NOBUGS 2016

Proceedings

Foreword

These are the proceedings of the eleventh NOBUGS Conference that was held in Copenhagen, Denmark on the 17th to 19th October 2016.
The NOBUGS (New Opportunities for Better User Group Software) Conference Series has the aim to foster collaboration and exchange between scientists and IT professionals working on software for X-ray, neutron and muon sources around the world. NOBUGS 2016 was jointly organised by the European Spallation Source (Data Management and Software Centre), University of Copenhagen (Niels Bohr Institute) and the MAX IV Laboratory.

In these proceedings you will just find the peer reviewed articles that have been submitted for publication. The conference website has other relevant information, for example slides or posters that have been presented. Please visit:

https://nobugs.esss.se

If you are interested in other NOBUGS conferences (past or upcoming) you should find more information on:

http://nobugsconference.org

The organisers are especially grateful for the financial support of the conference by our premium sponsors Tessella and QuantumWise.

Copenhagen, December 2016

Tobias Richter	For the local organisers: Petra Aulin Vincent Hardion Thomas Holm Rod Krister Larsson Stig Skelboe

Table of Contents

Community Driven Scientific Software Projects: Lessons Learned on Tools and Practices 1
Pascual-Izarra, Carlos; Falcón Torres, Carlos; Reszela, Zbigniew; Cuní, Guifré; Fernández-Carreiras, David; Jover-Manas, Gabriel; Rosanes Siscart, Marc

Addressing the Challenges of implementing the ESRF Data Policy 7
Gotz, Andy

Scientific data lifecycle at Elettra-Sincrotrone Trieste 13
Borghes, Roberto; Prica, Milan; Kourousias, Georgios

The growth of the ICAT family 17
Fisher, Stephen M; Gotz, Andy; Phipps, Kevin; Prica, Milan; Parker, Peter; Fair, Rebecca; Krahl, Rolf; Ren, Shelly; Da Graca Ramos, Sylvie; Griffin, Tom; Chung, Wayne; Barnsley, Frazer; Ritchie, Brian; de Maria, Alex; Alex, Potter-Dixon; Matthews, Brian; Prosser, Christopher;
Rao, Jianguo; Salt, Jody

Building a Prototype Data Analysis as a Service : the STFC experience 23
Barnsley, Frazer; Matthews, Brian; Griffin, Tom; Salt, Jody; Ross, Derek; Packer, Alison; Condurache, Catalin; Dibbo, Alexander

SIMEX: Simulation of Experiments at Advanced Laser Light Sources 29
Fortmann-Grote, Carsten; Briggs, Richard; Bussmann, Michael; Buzmakov, Alexey; Garten, Marco; Grund, Alexander; Huebl, Axel; Jurek, Zoltan; Loh, Duane; Rueter, Tonn; Samoylova, Liubov; Yakubov, Sergey; Yoon, Chun Hong; Mancuso, Adrian P.; Ziaja, Beata; Santra, Robin; Hauf, Steffen; Joy, Ashley; Wing, Matthew

The unified software package Sonix+. Analysis of development experience 35
Kirilov, Andrey

DonkiOrchestra: a scalable system for data collection and experiment management based on ZeