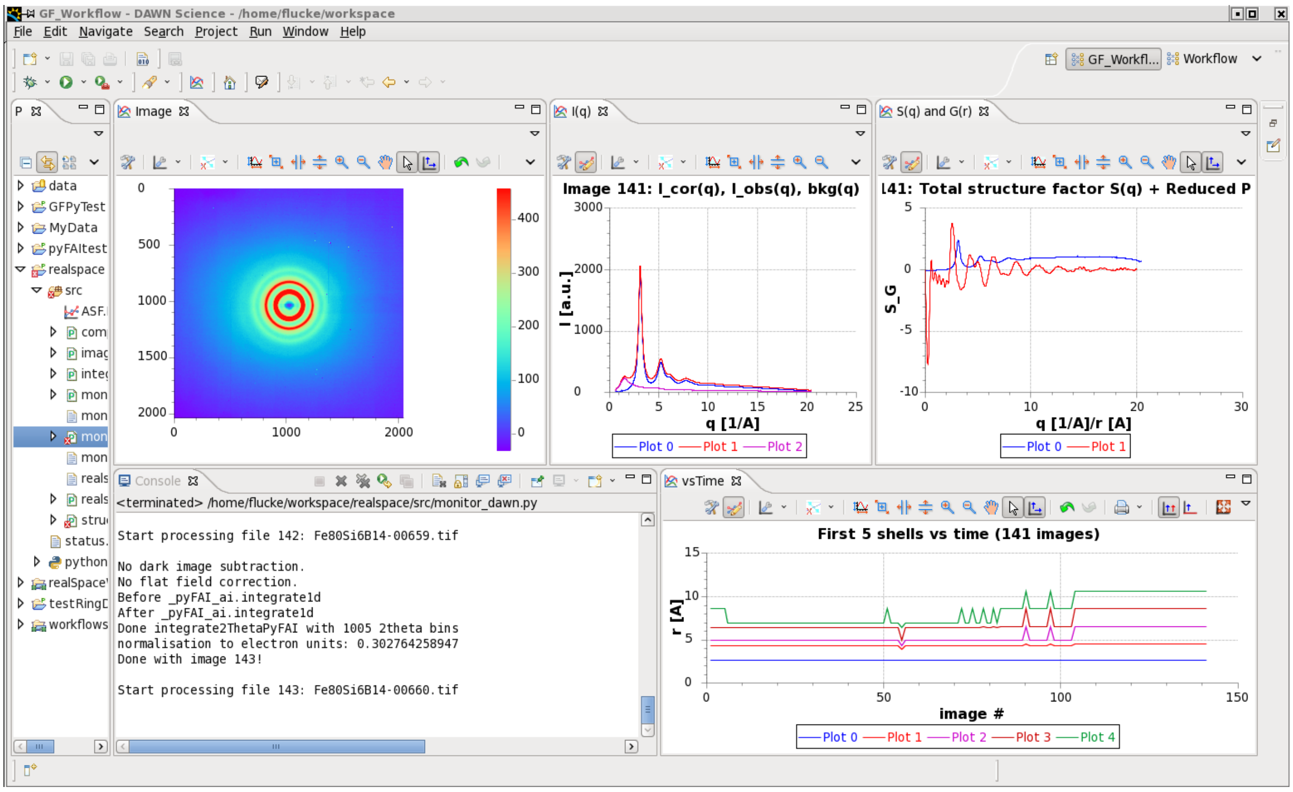
The main Python components, developed for the monitoring procedure, are the following:

- The main loop watches a directory for new files that match a given pattern: The code is re-used from a DPDAK plug-in. The files are opened according to their modification dates and the extracted 2D arrays are passed to other methods. If all new files are processed, the directory is scanned again for new files.
- A method `calcDiffractogram` azimutally integrates the image to create a diffractogram $I(q)$, using the photon wavelength and the geometry information obtained from a previous calibration. A mask for bad pixels can be given. The `pyFAI` package is used for the integration. The method `calcDiffractogram` multiplies the diffractogram with given sample absorption coefficient and subtracts a background image, recorded with no sample in the beam. The result is the diffractogram $I^{cor}(q)$.
- The class `AtomicScatteringFactor` calculates the atomic scattering factor $f_i(q)$ for any kind of atom i, based on a table provided by the publication stated in footnote 52.

- The class `MixedScatteringFactor` calculates $\langle f^2\rangle(q) = \sum_{i=1}^{N} c_i * f_i^2(q)$ and $\langle f\rangle^2(q) = \left[\sum_{i=1}^{N} c_i * f_i(q)\right]^2$ for a material composed of N kinds of atoms with fractions c_i.
- The class `CompontScatteringIntensity` calculates the incoherent (Compton) scattering intensities $I_{eu}^{inc}(q)$ (in electron units) of a compound from its composition fractions c_i and from the atomic numbers Z_i of its components.
- The method `normaliseToElectronUnits` calculates, given the integration range q_{min}^{norm} to q_{max}^{norm} , the normalisation factor α based on $\langle f^2\rangle(q)$, $I_{eu}^{inc}(q)$, and $I^{cor,flu}(q) = I^{cor}(q) - I_{fluo}$. The fluorescence contribution I_{fluo} is assumed to be isotropic and is an input parameter to be estimated by the user.
- The method `coherentScatteringIntensities` calculates the coherent scattering intensity $I_{eu}^{coh}(q) = \alpha * I^{cor,fluo}(q) - I_{eu}^{inc}(q)$.
- The method `calcTotalStructureFactor` uses `CompontScatteringIntensity`, `normaliseToElectronUnits` and `coherentScatteringIntensities` to calculate the total structure factor as $S(q) = 1 + \frac{I_{eu}^{coh}(q) - \langle f^2\rangle(q)}{\langle f\rangle^2}$. Usually, the q range is extended from its smallest value q_{min} towards 0 by assigning $S(q < q_{min}) = S(q_{min})$.
- The method `sineFourierTrafo` calculates the reduced PDF as $G(r) = \frac{2}{\pi}\int q * [S(q) - 1]\sin(rq)dq$ for a given range of radii r, using a given integration range q_{min}^{pdf} to q_{max}^{pdf}.
- The method `calcShellCentres` finds the positions of a given number of n shells as centres of gravity of subsequent regions where $G(r) > 0$, starting the search at a given r_{min}.
- Results are stored in an HDF5 file using the `h5py` package[53]. One Hdf5 group is created for each processed image, containing e.g. $S(q)$ and $G(r)$. An additional group contains one dataset for each of the n shell positions, storing their evolution with time. Care has to be taken to flush the file from time to time to ensure that the file is readable if the Python process is stopped from DAWN's PyDev console. This flushing is done whenever the directory is scanned twice in a row. A configurable delay is taken into account then before the next scan is done.
- The class `MonitorBase` organises the full process and reads all configuration parameters from a configuration file using Python's `ConfigParser`. The class `MonitorDAWN` inherits from `MonitorBase`, implementing the method for opening the files and the method for plotting, both by using DAWN's `scisoftpy.io` and `scisoftpy.plot` Python modules, respectively.

After all necessary classes have been prepared the project can be launched within the Python perspective. It is then possible to switch between the plot views and investigate details using tools provided by DAWN for, e.g. zooming, fitting, etc. The views can be interactively arranged by dragging them to convenient places. Once a final layout is found, it is saved as a new perspective.

[53] http://www.h5py.org/


GF_Workflow - DAWN Science - /home/flucke/workspace
File Edit Navigate Search Project Run Window Help
GF_Workfl... Workflow
data
GFPyTest
MyData
pyFAItest
realspace
src
Image
I(q)
S(q) and G(r)
Image 141: I_cor(q), I_obs(q), bkg(q)
I [a.u.]
q [1/A]
Plot 0 Plot 1 Plot 2
141: Total structure factor S(q) + Reduced P
S_G
q [1/A]/r [A]
Plot 0 Plot 1
Console
vsTime
First 5 shells vs time (141 images)
r [A]
image #
Plot 0 Plot 1 Plot 2 Plot 3 Plot 4

```
<terminated> /home/flucke/workspace/realspace/src/monitor_dawn.py

Start processing file 142: Fe80Si6B14-00659.tif

No dark image subtraction.
No flat field correction.
Before _pyFAI_ai.integrate1d
After _pyFAI_ai.integrate1d
Done integrate2ThetaPyFAI with 1005 2theta bins
normalisation to electron units: 0.302764258947
Done with image 143!

Start processing file 143: Fe80Si6B14-00660.tif
```

Figure 58: DAWN PDF online display

Figure 58 shows the customized perspective during a measurement. A $Fe_{80}Si_6B_{14}$ alloy in amorphous form is measured while it is constantly heated up. The Image view shows the most recently processed image. The ‘I(q)’ and ‘S(q) and G(r)’ views display the curves calculated for that image. In ‘I(q)’ the pure observed $I^{obs}(q)$, the background and the corrected $I^{cor}(q)$ are overlaid. Note that this screenshot was done with an older version of DAWN that did not yet allow passing names for the curves to be used in the legend. In ‘S(q) and G(r)’ the total structure factor and the reduced PDF are overlaid despite of their different x-axis values. One can clearly see the shells as areas where $G(r) > 0$. The little wiggles on top of this main structure are artefacts of the sine Fourier transformation of $S(q)$ that, e.g. due to imperfect calibration and normalisation, does not level out at 1 for large q as it should. The last view, ‘vsTime’, displays the evolution of the positions of the first five shells found. On the broad scale, they look rather constant besides the fact that they jump up or down sometimes. These jumps are artefacts caused by approximations made in the online application. For monitoring purposes this is acceptable. However, the final analysis has to take these effects carefully into account.

6.3.2 Diffraction monitoring using DPDAK

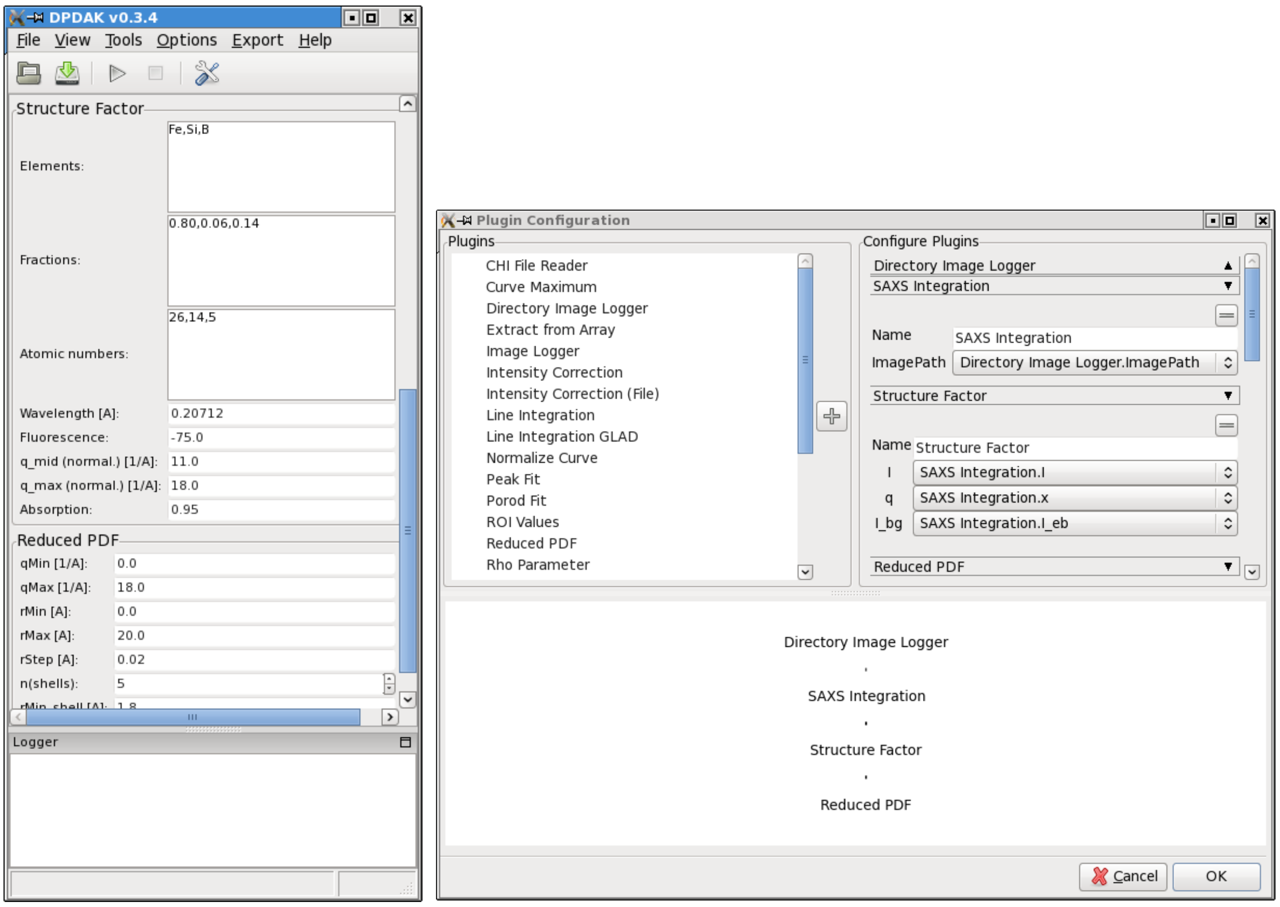
The DPDAK concept is well suited to detect newly created image files in a directory, to do calculations for each image and to display one and two dimensional data derived from the image data. Plug-ins for some central parts of the monitoring task are already available in DPDAK:

- The DirectoryImageLogger searches for new files in a directory and returns them one after the other sorted according to the modification time. The search pattern may include wild cards.
- The SAXSIntegration plug-in calculates the diffractograms $I(q)$ by azimutal integrations of images using calibration parameters and optionally an empty beam image and a bad pixels mask.

- The ImageDisplay plug-in visualises images. Several colour schemes in linear or logarithmic scale are available.
- The MultiPlot plug-in visualises one or more 1D datasets deduced from an image. In addition, the time development of scalar data deduced from image series can be displayed.

Two plug-ins are created that use Python code from the DAWN PDF application:

- The StructureFactor plug-in takes $I(q)$ and $I_{eb}(q)$ as inputs and calculates $S(q)$, using the method `calcTotalStructureFactor`. The empty beam background and the corrections for the sample absorption are taken into account.
- The ReducedPDF plug-in uses `sineFourierTrafo` to calculate the reduced PDF from $S(q)$ and `calcShellCentres` to search for the positions of up to five shells.



Structure Factor

Elements:	Fe,Si,B
Fractions:	0.80,0.06,0.14
Atomic numbers:	26,14,5
Wavelength [A]:	0.20712
Fluorescence:	-75.0
q_mid (normal.) [1/A]:	11.0
q_max (normal.) [1/A]:	18.0
Absorption:	0.95

Reduced PDF

qMin [1/A]:	0.0
qMax [1/A]:	18.0
rMin [A]:	0.0
rMax [A]:	20.0
rStep [A]:	0.02
n(shells):	5

Plugins

CHI File Reader
Curve Maximum
Directory Image Logger
Extract from Array
Image Logger
Intensity Correction
Intensity Correction (File)
Line Integration
Line Integration GLAD
Normalize Curve
Peak Fit
Porod Fit
ROI Values
Reduced PDF
Rho Parameter

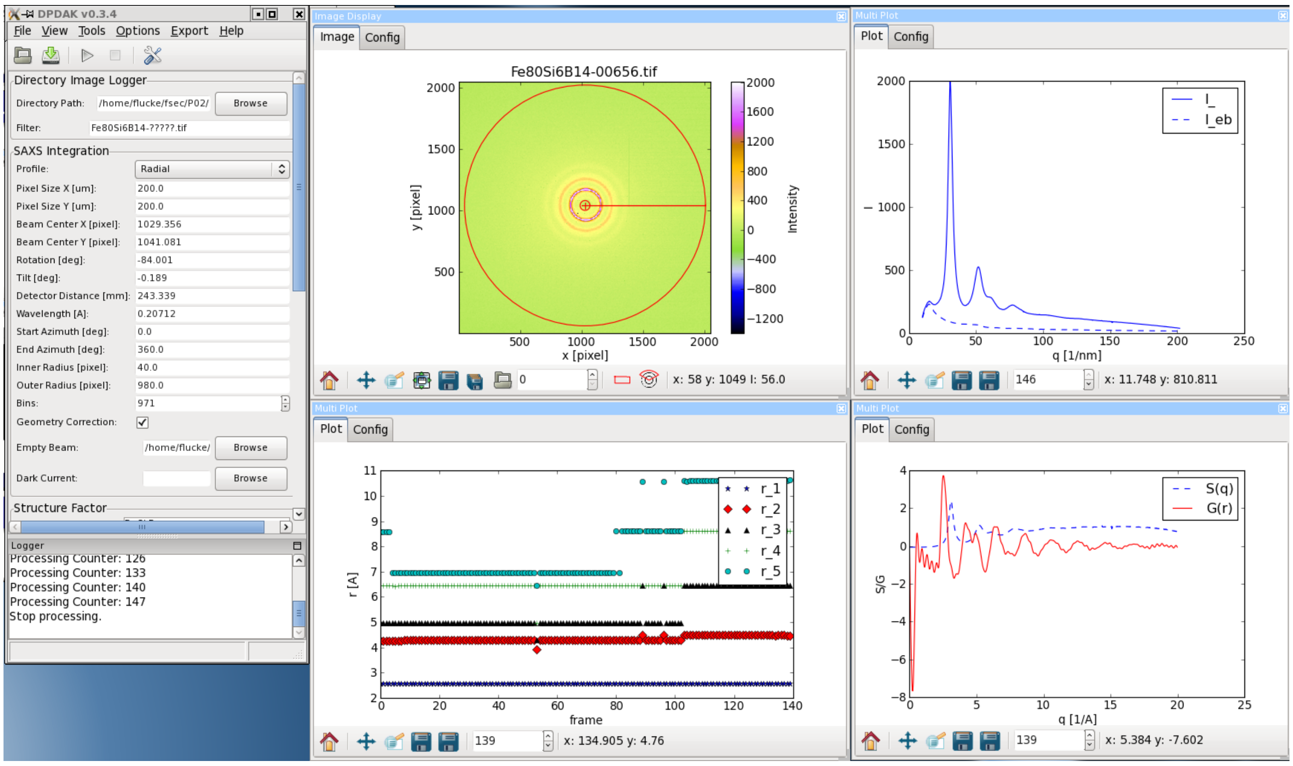
Configure Plugins

Directory Image Logger
SAXS Integration
Name: SAXS Integration
ImagePath: Directory Image Logger.ImagePath
Structure Factor
Name: Structure Factor
I: SAXS Integration.I
q: SAXS Integration.x
I_bg: SAXS Integration.I_eb
Reduced PDF

Directory Image Logger
SAXS Integration
Structure Factor
Reduced PDF

Figure 59: DPDAK configuration with newly created plug-ins.

Figure 59 shows DPDAKs configuration. Plug-ins are selected with the window on the right hand side. The main window (left) is divided into frames containing the parameters for each plug-in. The configuration is saved when DPDAK exits. In addition, configurations can be stored and loaded via the ‘File’ menu of the main DPDAK window.The display plug-ins are opened via the ‘Tools’ menu. They are individually configured. Unfortunately, the configuration display plug-ins is not preserved.



Directory Image Logger
Directory Path: /home/flucke/fsec/P02/
Filter: Fe80Si6B14-?????.tif

SAXS Integration
Profile: Radial
Pixel Size X [um]: 200.0
Pixel Size Y [um]: 200.0
Beam Center X [pixel]: 1029.356
Beam Center Y [pixel]: 1041.081
Rotation [deg]: -84.001
Tilt [deg]: -0.189
Detector Distance [mm]: 243.339
Wavelength [A]: 0.20712
Start Azimuth [deg]: 0.0
End Azimuth [deg]: 360.0
Inner Radius [pixel]: 40.0
Outer Radius [pixel]: 980.0
Bins: 971
Geometry Correction: ✓
Empty Beam: /home/flucke/
Dark Current:

Structure Factor

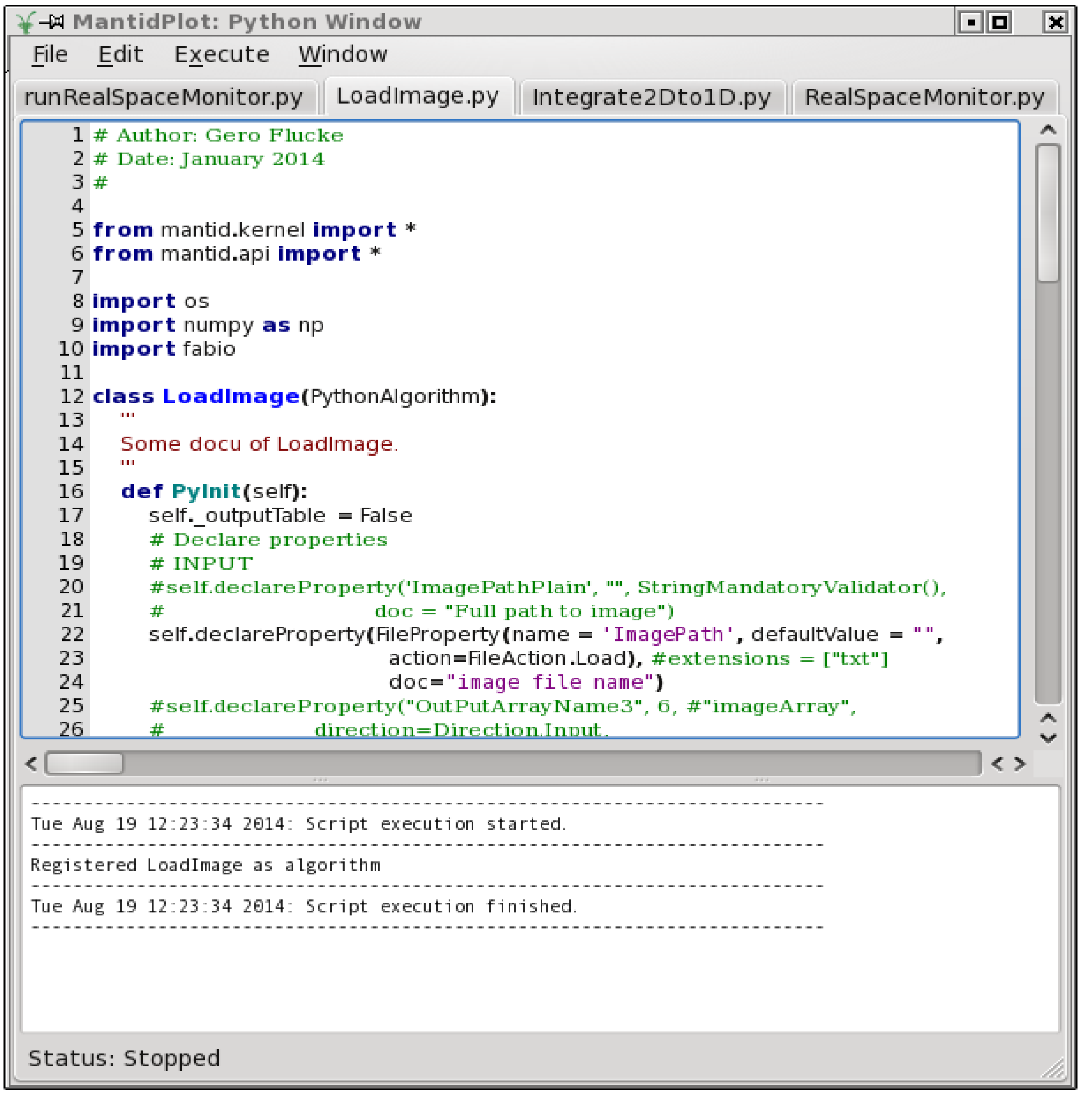
```
Logger
Processing Counter: 126
Processing Counter: 133
Processing Counter: 140
Processing Counter: 147
Stop processing.
```

Figure 60: DPDAK PDF online display

Figure 60 shows DPDAK's main window and the four configured display tools after analysing a full chain of 147 images. The 'Image Display' in the upper middle presents the last processed image. Whereas it is possible to interact with the display tools while processing, this is not advisable and slow. Processing can be stopped at any time via the stop button. Note that the integration region of 'SAXS Integration' is indicated with red lines in the image display. Having stopped processing, the display tools allow scrolling back through the data to visualise the results for each processed image, e.g. as done in the lower right 'Multi Plot' showing $S(q)$ and $G(r)$ of image 139. In the 'Multi Plot' in the lower middle, the distributions of the five shell positions versus image number are shown. These are not identical to the one obtained with DAWN (see Figure 58) due to slightly different settings and another integration algorithm, but they are pretty similar, especially the common jump around image 105. Unfortunately, the automatic positioning of the legend hides some data.

The DPDAK database contains all output of all processing plug-ins. This database can be saved via 'File -> Save Database' from the menu. Reloading it later restores also the plug-in configuration. For further processing of the data with other tools, the database can be exported as a whole or as the part of a single processing plug-in using one of the export plug-ins that create text files. The respective plug-ins can be selected via the 'Export' menu of the main DPDAK window.

6.3.3 Diffraction monitoring using Mantid



```
# Author: Gero Flucke
# Date: January 2014
#

from mantid.kernel import *
from mantid.api import *

import os
import numpy as np
import fabio

class LoadImage(PythonAlgorithm):
  '''
  Some docu of LoadImage.
  '''
  def PyInit(self):
    self._outputTable = False
    # Declare properties
    # INPUT
    #self.declareProperty('ImagePathPlain', "", StringMandatoryValidator(),
    #                 doc = "Full path to image")
    self.declareProperty(FileProperty(name = 'ImagePath', defaultValue = "",
                         action=FileAction.Load), #extensions = ["txt"]
                         doc="image file name")
    #self.declareProperty("OutPutArrayName3", 6, #"imageArray",
    #               direction=Direction.Input,
```

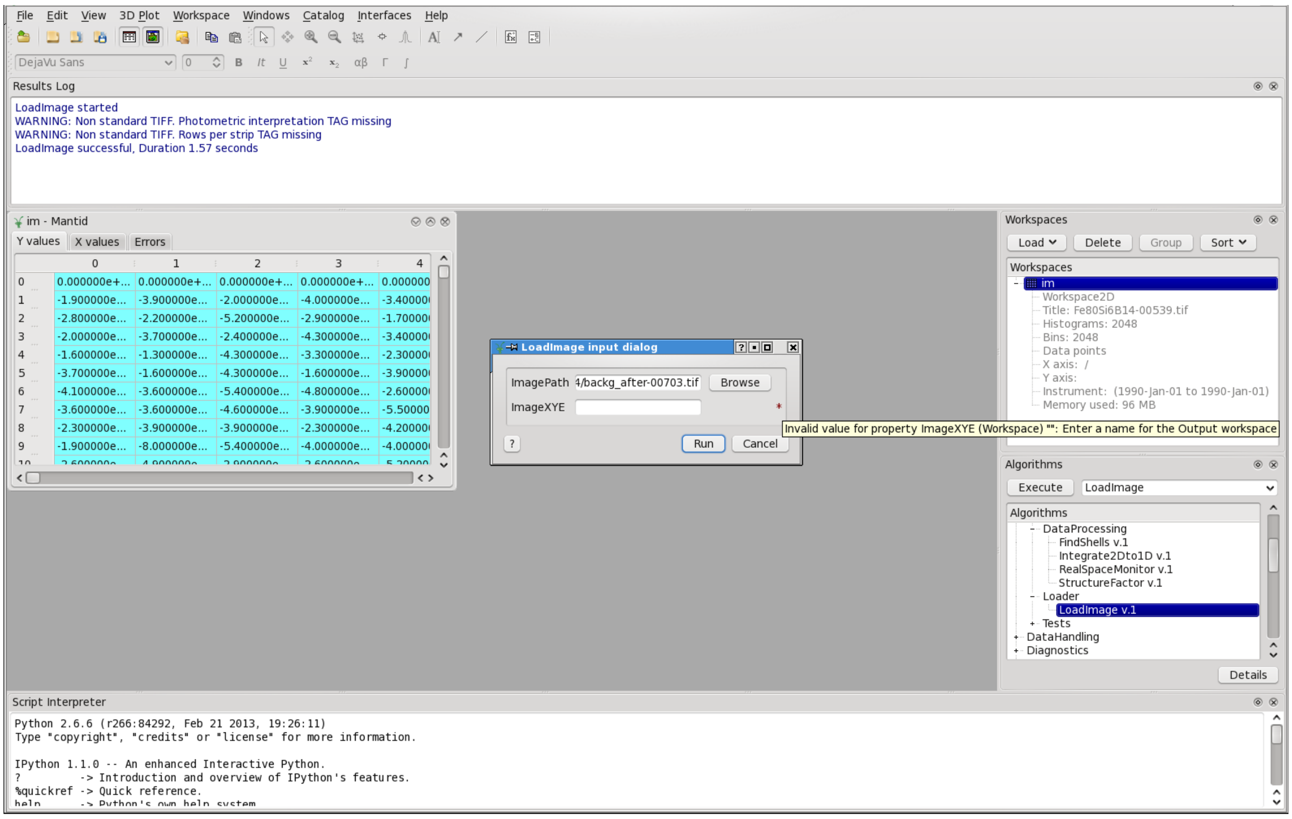
```
--------------------------------------------------------------------------------
Tue Aug 19 12:23:34 2014: Script execution started.
--------------------------------------------------------------------------------
Registered LoadImage as algorithm
--------------------------------------------------------------------------------
Tue Aug 19 12:23:34 2014: Script execution finished.
--------------------------------------------------------------------------------
```

Figure 61: Mantid's Python editor containing the LoadImage algorithm.

To implement the PDF monitoring procedure, Mantid's concept of algorithms working on workspaces has been used for all steps. The first hurdle to overcome is to get image data into a workspace. Although Mantid provides many algorithms to load files with different data formats, no algorithm exists to open an image file and to convert its intensity data into a workspace. Therefore a LoadImage algorithm had to be developed in Python.

This was done from within Mantid using its Python editor as shown in Figure 61. A Mantid algorithm written in Python is a class inheriting from Mantid's `PythonAlgorithm`. The algorithm class has to implement the method `PyInit` to declare its properties (i.e. its inputs, outputs or parameters) and the method `PyExec` to run the algorithm. Further methods are optional, e.g. `category` to assign the algorithm to a category to ease the navigation through the long list of available algorithms. The function call `AlgorithmFactory.subscribe(LoadImage)` makes the newly created class LoadImage known to Mantid. Once the file has been executed the algorithm is available in Mantid. After changing the code, re-execution is sufficient to make the changes available in the current Mantid session. The Mantid configuration defines a directory for all Python algorithms to be automatically loaded at program start.

Two kinds of workspaces can be created within Python algorithms. One kind is the table workspace that is like a spread sheet with various columns of possibly different data types. The other kind is a two dimensional matrix workspace containing rows with XYE data (see Mantid section 6.2.3). Both workspace types are not fully appropriate for a simple two dimensional dataset. The matrix workspace has been chosen as the LoadImage output type since it is the most commonly used workspace in Mantid. The intensity values are assigned to Y data whereas the X data is filled with horizontal pixel indices. The errors E are not used, i.e. stay zero. Since the image data has to be transferred to the workspace row-by-row in a Python loop, the algorithm is rather slow.



```
LoadImage started
WARNING: Non standard TIFF. Photometric interpretation TAG missing
WARNING: Non standard TIFF. Rows per strip TAG missing
LoadImage successful, Duration 1.57 seconds
```

	0	1	2	3	4
0	0.000000e+...	0.000000e+...	0.000000e+...	0.000000e+...	0.000000
1	-1.900000e...	-3.900000e...	-2.000000e...	-4.000000e...	-3.40000
2	-2.800000e...	-2.200000e...	-5.200000e...	-2.900000e...	-1.70000
3	-2.000000e...	-3.700000e...	-2.400000e...	-4.300000e...	-3.40000
4	-1.600000e...	-1.300000e...	-4.300000e...	-3.300000e...	-2.30000
5	-3.700000e...	-1.600000e...	-4.300000e...	-1.600000e...	-3.90000
6	-4.100000e...	-3.600000e...	-5.400000e...	-4.800000e...	-2.60000
7	-3.600000e...	-3.600000e...	-4.600000e...	-3.900000e...	-5.50000
8	-2.300000e...	-3.900000e...	-3.900000e...	-2.300000e...	-4.20000
9	-1.900000e...	-8.000000e...	-5.400000e...	-4.000000e...	-4.00000

ImagePath 4/backg_after-00703.tif
ImageXYE *

Invalid value for property ImageXYE (Workspace) "": Enter a name for the Output workspace

```
Workspaces
- im
    Workspace2D
    Title: Fe80Si6B14-00539.tif
    Histograms: 2048
    Bins: 2048
    Data points
    X axis:  /
    Y axis:
    Instrument:  (1990-Jan-01 to 1990-Jan-01)
    Memory used: 96 MB
```

```
Algorithms
  - DataProcessing
      FindShells v.1
      Integrate2Dto1D v.1
      RealSpaceMonitor v.1
      StructureFactor v.1
  - Loader
      LoadImage v.1
  + Tests
+ DataHandling
+ Diagnostics
```

```
Python 2.6.6 (r266:84292, Feb 21 2013, 19:26:11)
Type "copyright", "credits" or "license" for more information.

IPython 1.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
```

Figure 62: Mantid when working with the 'LoadImage' algorithm, but passing an invalid property.

Mantid reports between 1 s and 2 s for loading a tiff file with 2048x2048 pixels. Figure 62 shows Mantid when working with the new algorithm. It has been called from the list of algorithms in the 'Algorithms' sub-window on the right. The algorithm list is alphabetically ordered according to the algorithm category. The created workspace 'im' is listed in the 'Workspaces' sub-window on the right with some metadata: the image dimensions and, as workspace title, the name of the file. The matrix of the workspace, which is located on the left of the screen, has been created by dragging the workspace into the grey area.

To demonstrate Mantid's error handling LoadImage has been executed again (dialogue in the middle). The little red star on the right indicates an error because the run button has been pressed, although the 'ImageXYE' property field, that defines the name of the output workspace, was left empty. Moving the mouse over the red star generates a tooltip describing what has to be corrected. This mechanism to avoid starting an algorithm with invalid properties is very simple to implement in the `PyInit` method.

The following list of algorithms has been used for the monitoring procedure to process the data of each single image. Some of them exist already in Mantid, whereas the others had to be implemented.

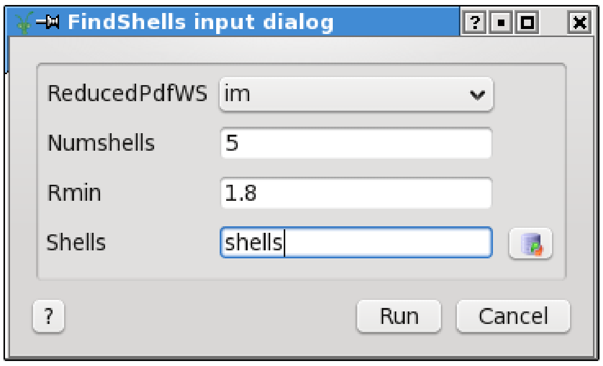
- The `LoadImage` algorithm was described above.
- The `Integrate2Dto1D` algorithm takes the image workspace as input and calculates the diffractogram $I(q)$, taking into account an image mask supplied with the geometry properties. The output is a matrix workspace with a single row. This algorithm makes use of `pyFAI` as well as the `calcDiffractogram` method from the DAWN monitoring procedure. Special care has to be taken to avoid the time consuming initialisation of `pyFAI`'s geometry class in each call to the `PyExec` method of the algorithm. The reason is that Mantid creates new instances of the algorithm class for each call to `PyExec`. That means that `pyFAI`'s geometry object must have class or module scope instead of being a simple instance attribute of the algorithm class.
- The `StructureFactor` algorithm is very similar to the StructureFactor plug-in developed for DPDAK using the `calcTotalStructureFactor` method. It receives two matrix workspaces produced by `Integrate2Dto1D`, one for the signal image and one for the empty beam background. The output is again a matrix workspace with one row for the structure factor $S(q)$.
- The `PDFFourierTransform` algorithm, already existing in Mantid, calculates the PDF $G(r)$ from $S(q)$.
- The `FindShells` algorithm uses the method `calcShellCentres` developed for DAWN to extract the first n shell positions from $G(r)$. The output is a matrix workspace with one row and n columns.

The main loop iterating over the images has been implemented as an algorithm called `RealSpaceMonitor`. It invokes the above mentioned sequence of algorithms for each image, it produces the graphics output and it stores the calculated curves $I(q)$, $S(q)$ and $G(r)$ as well as the shell positions.

Data is stored in an HDF5 file using the same Python methods as in the DAWN procedure. A genuine Mantid procedure would have been to convert the data into matrix workspaces with one row for each processed image and to call the `SaveNexus` algorithm for each of these workspaces creating four files instead of one.

The `RealSpaceMonitor` needs the name of a configuration file as an input. The file is interpreted using Python's `ConfigParser`. It contains all properties for the algorithms applied.

Like in the preceding examples, the `RealSpaceMonitor` has to detect newly created image files and read them. The corresponding Python code is very similar to the DAWN code (which was inspired by DPDAK) except that Mantid opens the image file with the `LoadImage` algorithm.


FindShells input dialog

ReducedPdfWS	im
Numshells	5
Rmin	1.8
Shells	shells

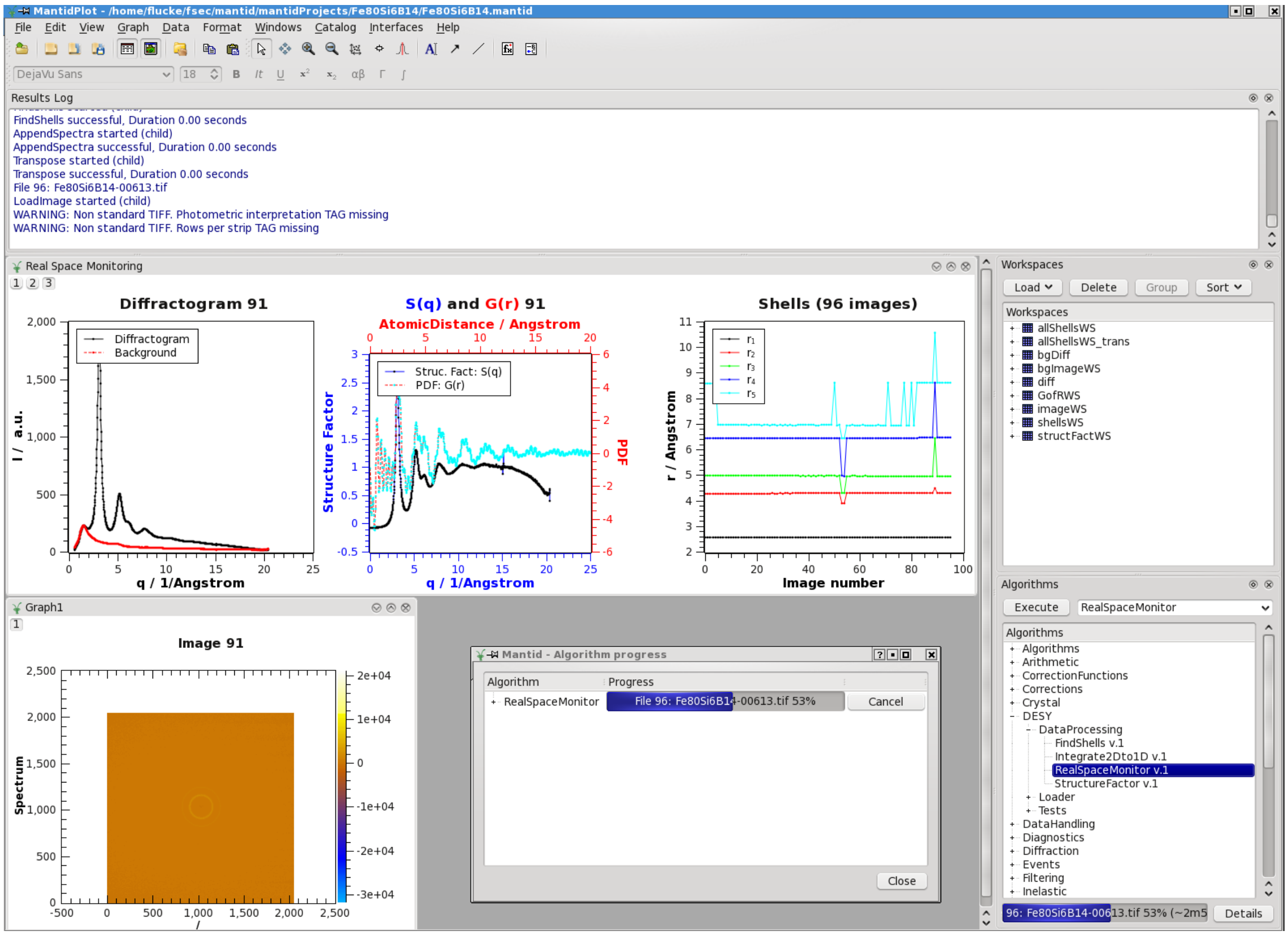
Figure 63: Mantid FindShells dialogue

It is worth mentioning that the Python interface to algorithms is very simple because Mantid wraps some code around the algorithm classes and exports them via a simple function interface. For example the `FindShells` algorithm, the Mantid dialogue is shown in Figure 63, is invoked by:

```
shellPosWS = FindShells(inputWS_GofR, Numshells=5, Rmin=1.8)
```

The function returns the first five shell positions in the reduced PDF $G(r)$, starting the search at $r = 1.8$ Å. The first argument is the Python variable containing the input workspace, followed by other parameters. To ease code readability, the Python keyword syntax has been applied. The function returns an output workspace.

The user must have the option to terminate the execution of the PDF online application. This feature is implemented by means of the Progress class. Instances of this class sense user initiated abort requests and invoke exit-functions that close HFD5 files and clean resources. The main purpose of this class, to display the execution progress, cannot be reasonably used because there is no way to estimate how many images have to be processed.



```
FindShells successful, Duration 0.00 seconds
AppendSpectra started (child)
AppendSpectra successful, Duration 0.00 seconds
Transpose started (child)
Transpose successful, Duration 0.00 seconds
File 96: Fe80Si6B14-00613.tif
LoadImage started (child)
WARNING: Non standard TIFF. Photometric interpretation TAG missing
WARNING: Non standard TIFF. Rows per strip TAG missing
```

Figure 64: Mantid, PDF online display

The PDF online display is shown in Figure 64. The ‘Results Log’ area in the top reports the output and the timing of the various algorithms. The ‘Workspaces’ area on the right shows the involved workspaces that are created by and passed between the algorithms. The central part shows the graphics window with three layers displaying the well-known curves. Images are displayed in the lower left part of the screen using generic functions to plot matrix workspaces. These functions provide less control of the graphical output, e.g. the axes labels are set to some defaults. The range of the colour scale is dominated by a few pixels. In principle this can be fixed by setting a proper range or choose a logarithmic scale, but this has not been attempted from Python. There is no Mantid function that displays a sequence of images consecutively in a static window. Instead the window has to be closed and re-opened for each new image.

Overall, the monitoring has been successfully implemented in Mantid, similar to what has been achieved with the other programs. Mainly due to the slow `LoadImage` algorithm, processing is a bit more time consuming. This hampers the use of the procedure for an offline analysis, running the procedure again and again while updating calibration and other parameters. But for online monitoring purposes at data rates of about 0.1 Hz, as in this science case, this does not pose any problem.

6.4 Visualisation and analysis of grazing incidence and small angle scattering data

Small angle scattering and grazing incidence small angle x-ray scattering experiments (GISAXS) involve typical data processing tasks: images are reduced to one-dimensional line cuts or converted to two-dimensional maps. The derived data are fitted for a quantitative analysis. Normalization and calibration steps are often applied.

Description of the science case

In a GISAXS experiment the growth of a gold layer on different surfaces is investigated in-situ. The X-ray scattering intensity in horizontal direction is examined as a function of time. Peak positions and widths, etc. are determined. The final goal (but beyond the scope of this report) is to compare models describing the growth of the gold surface with measured data. The experimental procedure has been as follows, similar to recent published results[54]:

- The substrate is placed in a sputtering chamber in such a way that the X-ray beam of wavelength $\lambda = 0.09445\ \mathrm{nm}$ hits its surface under a small angle of 0.5°.
- A 2D detector with pixels of size 172*172 μm^2 is placed at a distance of $1836.2\ \mathrm{mm}$ from the sample.
- During data taking 487*619 detector pixels are read-out at a frequency of 10 Hz

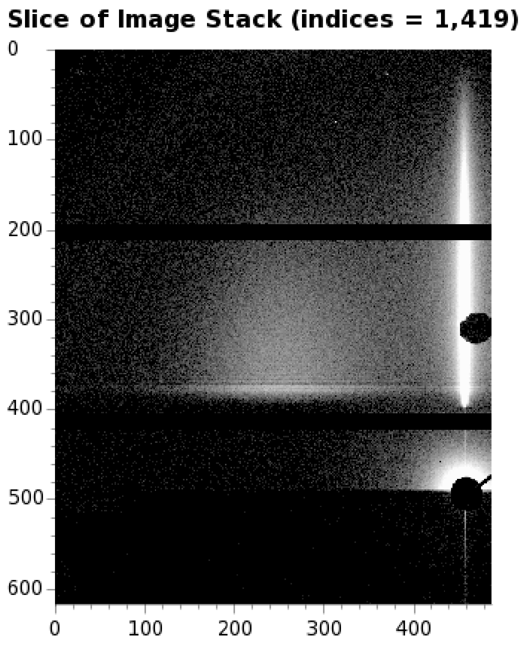


Figure 65: GISAXS image

[54] Hernandez, C. Domingo, and S. V. Roth, *Appl. Phys. Lett.* **104**, 243107 (2014).

S. Yu, G. Santoro, K. Sarkar, B. Dicke, P. Wessels, S. Bommel, R. Döhrmann, J. Perlich, M. Kuhlmann, E. Metwalli, J. F. H. Risch, M. Schwartzkopf, M. Drescher, P. Müller-Buschbaum and S. V. Roth, *J. Phys. Chem. Lett.* **4**, 3170 (2013).

M. Schwartzkopf, A. Buffet, V. Körstgens, E. Metwalli, Kai Schlage, G. Benecke, J. Perlich, M. Rawolle, A. Rothkirch, B. Heidmann, G. Herzog, P. Müller-Buschbaum, R. Röhlsberger, R. Gehrke, N. Stribeck and S. V. Roth, *Nanoscale* **5**, 5053 (2013).

and stored in the tif format.

- Soon after the start of data taking, the sputtering of gold atoms is started simultaneously with opening the beam shutter to illuminate the sample.
- Data taking is continued for about 1000 s, i.e. until 10k images are recorded.
- The procedure is then repeated for several samples with different surfaces.

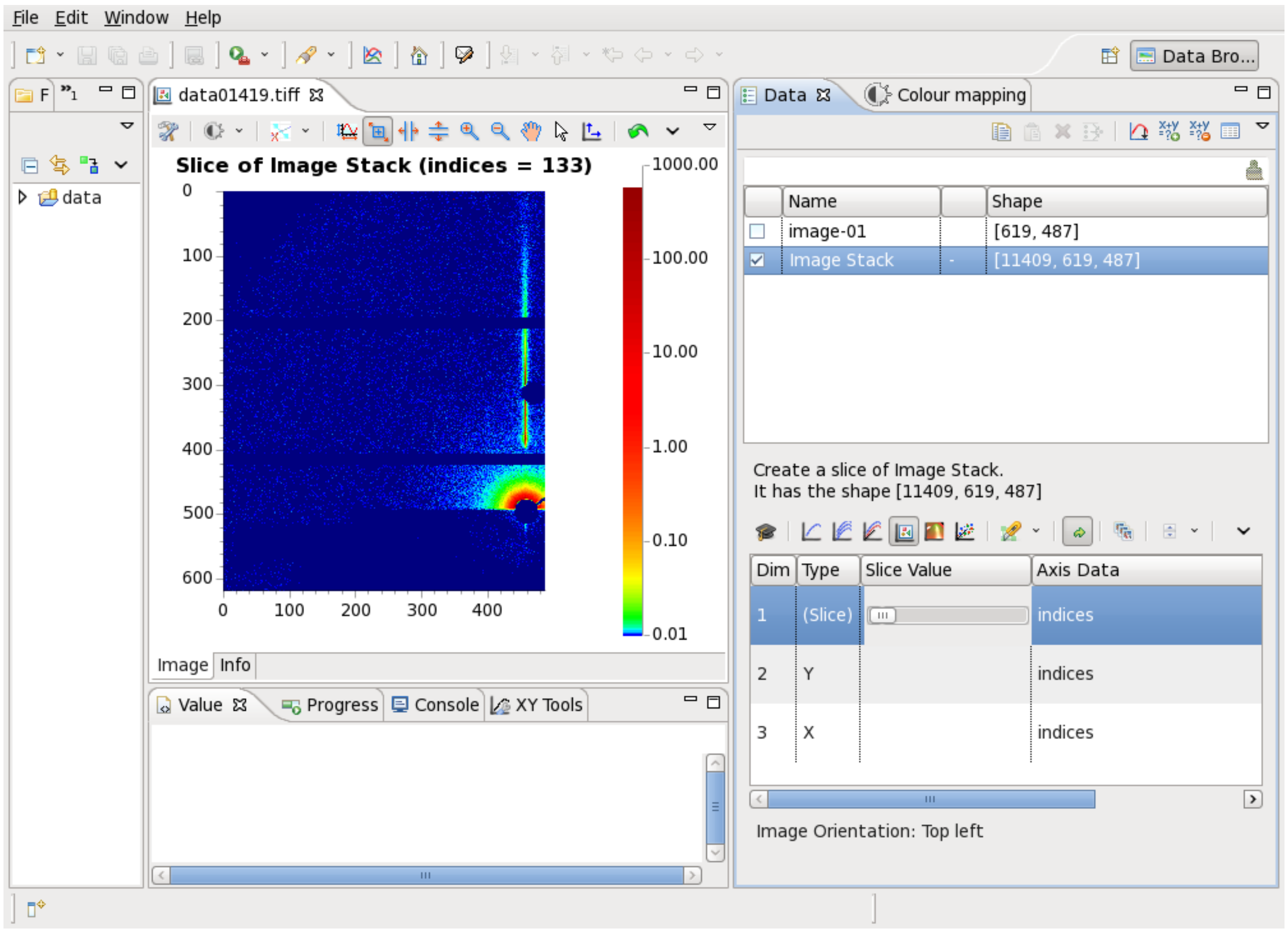
Figure 65 shows one of the images recorded. The direct beam aiming at the detector position around pixel 450 in horizontal and 490 in vertical direction is blocked by a beam stop. Similarly, the specular peak at about pixel 320 in vertical direction is blocked. Of major interest is the slightly enhanced intensity at about pixel 375 vertically and 250 horizontally. The position of this broad peak moves to the right with increasing sputtering time and the shape and position of the peak depend on the way the sputtered gold atoms organise on the surface. To improve the understanding of this kinetics of growth is the goal of the analysis. The analysis steps are as follows:

- An image is opened and a background image is subtracted.
- With geometrical information of the detector and the X-ray wavelength, axis indices are converted to momentum transfer in horizontal (q_y) and vertical (q_z) direction. The pixel position of the direct beam on the detector has been measured previously using an attenuated beam without beam stop.
- A horizontal slice extending the full q_y range and covering few pixels in the q_z direction is extracted at q_z matching the Yoneda peak at the critical angle of the substrate.
- The intensities of the pixels in q_z direction are summed, resulting in a diffractogram $I(q_y)$ that shows two or three peaks and some background.
- A functional description of $I(q_y)$ has to be found and its parameters to be determined by a fit.
- The fit parameters like position, width and normalisation of the peaks are visualised as a function of the image number and thus of time or of the average gold layer thickness, respectively.
- These distributions for the different substrates are compared to each other and to model predictions.

6.4.1 Small angle scattering analysis with DAWN

DAWN provides many options to quickly investigate the data interactively to get a first impression. A full processing of the data may require more flexibility as provided by scripting. Therefore, GISAXS data are first analysed interactively and then processed by Python scripts.

6.4.1.1 Interactive analysis with the DAWN GUI



	Name		Shape
☐	image-01		[619, 487]
☑	Image Stack	-	[11409, 619, 487]

Create a slice of Image Stack.
It has the shape [11409, 619, 487]

Dim	Type	Slice Value	Axis Data
1	(Slice)		indices
2	Y		indices
3	X		indices

Image Orientation: Top left

Figure 66: DAWN GISAXS image viewed as part of an Image Stack

DAWN's Data Browsing and DExplore perspectives are convenient to investigate multi-dimensional data. A sequence of image files with names like data00000.tiff to data11408.tiff is recognised as a three dimensional dataset if any of its files is opened. After the data is loaded the user may choose to view the images as an “Image Stack”. If this option is selected, it is possible to scroll through the images with a slider (Figure 66).

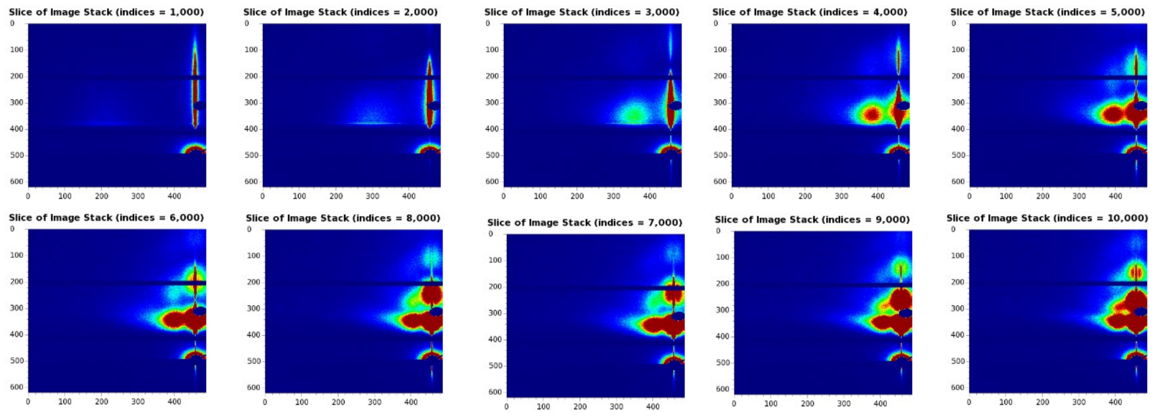
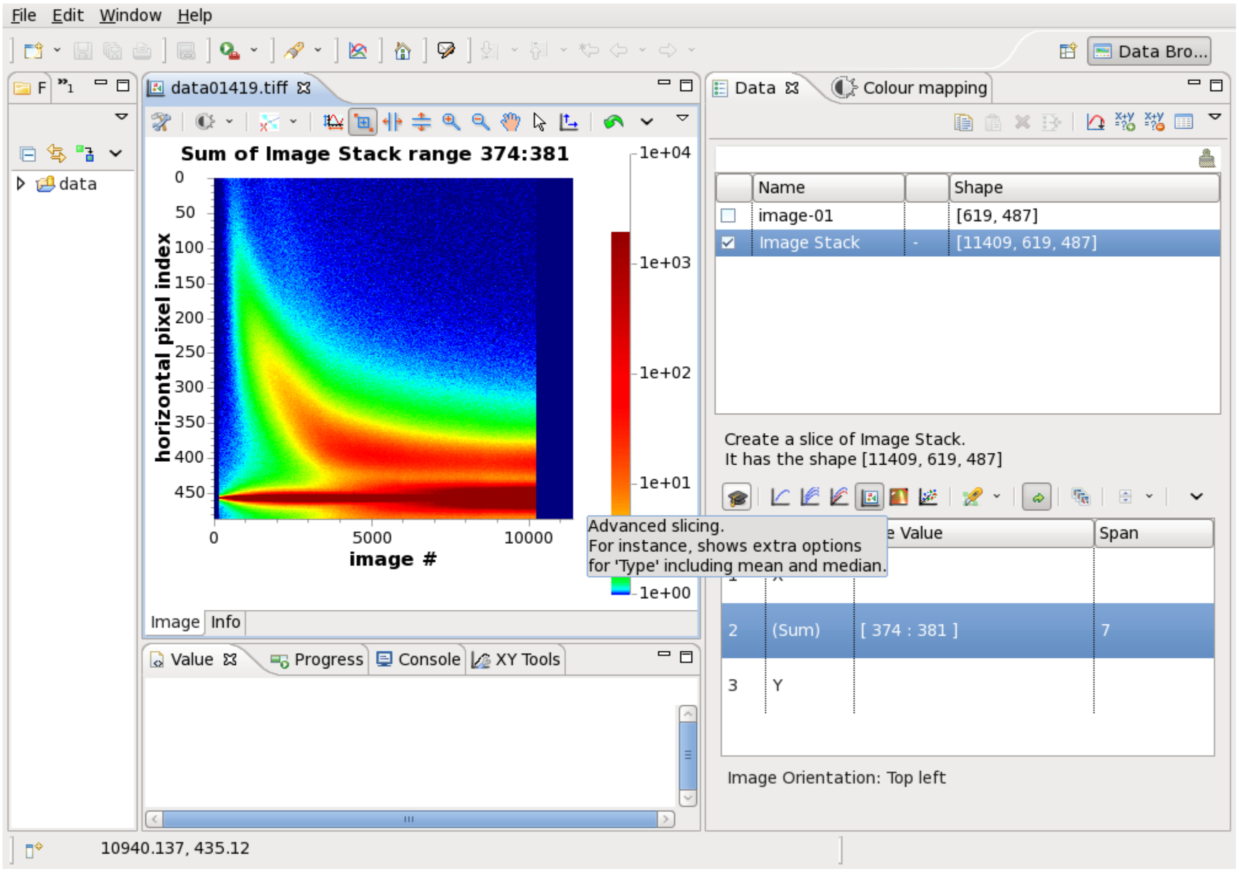


Figure 67: DAWN scrolling through an image stack

To locate the Yoneda peak and to investigate what is going on there in horizontal direction, the intensity range is set to 0-200 and locked between image updates for new slices. This allows scrolling back and forth through the images; some of them are shown in Figure 67. Zooming into the relevant image region and adjusting the colour mapping helps to identify the relevant vertical slice region as indices 374-380.



	Name		Shape
☐	image-01		[619, 487]
✓	Image Stack	-	[11409, 619, 487]

Create a slice of Image Stack.
It has the shape [11409, 619, 487]

Advanced slicing.
For instance, shows extra options for 'Type' including mean and median.

		Value	Span
2	(Sum)	[374 : 381]	7
3	Y		

Figure 68: DAWN advanced slicing perpendicular to the image plane.

Data can also investigated by cutting perpendicular through the image stack. A typical use case is to assign the image number to the x-axis and the horizontal image axis to y-axis. Slicing is performed in vertical direction of the image. Furthermore, the 'Advanced slicing' option allows using the mean, median, sum, maximum or minimum of an index range instead of a simple slice. For Figure 68 the sum of the indices between 374 and 380 (vertical image direction) has been selected. The axis labels have been added by hand using the 'Configure settings' button. Of course, this slice requires all image files to be opened which takes a while for the 11409 images.

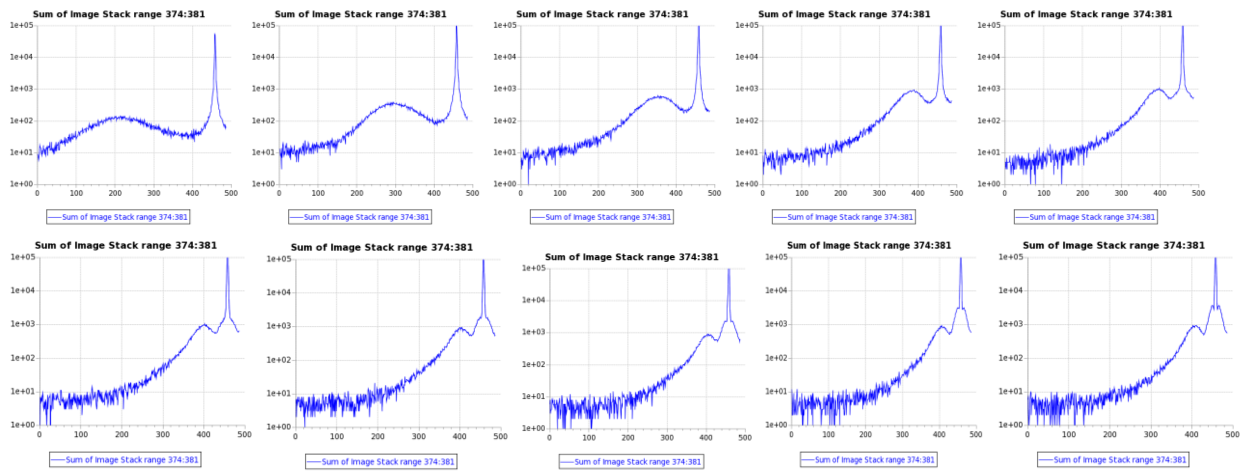
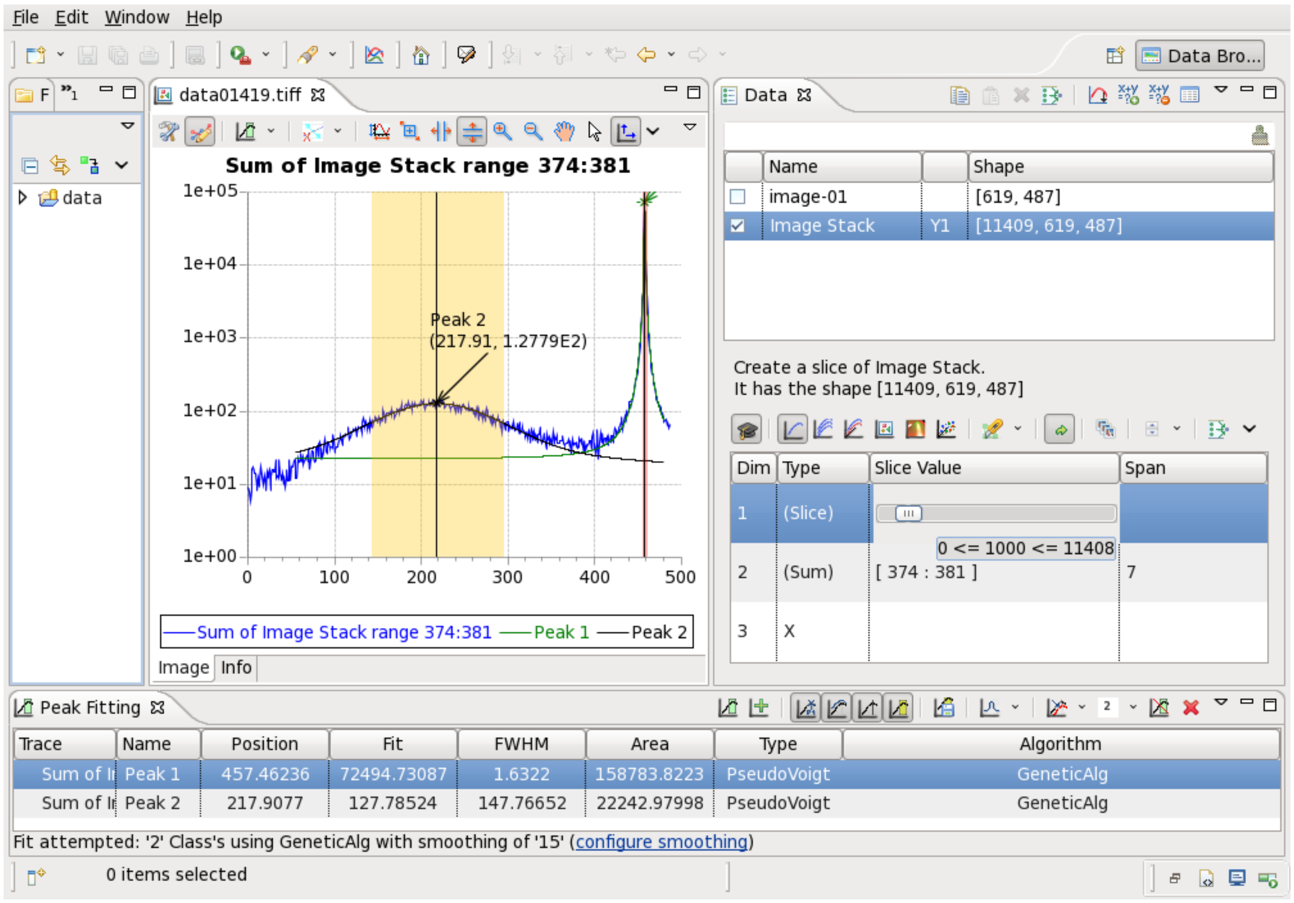


Figure 69: Intensity distributions for the sum of the vertical indices 374-380 as a function of the horizontal index for images 1000 (upper left) to 10000 (lower right).

The image stack can also be converted to 1-dimensional intensity distributions (Figure 69). The x-axis is identical to the x-axis of the images. The y-axis represents the intensities summed over the index range between 374 and 380 in vertical direction. Each distribution corresponds to a specific sputtering time. A strong peak is located near index 455, stable in time. In addition, there is another peak which is much broader and has a much lower intensity. Its position moves in time from about index 200 towards the main peak. Furthermore, the main peak develops a kind of pedestal that finally evolves to become two side peaks at the left and right of the main peak.



	Name		Shape
☐	image-01		[619, 487]
✓	Image Stack	Y1	[11409, 619, 487]

Create a slice of Image Stack.
It has the shape [11409, 619, 487]

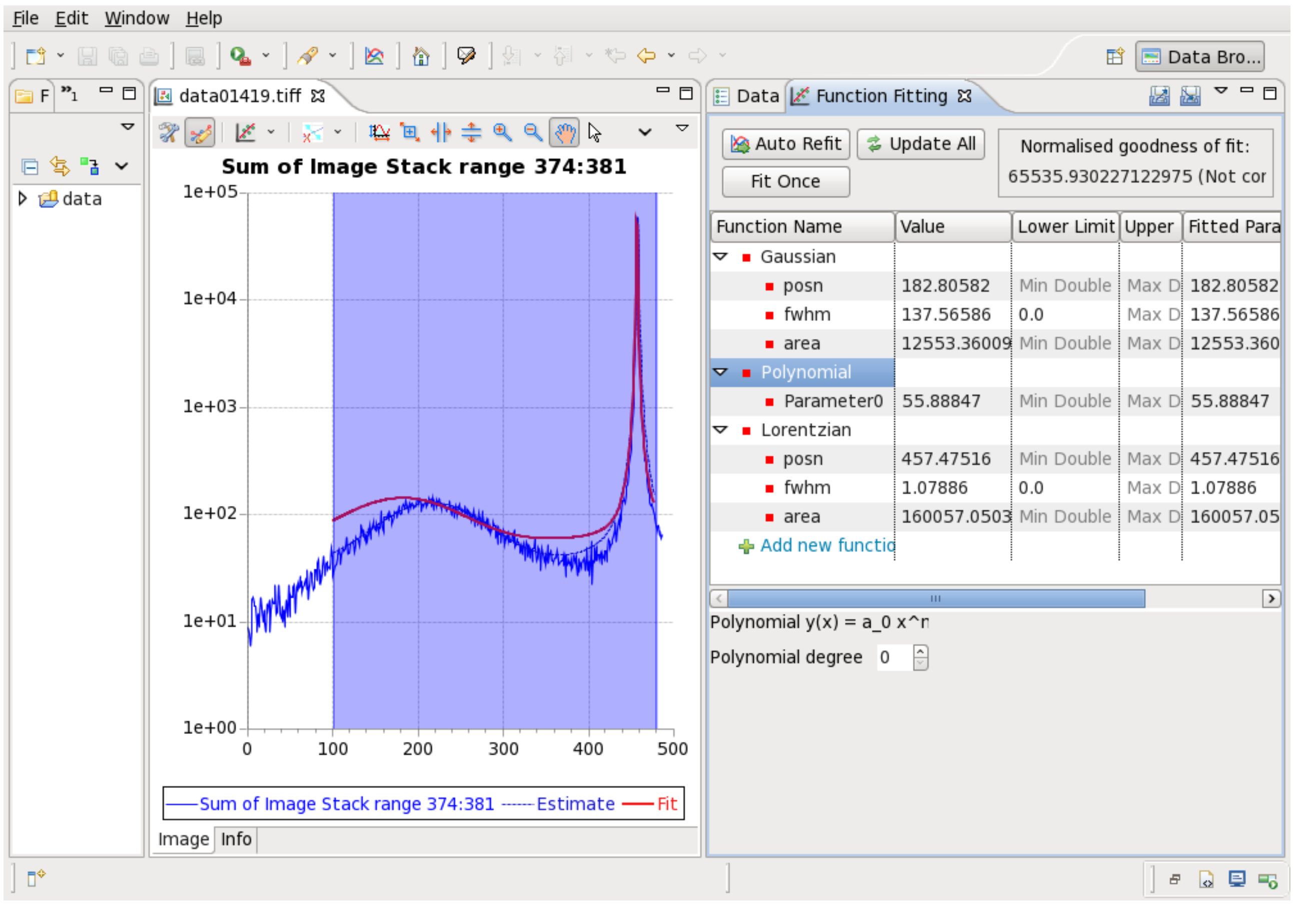
Dim	Type	Slice Value	Span
1	(Slice)	0 <= 1000 <= 11408	
2	(Sum)	[374 : 381]	7
3	X		

Trace	Name	Position	Fit	FWHM	Area	Type	Algorithm
Sum of Ir	Peak 1	457.46236	72494.73087	1.6322	158783.8223	PseudoVoigt	GeneticAlg
Sum of Ir	Peak 2	217.9077	127.78524	147.76652	22242.97998	PseudoVoigt	GeneticAlg

Fit attempted: '2' Class's using GeneticAlg with smoothing of '15' (configure smoothing)

Figure 70: DAWN peak fitting tool applied a slice of an image

While all this behaviour can already be seen in Figure 68, it is still rather qualitative. In a first attempt to get more quantitative results, the ‘Peak Fitting’ tool can be used. The peak type is set to PseudoVoigt and the maximum number of peaks to two. The fitting range is set graphically to cover almost the full range. Since the small peak is rather noisy, the smoothing option has to be set to about 15. Figure 70 shows the result for image 1000. Vertical lines annotate the peak positions and the shaded areas indicate their FWHMs. But both peaks are fitted independently with constant background terms.



Normalised goodness of fit:
65535.930227122975 (Not cor

Function Name	Value	Lower Limit	Upper	Fitted Para
Gaussian				
posn	182.80582	Min Double	Max D	182.80582
fwhm	137.56586	0.0	Max D	137.56586
area	12553.36009	Min Double	Max D	12553.360
Polynomial				
Parameter0	55.88847	Min Double	Max D	55.88847
Lorentzian				
posn	457.47516	Min Double	Max D	457.47516
fwhm	1.07886	0.0	Max D	1.07886
area	160057.0503	Min Double	Max D	160057.05

Polynomial y(x) = a_0 x^n

Polynomial degree 0

Figure 71: Function fitting using the sum of a Gaussian, a Lorentzian and a constant background to the blue shaded fit range.

The Function Fitting tool provides flexible ways to define a common function fitted to data distributions. After starting the tool, the fit range is graphically defined to be around the small peak by moving the edges of a blue shaded area. The fit function is first composed of a Gaussian and a Polynomial of degree 0 (i.e. a constant). Start values for the Gaussian parameters are given close to the results of the previous peak fit approach. After the fitting procedure is executed, the fit range is extended to include the main peak and a Lorentzian is added to the fit function. Its start parameters are set to the corresponding values of the peak fit. The Gaussian start parameters are kept by DAWN as the result of the fit in reduced range. The result of the fit can be found in Figure 71 along with the function estimate using the start values: The fit has not well converged, as (partially) indicated in the upper right corner of the 'Function Fitting' view, but that could probably be overcome by adjusting the fitting function, the start values and the fitting options.

While interactive analysis with the DAWN GUI provides a lot of insight into the investigated data, several issues cannot be covered:

- All axis units need to be converted from indices to momentum transfer. This would work in the GUI if the image stack would be converted to a 3D dataset in an HDF5 file and if the correct axis data would be stored in the same file.
- Subtraction of a background image is missing.
- The two little side peaks next to the main peak are symmetric around it. Currently the function fitting tool is not flexible enough to treat that correctly. In principle one could extend DAWN by an appropriate plug-in containing the desired function, but this expert task lacks precise documentation.

- Finally, the fit has to be applied to all images and the fit parameters stored, but currently the function fitting tool cannot be used for such data reduction.

Therefore, the analysis is continued using DAWN's scripting language Python.

6.4.1.2 Python Scripting with DAWN

For the Python analysis a new PyDev project has been created, using the name 'saxs'. It is configured to make use of Python modules in a utility project containing methods and classes of general use.

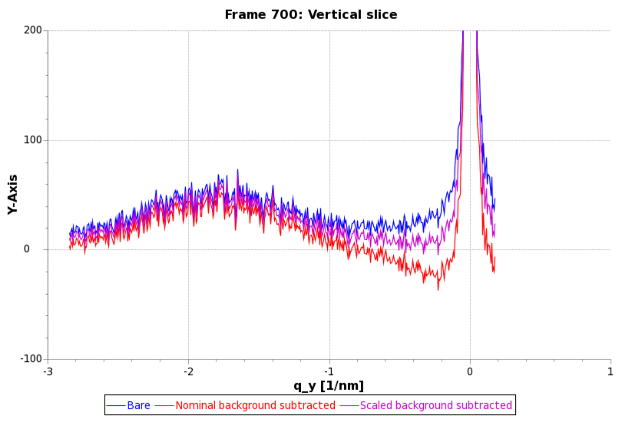
Since the handling of a single HDF5 file containing many images is simpler than dealing with many .tiff files, the latter have been converted. In principle the conversion to HDF5 can be done using DAWN's data conversion wizard. However, this takes quite a long time (25 min for 14 GB) and the data are not compressed. Therefore the data have been converted and compressed by IDL generating a file of 2.4 GB. This file can also be used for the interactive analysis as described in the preceding section.

An analysis class, called `GisaxsSputter`, has been written with a master method to control the loop over the image frames. For each processed frame, the vertical slice is extracted as a diffractogram, taking into account the background. Then a method is called that fits a model function to the diffractogram after defining the fit range and finding appropriate start values. Eventually the fit results are stored in an HDF5 file. Auxiliary code has been added to display data and fitted function for a subset of the processed images and to display the development of the fitted parameters as a function of time. Care has to be taken that the procedure continues to process the remaining images in case a fit failed.

The image files are opened using DAWN's `scisoftpy.io` module. The numpy syntax is applicable to the dataset class of `scisoftpy.io`. It is well suited to extract a vertical slice of a single image out of the 3D stack. If the order of the dimensions is frame, vertical, horizontal, then

```
verticalSlices = dataset[1234, 374:381, :] # 7 vertical slices
verticalSlice = verticalSlices.sum(axis=0) # sum vertically
```

returns the horizontal slice for frame 1234, summing over the vertical indices 374 to 380. The horizontal pixel centres p_h are converted to the horizontal momentum transfer $q_y = \frac{4\pi}{\lambda} * \sin\left(\frac{(p_h - b_h)*s}{2\,d}\right) * \cos\left(\frac{(p_v - b_v)*s}{d} - \alpha_i\right)$ using the horizontal (vertical) position of direct beam b_h (b_v), measured in pixels, the pixel size s, the wavelength λ, the sample detector tance d, the central vertical pixel position p_v (i.e. 377.5 for the slice above) and the incident gle α_i.



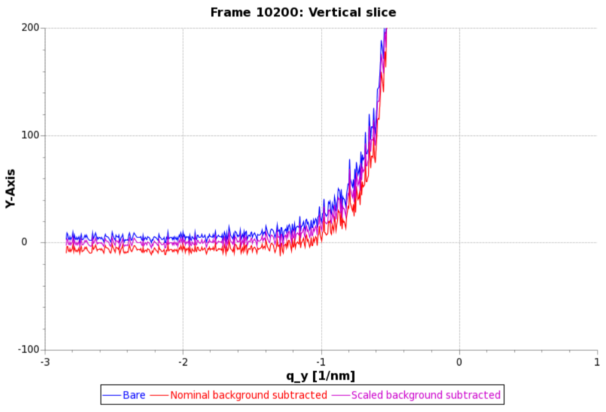


Figure 72: Example vertical slices without, with nominal and with scaled background subtracted.

The background image has been recorded with the same setup as the other data, but without a sample in the beam. It has been illuminated for 10 s to reduce statistical noise and thus nominally has to be scaled by a factor 0.01. Example slices are shown in Figure 72, before and after the nominal background subtraction. The corrected intensities are partially below 0 indicating that the background is overestimated (red lines). To take this into account a scaling factor for the background has been determined by comparing the background image with regions of real images that are expected to be signal-free. This is the procedure: for a given q_y range (≈ -2.5 to -1.5 1/nm) the sums of intensities were calculated for the background image and for one of the last images in the sequence (signal-free) The new scaling factor is the ratio of the sums: 0.00428. This value is almost independent of the q_y range and the selected real image. Figure 72 shows that the new scaling factor produces a better estimate of the background (pink lines). Along with the corrected intensities, their errors are computed to be used as input for the following fit procedure. The computation assumes Poisson statistics for the intensities and implements standard error propagation for the background subtraction.

The development of a fit function that can properly be applied to the changing form of the vertical slices for all images has been a major part of this analysis. This includes the estimation of suitable start values for the function parameters and the definition of the fit range. Since the DAWN fitting routines are not available in Python, the method `optimize.curve_fit` from the Scipy[55] package has been employed. Various functional forms could be tested thanks to the flexibility of `curve_fit` to accept any user defined Python function. Unfortunately `curve_fit` does not allow fixing the function parameters. Therefore a wrapper has been developed. Since the wrapper and all tested functions might be re-used in other analyses, the Python project `xrayutils` has been created in DAWN as a container for these tools. The final form of the fit function is illustrated in Figure 73 and its functional form is

$$\begin{aligned}
&I(q_y, \mu_m, \sigma_m, N_m, \delta_s, w_s, N_s, \delta_o, \sigma_o, N_o) = \\
&+G(q_y, \mu_m, \sigma_m, N_m) \\
&+L(q_y, \mu_m + \delta_s, w_s, N_s) + L(q_y, \mu_m - \delta_s, w_s, N_s) \\
&+G(q_y, \mu_m + \delta_o, \sigma_o, N_o) + G(q_y, \mu_m - \delta_o, \sigma_o, N_o).
\end{aligned}$$

[55] http://scipy.org/scipylib/

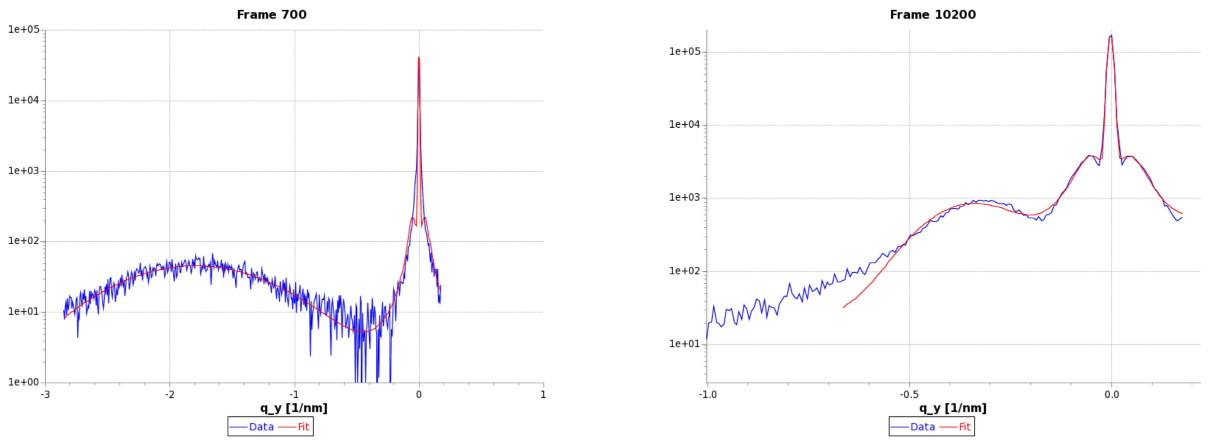
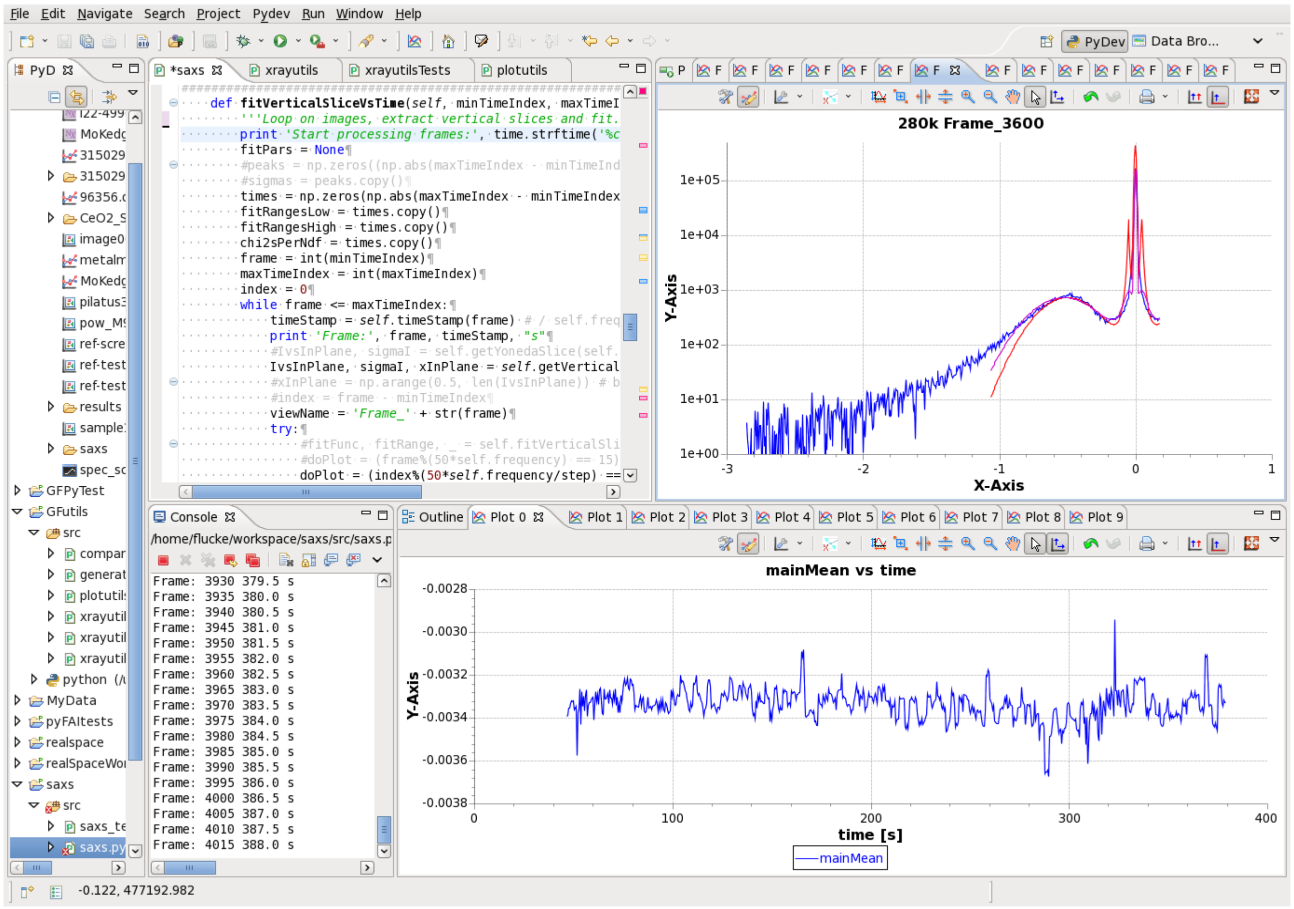


Figure 73: Data and fitted functions for an early and a late diffractogram. Note the zoomed q_y range on the right.

The fit function has a central Gaussian G (for the main peak) with the parameters mean μ_m, standard deviation σ_m and normalisation N_m, two Lorentzians L symmetrically around the main peak (for side peaks that appear for higher image indices) with peak distances δ_s from the main peak, FWHM w_s and normalisation N_s, and two similarly symmetric Gaussians (for the outer broad peak of low intensity moving towards the main peak) with peak distances δ_o, standard deviation σ_o and normalisation N_o. In practice, the latter would not have to be symmetric since the q_y range in the data covers only one side here, but it would be needed if a broader detector would be used with the direct beam in the centre.

Start values are defined, besides some details, in the following way. First, the outer peak is considered. In the range of the 425 most left intensity values (i.e. outside the range of the main and side peaks), the maximum value and its position are found. The FWHM is estimated in the usual way. Due to low statistics, there is relevant noise and the data have to be smoothed before. Then a simple Gaussian is fitted by using these values as start points. The fit is repeated in a subrange 2.5 σ around the peak. Subsequently the position, width and normalisation of the main peak are determined in a similar procedure (but without smoothing) in the full range. The side peaks are estimated to have twice the width and about 10% of the height of the main peak. Their positions are first estimated to be about δ_s=0.05 nm^{-1} away from the main peak and this distance is fixed, as discussed below. The final fit range starts at the left end of the fit range for the outer peak described above and then covers the full data range to the right. As one can see from Figure 73 (left part), the diffractogram from shortly after start of the sputtering does not yet show the side peaks, but they help to improve the description by partially mimicking the pedestal below the main peak. On the other hand, these side peaks are prominent in the later diffractogram.



```
########################################################################
    def fitVerticalSliceVsTime(self, minTimeIndex, maxTimeI
        '''Loop on images, extract vertical slices and fit.
        print 'Start processing frames:', time.strftime('%c
        fitPars = None
        #peaks = np.zeros((np.abs(maxTimeIndex - minTimeInd
        #sigmas = peaks.copy()
        times = np.zeros(np.abs(maxTimeIndex - minTimeIndex
        fitRangesLow = times.copy()
        fitRangesHigh = times.copy()
        chi2sPerNdf = times.copy()
        frame = int(minTimeIndex)
        maxTimeIndex = int(maxTimeIndex)
        index = 0
        while frame <= maxTimeIndex:
            timeStamp = self.timeStamp(frame) # / self.freq
            print 'Frame:', frame, timeStamp, "s"
            #IvsInPlane, sigmaI = self.getYonedaSlice(self.
            IvsInPlane, sigmaI, xInPlane = self.getVertical
            #xInPlane = np.arange(0.5, len(IvsInPlane)) # b
            #index = frame - minTimeIndex
            viewName = 'Frame_' + str(frame)
            try:
                #fitFunc, fitRange, _ = self.fitVerticalSli
                #doPlot = (frame%(50*self.frequency) == 15)
                doPlot = (index%(50*self.frequency/step) ==
```

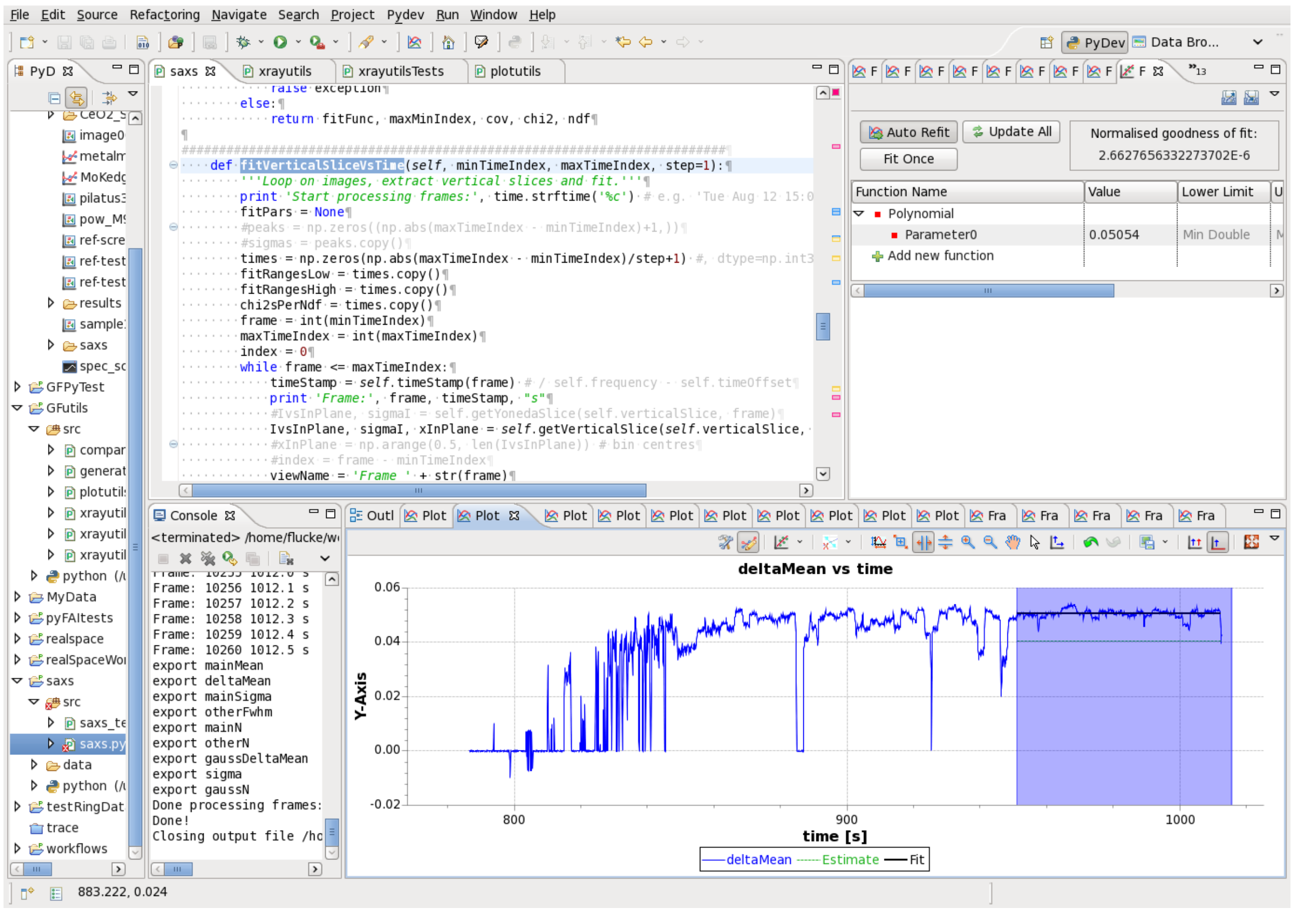
/home/flucke/workspace/saxs/src/saxs.p

```
Frame: 3930 379.5 s
Frame: 3935 380.0 s
Frame: 3940 380.5 s
Frame: 3945 381.0 s
Frame: 3950 381.5 s
Frame: 3955 382.0 s
Frame: 3960 382.5 s
Frame: 3965 383.0 s
Frame: 3970 383.5 s
Frame: 3975 384.0 s
Frame: 3980 384.5 s
Frame: 3985 385.0 s
Frame: 3990 385.5 s
Frame: 3995 386.0 s
Frame: 4000 386.5 s
Frame: 4005 387.0 s
Frame: 4010 387.5 s
Frame: 4015 388.0 s
```

Figure 74: DAWN monitoring the fit procedure

The execution of the master loop is monitored by plotting the evolution of the fitted parameter values as a function of time. Also a subset of the diffractograms is plotted, overlaid with the fit results. An example DAWN screen is displayed in Figure 74. While the Python script is active, the plot views can be investigated by the usual DAWN tools, e.g. zooming, fitting, etc.

The evolution of μ_m as a function of time after sputtering starts, shown in the lower right of Figure 74, is rather constant, but has a clear offset from 0. This indicates that the horizontal position of the direct beam on the detector b_h was not precisely measured. To improve that, the procedure is re-run without the conversion of the horizontal indices to q_y values. The procedure described above to find appropriate start parameters of the fit function can automatically deal with the change of scale, except that δ_s has to be adjusted by hand to 7.88. The resulting μ_m values for all considered diffractograms are averaged by fitting a constant to the plotted $\mu_m(t)$ with DAWN's function fitting tool. This is repeated for the different substrate samples since it turns out that the offset slightly differs between them.


```
            raise exception
        else:
            return fitFunc, maxMinIndex, cov, chi2, ndf

#############################################################################
    def fitVerticalSliceVsTime(self, minTimeIndex, maxTimeIndex, step=1):
        '''Loop on images, extract vertical slices and fit.'''
        print 'Start processing frames:', time.strftime('%c') # e.g. 'Tue Aug 12 15:0
        fitPars = None
        #peaks = np.zeros((np.abs(maxTimeIndex - minTimeIndex)+1,))
        #sigmas = peaks.copy()
        times = np.zeros(np.abs(maxTimeIndex - minTimeIndex)/step+1) #, dtype=np.int3
        fitRangesLow = times.copy()
        fitRangesHigh = times.copy()
        chi2sPerNdf = times.copy()
        frame = int(minTimeIndex)
        maxTimeIndex = int(maxTimeIndex)
        index = 0
        while frame <= maxTimeIndex:
            timeStamp = self.timeStamp(frame) # / self.frequency - self.timeOffset
            print 'Frame:', frame, timeStamp, "s"
            #IvsInPlane, sigmaI = self.getYonedaSlice(self.verticalSlice, frame)
            IvsInPlane, sigmaI, xInPlane = self.getVerticalSlice(self.verticalSlice, 
            #xInPlane = np.arange(0.5, len(IvsInPlane)) # bin centres
            #index = frame - minTimeIndex
            viewName = 'Frame ' + str(frame)
```

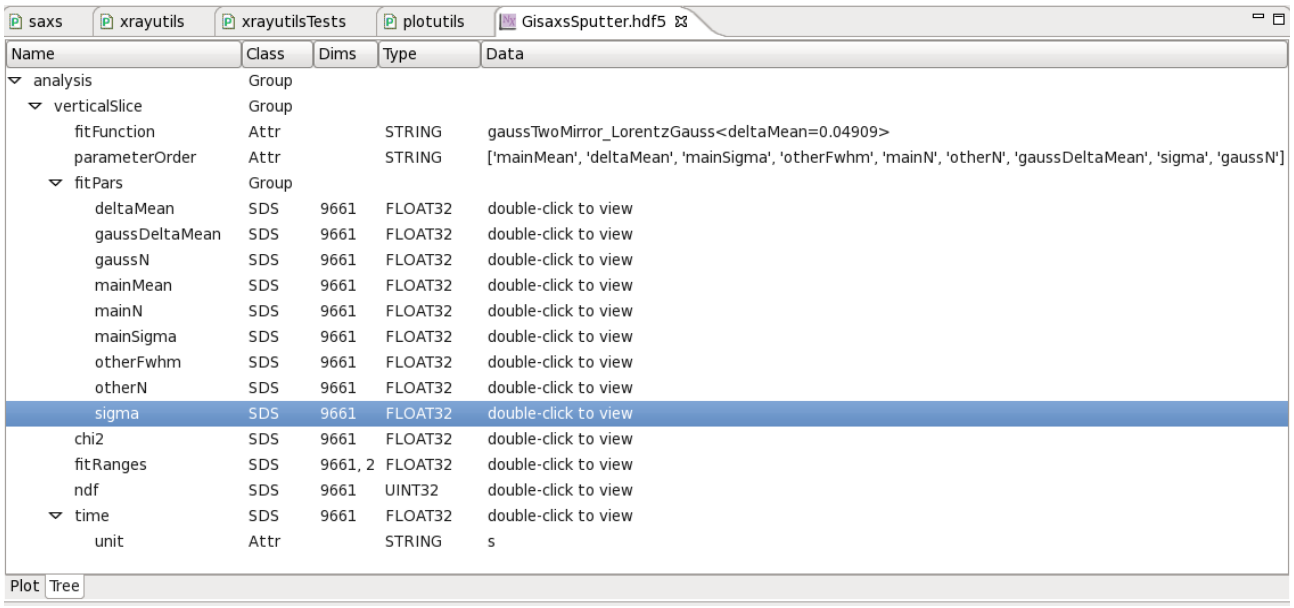
```
<terminated> /home/flucke/w
Frame:  10256 1012.1 s
Frame:  10257 1012.2 s
Frame:  10258 1012.3 s
Frame:  10259 1012.4 s
Frame:  10260 1012.5 s
export mainMean
export deltaMean
export mainSigma
export otherFwhm
export mainN
export otherN
export gaussDeltaMean
export sigma
export gaussN
Done processing frames:
Done!
Closing output file /ho
```

Normalised goodness of fit: 2.6627656332273702E-6

Function Name	Value	Lower Limit
Polynomial		
Parameter0	0.05054	Min Double
Add new function		

Figure 75: Determination of the side peak position δ_s using the function fitting tool for a subrange of the data.

The last step before the complete data are finally processed is to re-evaluate the fixed parameter δ_s. To do so, the fitting procedure is applied to those diffractograms that show the side peaks. This time the parameter is not fixed. Figure 75 shows the resulting time dependence of δ_s and how the function fitting tool is again used to fit a constant value by restricting the fit to a range where the side peaks are clearly visible and where δ_s shows almost constant behaviour.



Name	Class	Dims	Type	Data
analysis	Group			
verticalSlice	Group			
fitFunction	Attr		STRING	gaussTwoMirror_LorentzGauss<deltaMean=0.04909>
parameterOrder	Attr		STRING	['mainMean', 'deltaMean', 'mainSigma', 'otherFwhm', 'mainN', 'otherN', 'gaussDeltaMean', 'sigma', 'gaussN']
fitPars	Group			
deltaMean	SDS	9661	FLOAT32	double-click to view
gaussDeltaMean	SDS	9661	FLOAT32	double-click to view
gaussN	SDS	9661	FLOAT32	double-click to view
mainMean	SDS	9661	FLOAT32	double-click to view
mainN	SDS	9661	FLOAT32	double-click to view
mainSigma	SDS	9661	FLOAT32	double-click to view
otherFwhm	SDS	9661	FLOAT32	double-click to view
otherN	SDS	9661	FLOAT32	double-click to view
sigma	SDS	9661	FLOAT32	double-click to view
chi2	SDS	9661	FLOAT32	double-click to view
fitRanges	SDS	9661, 2	FLOAT32	double-click to view
ndf	SDS	9661	UINT32	double-click to view
time	SDS	9661	FLOAT32	double-click to view
unit	Attr		STRING	s

Figure 76: Structure of the HDF5 analysis file as seen in DAWN's tree view

To make the fit results available for later analysis an HDF5 file is created. The structure of the file is shown in Figure 76.

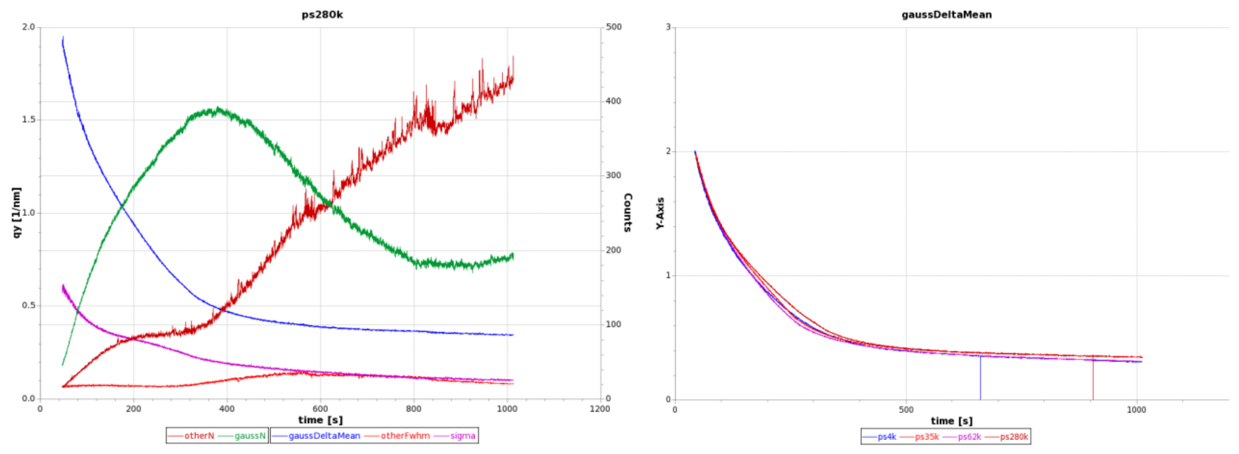


Figure 77: Example analysis results plotted from Python

The file containing the fit results can be explored with the DAWN GUI. The result of this analysis is twofold: first, the overlay of the time dependencies of the relevant fit parameters for each sample, as shown for one sample in the left of Figure 77. Second, the comparison of each fit parameter's time dependence between the samples, as shown for one parameter on the right. This is the input for the scientists who compares these results with models about the growth of the gold layer on the different surfaces of the substrates.

6.4.2 Small angle scattering analysis with DPDAK

DPDAK was developed in close collaboration with a team of scientists investigating grazing incident small angle scattering. So the core set of plug-ins for this analysis exists and is shipped with the program:

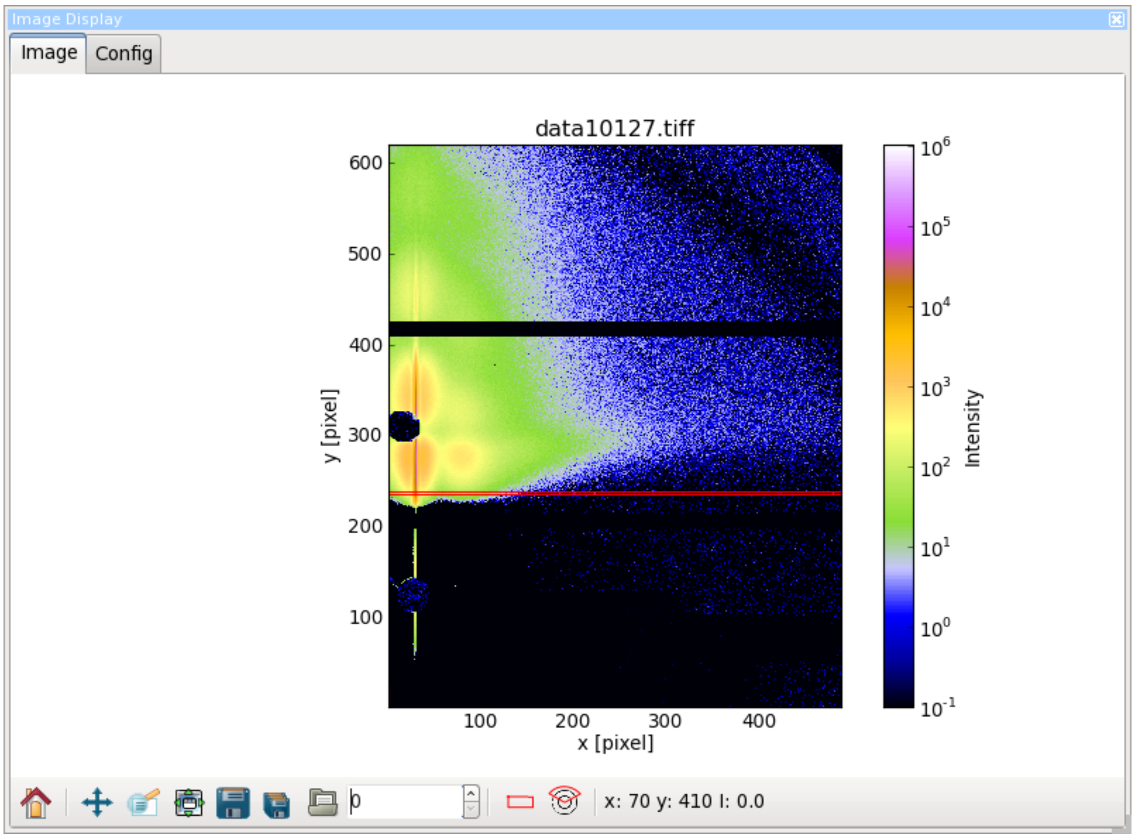
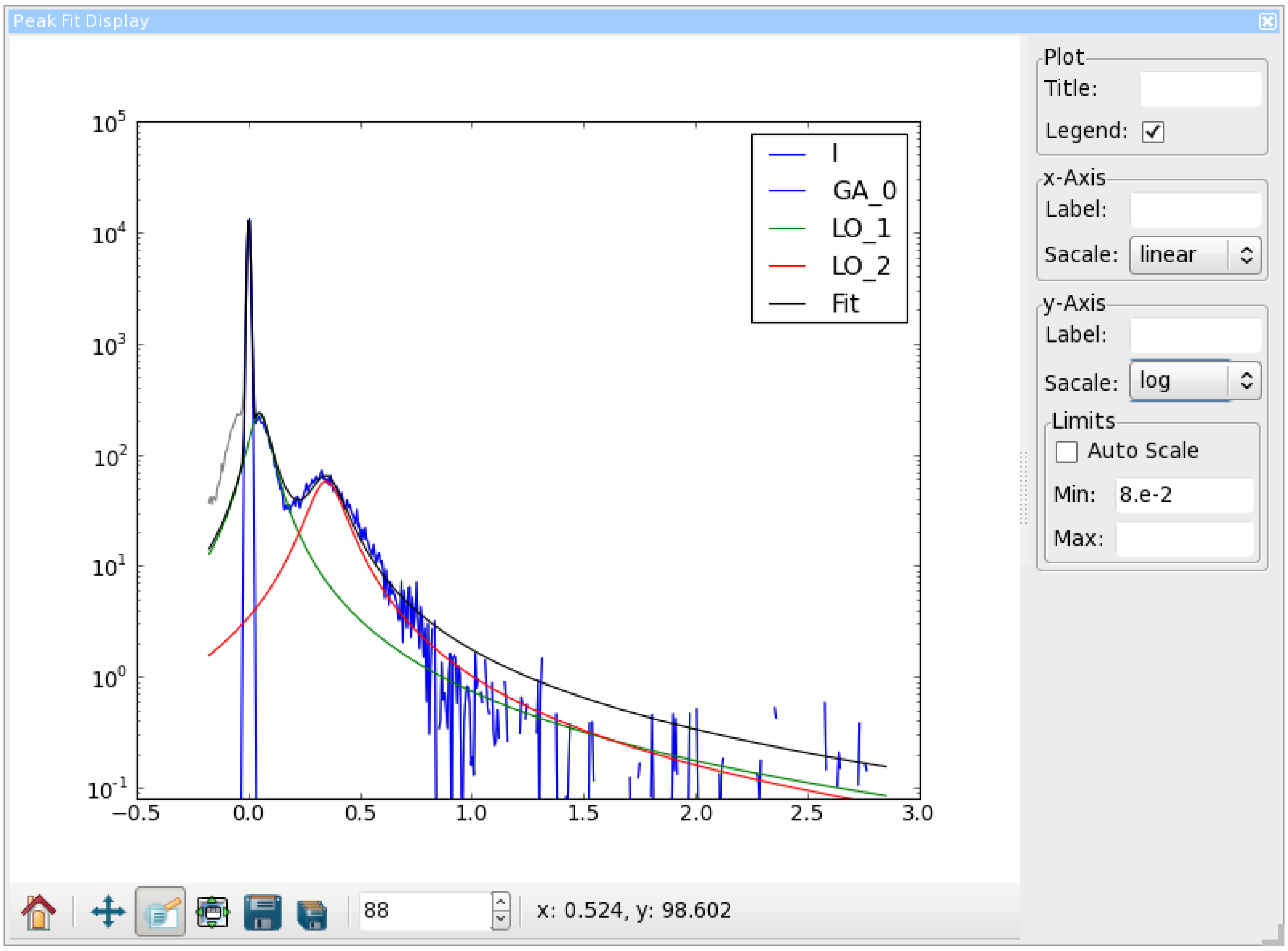


Figure 78: DPDAK's image display automatically indicating the horizontal line cut.

- The ImageLogger creates image file names based on a template path and a sequence defined e.g. by 'start-end:step'. The template path can contain wildcard characters. A step parameter of 10 means that every 10th images is read. This feature is used in pre-analysis runs.
- The LineIntegrationGISAXS plug-in integrates a horizontal or vertical slice of the image data taking into account the geometry of the experimental setup (wavelength, detector-sample distance, pixel dimensions, beam angle on surface and direct beam position, provided as plug-in parameters) and the background. The result is an intensity distribution as a function of the horizontal scattering vector q_y.
- The PeakFit plug-in performs fits. A fit function can be assembled as a sum of peaks of different shapes and a linear or constant background. Start values of the fit parameters have to be provided and can be marked as fixed parameters. Instead of constant start values, an option allows to use the result of the fit of the previously processed image instead. Further options are the fit range, the maximum number of iterations and the fit type, switching between least squares and total least squares. The results of the plug-in are the fit parameters and their standard deviations (i.e. the square roots of the diagonal elements of the covariance matrix) and the function values at the input x-values, using the fit result for the parameters.
- The ExtractFromArray plug-in extracts parameters from a sequence containing the fit results. This plug-in is needed to plot each parameter versus the image number.

The ‘Image Display’ tool in Figure 78 shows the result for the last processed image. The position of the horizontal cut is indicated.



Plot
Title:
Legend: ✓

x-Axis
Label:
Sacale: linear

y-Axis
Label:
Sacale: log

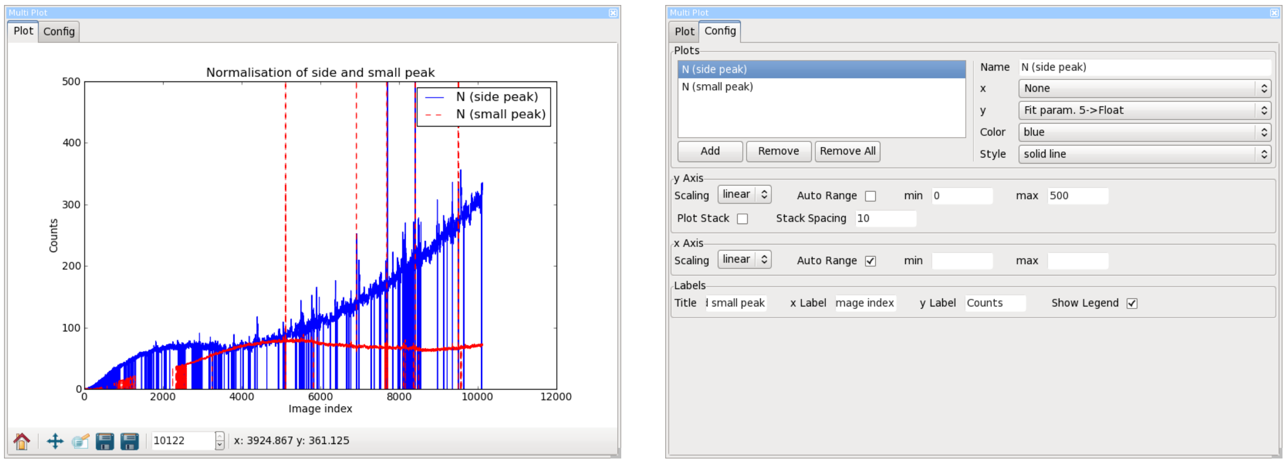
Limits
Auto Scale
Min: 8.e-2
Max:

Figure 79: Peak fit display showing the result of the function fit to image 8800, automatically indicating the contribution of the different fit function components.

The diffractogram $I(q_y)$ extracted by the LineIntegrationGISAXS plug-in is fed to the PeakFit plug-in. DPDAK does not foresee defining arbitrary fit functions, but just the sum of several peaks and optionally constant or linear background. The fit range fit range is fixed to be 0-2.85 nm^{-1} for all images and the Gaussian to describe the small outer peak is replaced, compared to the DAWN analysis, by a Lorentzian. The latter has the advantage to account for some background in the high q_y tail.

The PeakFit plug-in needs a parameterisation of the fit function: each peak function is stated with its shortcut (GA or LO), followed by the start values of their parameters. A parameter value followed by a star '*' is kept fixed during the function fit. If the option 'Reuse Fit Results' is chosen, all fits but the first one use the results of the preceding fits. This is a valuable option especially for a peak moving with time as in the case of this analysis. In case that some fits do not converge the option becomes unusable.

The Peak Fit Display, shown in Figure 79, allows investigating the fit result and the contributions of all fit function components. It is an interactive tool. The display is immediately updated whenever the user changes a fit parameter. Note that data outside of the fit range is displayed in grey, in contrast the data used in the fit.



Plots: N (side peak), N (small peak)

Name: N (side peak)
x: None
y: Fit param. 5->Float
Color: blue
Style: solid line

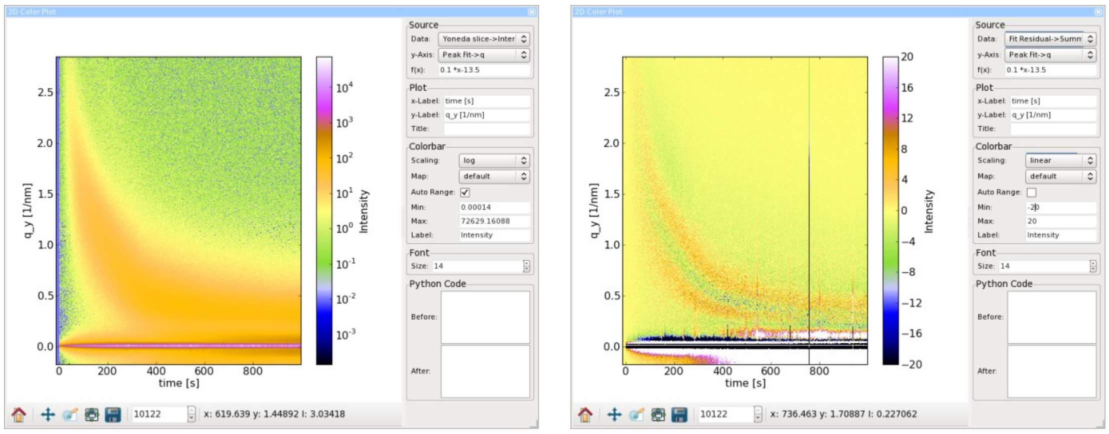
y Axis
Scaling: linear | Auto Range: ☐ | min: 0 | max: 500
Plot Stack: ☐ | Stack Spacing: 10

x Axis
Scaling: linear | Auto Range: ✓ | min: | max:

Labels
Title: l small peak | x Label: mage index | y Label: Counts | Show Legend: ✓

Figure 80: Evolution of normalisation fit parameters (left) and respective 'Multi Plot' display configuration (right).

One way to check the quality of the fit is to investigate the temporal evolutions of fit parameters. Figure 80 shows an overlay of the normalisation parameter evolutions of the side peak and the small peak. It is obvious that many fit results deviate from the main trend. The deviations originate from fits that do not properly converge.



Source
Data: Yoneda slice->Inter
y-Axis: Peak Fit->q
f(x): 0.1 *x-13.5
Plot
x-Label: time [s]
y-Label: q_y [1/nm]
Title:
Colorbar
Scaling: log
Map: default
Auto Range: ✓
Min: 0.00014
Max: 72629.16088
Label: Intensity
Font
Size: 14
Python Code
Before:
After:

Source
Data: Fit Residual->Sumn
y-Axis: Peak Fit->q
f(x): 0.1 *x-13.5
Plot
x-Label: time [s]
y-Label: q_y [1/nm]
Title:
Colorbar
Scaling: linear
Map: default
Auto Range: ☐
Min: -20
Max: 20
Label: Intensity
Font
Size: 14
Python Code
Before:
After:

Figure 81: '2D Colour Plot' plug-in to visualise the evolution of data (left) and fit residual (right).

For a comparison of the data and the fits three maps are created. The first map consists of diffractograms (cuts at the Yoneda peak) sorted by image number. This map is displayed in the left of Figure 81. The intensities are represented by colours. A second map is created containing intensities as expected from the fits. The third map is the difference of the other two. The map of the fit residuals is displayed in the right of Figure 81. Besides the bad description for $q_y < 0$, i.e. outside the fit range and where the additional side peak, symmetric to the main peak is located, one can see that the development of the side peak is not described. In addition, the tails of the small peak moving towards $q_y = 0$ are underestimated by the fit. Nevertheless, the discrepancies observed do not disturb the relevant features observed in the data.

Comparing results of different substrates by overlaying the evolution of their fit parameters has to be done using external tools that read in the exported text files. The external tool can then also be used to filter out the unreasonable fit parameters from fits that are not converged.

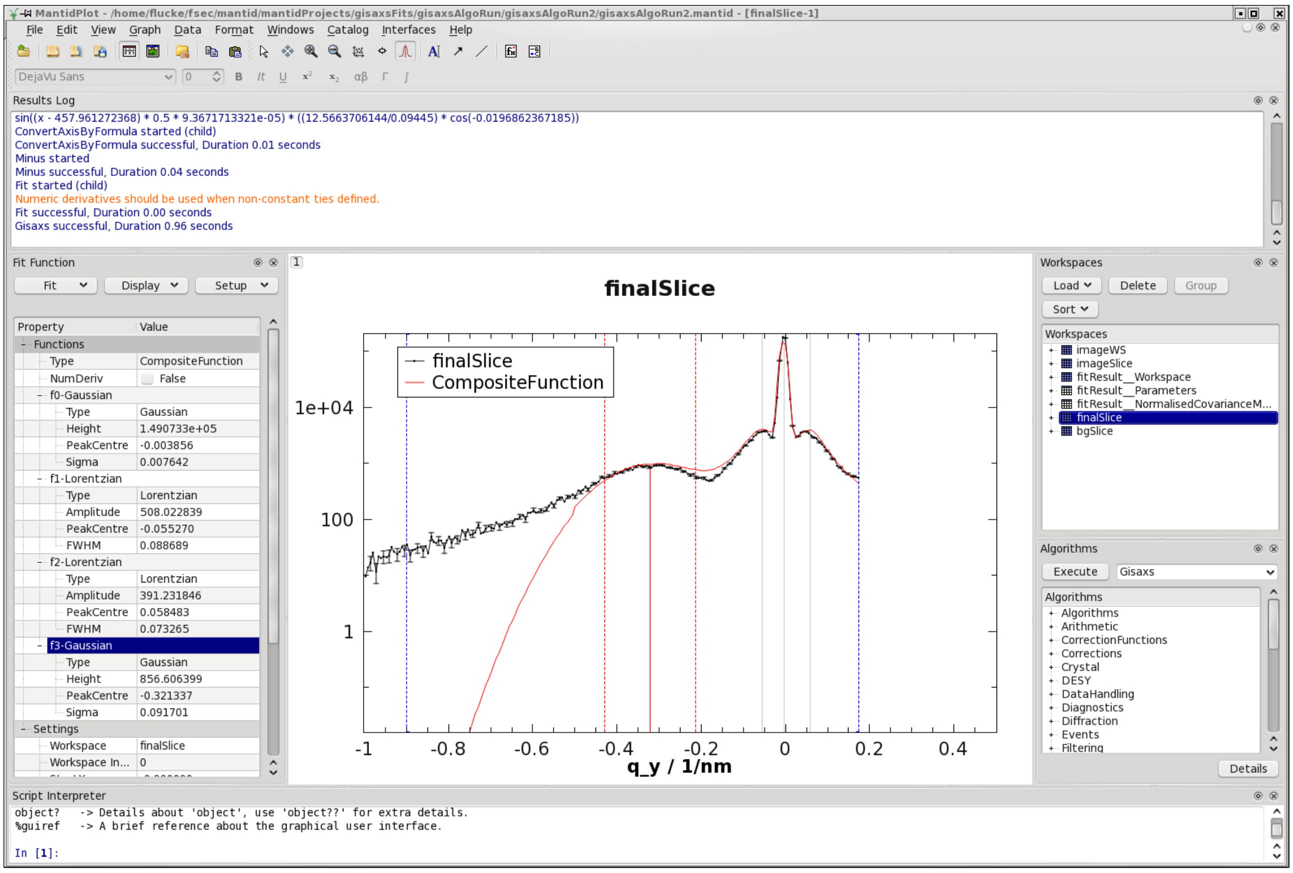
6.4.3 Small angle scattering analysis with Mantid:

The following steps are executed to perform an interactive analysis of a single image in Mantid:

- The LoadImage algorithm, as developed the Mantid PDF application, has been re-used. It was extended to include a 'FillErrors' option setting the workspace error values to square roots of the intensities. The errors are an important input to the fit procedure.
- Mantid's PerformIndexOperation creates a horizontal slice by summing over the vertical index range 374-380. The result is a Workspace2D with a single row and with the appropriate errors for summing intensity data.
- The ConvertAxisByFormula algorithm transforms the x-axis of the horizontal slice into q_y space. Also axis title and units can be defined with this algorithm.
- The same procedure as above is applied to the background image.
- The background image is scaled by the factor (0.00428) found during the DAWN analysis. The background is subtracted from the image. Scaling and subtraction were executed from the Mantid Python command line. Arithmetic operators are overloaded to act on workspaces and numbers. In this case the syntax is simply

  ```
  subtractedWS = imageWS - factor * backgroundWS
  ```

  Mantid properly propagates the errors.
- The fit function and the start values of the parameters are prepared by Mantid's graphical fit interface. The function is the sum of a Gaussian for the main peak, two Lorentzians for the side peaks and a Gaussian for the small peak that moves with time towards the main peak. Figure 82 shows a screenshot. The blue vertical lines indicate the fit range. The start values of the position, the height and the width are displayed in red and grey. The colour red identifies the currently edited peak. The start values can be specified by numbers or with the mouse.



```
sin((x - 457.961272368) * 0.5 * 9.3671713321e-05) * ((12.5663706144/0.09445) * cos(-0.0196862367185))
ConvertAxisByFormula started (child)
ConvertAxisByFormula successful, Duration 0.01 seconds
Minus started
Minus successful, Duration 0.04 seconds
Fit started (child)
Numeric derivatives should be used when non-constant ties defined.
Fit successful, Duration 0.00 seconds
Gisaxs successful, Duration 0.96 seconds
```

Property	Value
Functions	
Type	CompositeFunction
NumDeriv	False
f0-Gaussian	
Type	Gaussian
Height	1.490733e+05
PeakCentre	-0.003856
Sigma	0.007642
f1-Lorentzian	
Type	Lorentzian
Amplitude	508.022839
PeakCentre	-0.055270
FWHM	0.088689
f2-Lorentzian	
Type	Lorentzian
Amplitude	391.231846
PeakCentre	0.058483
FWHM	0.073265
f3-Gaussian	
Type	Gaussian
Height	856.606399
PeakCentre	-0.321337
Sigma	0.091701
Settings	
Workspace	finalSlice
Workspace In...	0

Workspaces
- imageWS
- imageSlice
- fitResult__Workspace
- fitResult__Parameters
- fitResult__NormalisedCovarianceM...
- finalSlice
- bgSlice

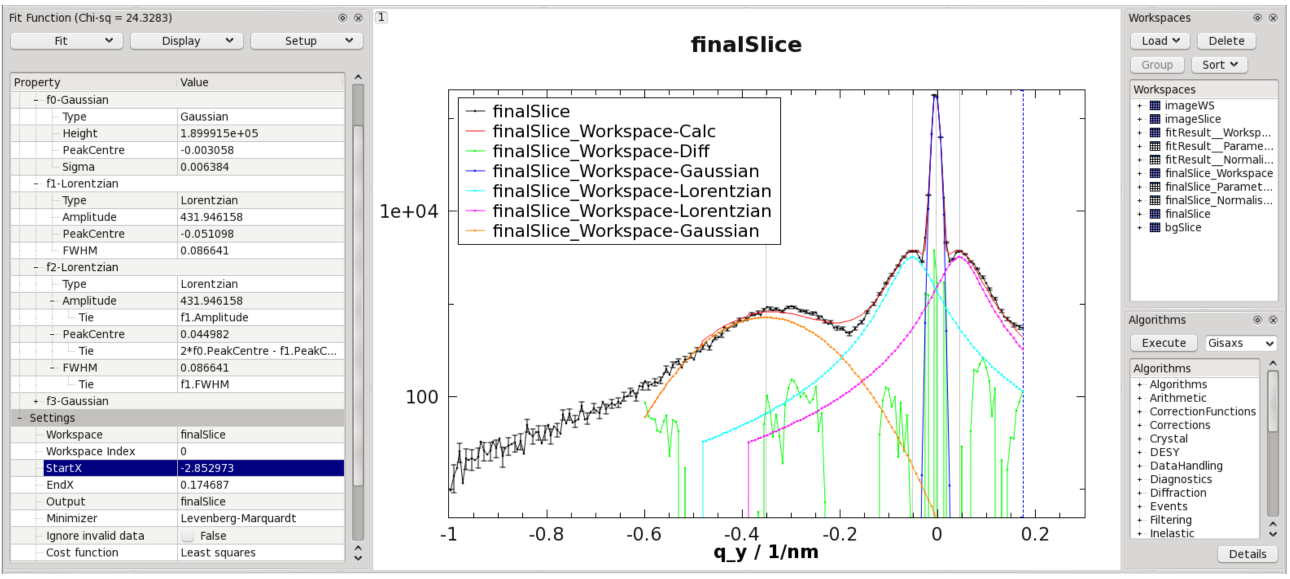
Algorithms
- Algorithms
- Arithmetic
- CorrectionFunctions
- Corrections
- Crystal
- DESY
- DataHandling
- Diagnostics
- Diffraction
- Events
- Filtering

```
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Figure 82: Mantid's graphical fitting interface used to define the function as sum of four peaks.

- Before actually fitting the function, the parameters of the second Lorentzian are declared to be 'tied' in such a way that the two Lorentzians are symmetric around the main peak. That means that their FWHM and amplitude are identical and that the peak position of the right Lorentzian fulfils $p_r = 2 * p_m - p_l$. Here p_m and p_l are the peak positions of the main peak and of the left Lorentzian, respectively. Mantid uses the term 'tie' to express constraints between fit parameters.
- The fit is performed as a least squares fit using Mantid's default 'Levenberg-Marquardt minimiser' taking the errors into account.
- The fit result is displayed in Figure 83 showing the data (black), the fitted function (red), the individual peaks as well as the difference between the fit and the data (green).


Fit Function (Chi-sq = 24.3283)
Fit
Display
Setup
1
finalSlice
finalSlice
finalSlice_Workspace-Calc
finalSlice_Workspace-Diff
finalSlice_Workspace-Gaussian
finalSlice_Workspace-Lorentzian
finalSlice_Workspace-Lorentzian
finalSlice_Workspace-Gaussian
1e+04
100
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
q_y / 1/nm
Workspaces
Load
Delete
Group
Sort
Algorithms
Execute
Gisaxs
Details

Property	Value
- f0-Gaussian	
Type	Gaussian
Height	1.899915e+05
PeakCentre	-0.003058
Sigma	0.006384
- f1-Lorentzian	
Type	Lorentzian
Amplitude	431.946158
PeakCentre	-0.051098
FWHM	0.086641
- f2-Lorentzian	
Type	Lorentzian
- Amplitude	431.946158
Tie	f1.Amplitude
- PeakCentre	0.044982
Tie	2*f0.PeakCentre - f1.PeakC...
- FWHM	0.086641
Tie	f1.FWHM
+ f3-Gaussian	
- Settings	
Workspace	finalSlice
Workspace Index	0
StartX	-2.852973
EndX	0.174687
Output	finalSlice
Minimizer	Levenberg-Marquardt
Ignore invalid data	False
Cost function	Least squares

Workspaces
- imageWS
- imageSlice
- fitResult__Worksp...
- fitResult__Parame...
- fitResult__Normali...
- finalSlice_Workspace
- finalSlice_Paramet...
- finalSlice_Normalis...
- finalSlice
- bgSlice

Algorithms
- Algorithms
- Arithmetic
- CorrectionFunctions
- Corrections
- Crystal
- DESY
- DataHandling
- Diagnostics
- Diffraction
- Events
- Filtering
- Inelastic

Figure 83: Mantid's graphical fit interface after fitting a function composed of four peaks with tied parameters.

- The settings of the fit functions are saved within the project. They comprise the function composition, the start parameters and the ties between the parameters. The settings can be copied as text to the clipboard.

The interactive work explained so far is a preparation for the analysis of a full dataset consisting of about 10.000 images. Data processing of the image series is carried out by Python scripts involving the following steps:

- A Python loop executes the following tasks for a series of images.
- The LoadImage, PerformIndexOperation, ConvertAxisByFormula algorithms are called for each file. The background is subtracted as described above.
- Mantid's fit algorithm is applied. The fit range is dynamically adjusted like in the GISAXS DAWN example.
- The fit results are extracted from the workspaces created.
- The time dependence of the fit parameters are displayed and stored.

Most of the aforementioned steps have already been demonstrated in the preceding sections. The Python interface to the fit algorithm has not been covered yet. The following input is required:

- a workspace containing the data points and their errors,
- a string specifying the function and its start parameters,
- a string specifying ties between parameters,
- the maximum number of iterations for the minimiser,
- the fit range
- and optionally further parameters, e.g. the choice of another minimiser.

A semicolon separated list of substrings specifies the function, each substring defines a peak by supplying a function name and a comma separated list of parameters, e.g.

```
name=Gaussian,Height=189893,PeakCentre=0.,Sigma=0.01
```

Ties are also defined by a comma separated list where each element defines the tie of one parameter, e.g.

```
f2.PeakCentre = 2 * f0.PeakCentre - f1.PeakCentre
```

Here `fn` stands for function component `n` (counting from zero) and the first component is the main Gaussian, the second (third) the left (right) Lorentzian side peak.

The fit result consists of a status string, the χ^2/ndf value as a goodness of fit estimator and three workspaces. One workspace contains the fit parameters and their errors; a second workspace contains the normalised covariance matrix. The last workspace contains the fitted data, the function estimate, and the contributions of all peak functions.
The aforementioned functions setting are re-used by the Python procedure. They contain the peak function composition, start values for the parameter and parameter ties.

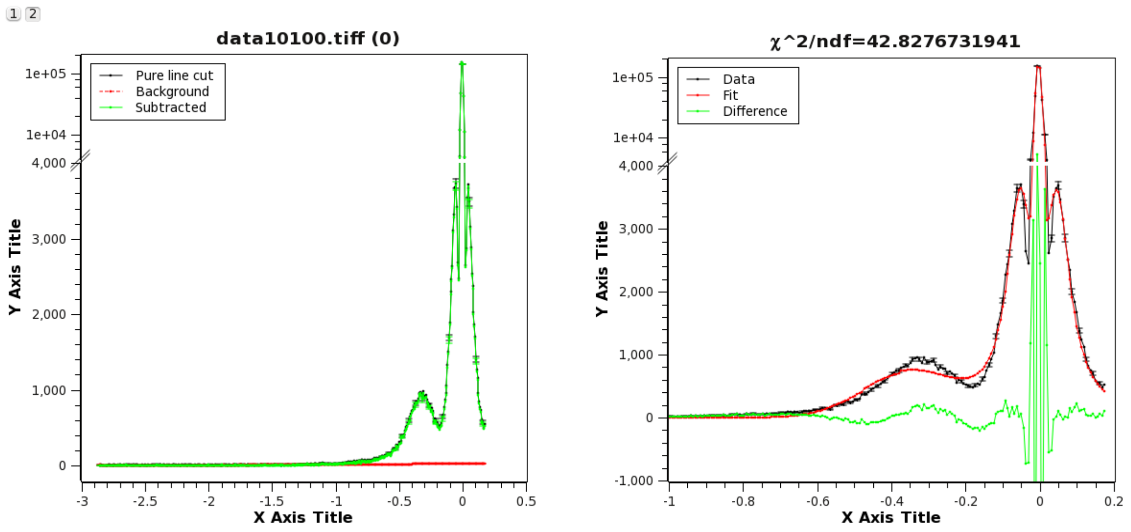


Figure 84: Mantid fit results acquired within Python, displayed using vertical axes with breaks.

Figure 84 shows the result of a fit carried out by a Python script. The covariance matrix contains the precision of the fitted parameters and their correlation.

It has been demonstrated that a full GISAXS dataset can be analysed in Mantid. The availability of the LoadImage algorithm, which was developed within this project, is a prerequisite.

6.5 Comparing Remarks

The three investigated programs provide an abundant set of tools and features. The wide variety of available options prevented us from evaluating the frameworks in all details. Therefore, the summarizing remarks below are restricted to the experiences gained during the implementation of the two use cases. An important difference has to be stated in advance: DAWN and Mantid are big, 'fully-featured', mature frameworks supporting a large community. They are substantially supported by their home institutes. DPDAK does not belong to this category. Its functionality is comparatively limited and it receives less support.

The next paragraphs address the frameworks individually. The section ends with a table comparing keys issues.

- DAWN has been created to analyse and visualise arbitrary data from synchrotron radiation experiments. It supports many different file formats, facilitates generic multi-dimensional browsing, slicing, profiling of data, etc.
  The HDF5 viewer is a highlight which has proven to be useful for all kinds of HDF5 files.
  Eclipse is DAWN's strength and weakness. Customized applications can be created using Perspectives and Views. This is a powerful feature. Furthermore, Eclipse imposes a concept to extend DAWNs functionality in a standardized way. The only weak point about Eclipse is that it needs considerable expertise to make best use of it.

DAWN is written in Java. Therefore, its natural scripting language is Jython. On the other hand, researchers tend to prefer CPython because of the availability of scientific software packages. DAWNs Python and Jython APIs are not fully equivalent.
Documentation is always an issue, also for DAWN. In the course of the project it was sometimes pointed out that certain features were not fully described and that the operation of certain components is not very intuitive.

- DPDAK is designed for online and offline processing of image series obtained from X-ray scattering experiments. It is a light-weight plug-in system. The plug-in interfaces are comparatively simple making DPDAK easily extendable. Even unexperienced users are able to configure data processing chains. Another nice feature is that all plug-in output is transferred to an internal database from where data are fetched for a detailed inspection or for an export to files. Unfortunately, the internal database has a scalability problem, if thousands of images are processed.
  Data types passed between the plug-ins are restricted to scalars, 1D arrays, strings, and file paths. The latter can be used to pass two-dimensional data using (temporary) image files. The DPDAK documentation is partly incomplete.
- Mantid is designed for scientific procedures used in neutron scattering and muon spectroscopy. It provides the usual data processing and visualisation tools. In addition, Mantid creates 2D or 3D instrument views displaying the integrated neutron intensities in the detectors.
  The workspace concept is Mantid's central aspect. Workspaces hold data that are processed by algorithms. The history of applied algorithms is tracked. Metadata are also stored in workspaces. Mantid allows users to specify errors enabling a proper statistical data analysis. A problem related to workspaces is that it complicates the import of arbitrary multidimensional data. Special loaders have to be developed.
  The interactive function fitting tool is one of Mantid's highlights. Especially appealing is the option to define position, height and width of peaks using the mouse. Furthermore, Mantid is the only investigated program that allows defining constraints between fit parameters.
  An exceptional property of Mantid is the deep integration of Python as scripting language. This enables the user to extend Mantid's core functionalities using Python or to run batch scripts using all non-graphical Mantid functionality.

Table 7: Comparison of DAWN, DPDAK and Mantid

	DAWN	DPDAK	Mantid
Main supporting institutes	Diamond, ESRF	DESY, MPI of Colloids and Interfaces	ISIS at RAL, Oak Ridge
Main focus	Synchrotron experiments	WAXS, SAXS, GIWAXS, GISAXS	Neutron scattering, muon spectroscopy
Supported platforms	Windows, Linux	Windows, Linux	Windows, Linux, Mac OS
Language	Java	Python	C, C++, Python
Version control by DOIs	No	No	Yes
Active programmers	Many	Few	Many
Documentation and examples	A lot, but incomprehensive and scattered	Not complete	Comprehensive
Channel for user questions	Feedback button	Mailing list	Mail address reachable via 'Help' menu
Learning curve	Slow increase	Steep increase	Medium increase
Scripting language	Python/Jython	Python	Python
Script development	Excellent	---	Yes

	DAWN	DPDAK	Mantid
support			
Input data formats	Hdf5/NeXus, images, .dat, .edf, .fio, .mar, pilatus, numpy, etc.	Images, text files	Specific (NeXus-) variants
NeXus/Hdf5 browsing	Yes	No	No
Visualisation	1D, 2D, 3D, Hyper3D	1D, 2D	1D, 2D, 3D, instruments
Interactive control of plot appearance	Good	Medium	Comprehensive
Control of plot appearance from script	Limited	Undocumented expert tool available	Comprehensive for 1D, limited for 2D
Masks for image data	Yes, create and apply	Yes, but only apply	No
Data history tracking	No	No	Yes
Peak fitting	Yes, but not from Python	Yes	Excellent
Generic fitting	Yes, but neither from Python nor usable in data reduction	No	Excellent
Treatment of data errors e.g. for fitting	No	No	Yes, built-in in workspaces
Extendibility	Eclipse system offers many 'extension' points, but their usage remains expert task.	Easy to add plug-ins for scientific procedures, but data formats between plugins limited.	Excellent support to add scientific procedures using Python, but limited set of data formats ('workspaces') to exchange data between algorithms
Batch Processing	Yes, for workflows	No	Yes, for Python
Storage of analysis output	HDF5 output for data reduction	Results of all plug-ins kept in memory, exportable to text; scalability questionable	Workspaces can be stored as NeXus files

Comparing DAWN and Mantid it seems that the DAWN developers have a more generic approach when implementing features. Furthermore, Mantid offers less functionality to investigate multi-dimensional data. The reason is perhaps there is so far no real use case in neutron science. But Mantid is deeply integrated into the ISIS and SNS experiments, exemplified by the instrument views and the workspace concept. In addition, Mantid has an excellent Python integration. DPDAK is flexible and simple. It requires little effort to customize DPDAK according to the need of users.

7 Conclusions

The survey of the implementation of the PaN-data ODI services shows the progress that has been made at the participating institutes.

- The "Common policy framework on scientific data", as proposed in the PaN-data Europe deliverable D2.1 is widely accepted among the partners.
- The digital user offices of the majority of the PaN-data institutes accept credentials from the common authentications system Umbrella.
- The deployment of the common metadata catalogue ICAT is making large progress. In addition to ISIS and DLS having fully operational ICAT services other institutes are close to reach a production state.
- NeXus became the de facto standard data format of experiment data measured at European neutron and synchrotron radiation sources.

The Science3D portal is a demonstrator for an open access database. Tomographic data from biological samples have been made available for experts and for all who are interested in the microstructure of organisms. The explanations accompanying the 3D images make Science3D suitable for educational purposes.

The existing ICAT services have been evaluated based on VL1 and VL2 examples together with investigators who created the original data. Various methods of data discovery have been discussed and also the role of metadata.

An overview of the NeXus deployment revealed how the participating institutes create NeXus file structures depending on the specific experimental techniques. This comprises the organization of raw data and the choice of descriptive information.

A comparison of the data processing frameworks DAWN, DPDAK and Mantid has been carried out. The first test case, originating from VL1, is an online monitoring task for studies of disordered or amorphous materials by means of the reduced pair distribution function. Three applications, one for each of the frameworks, have been developed under the supervision of researchers. The applications were tested at real experiments. The second use case, a visualization and analysis task for small angles scattering data, is an offline application from VL2. Again it was implemented for each of the considered frameworks. Following this approach it was possible to identify the advantages and shortcomings of the aforementioned frameworks with respect to the considered use cases.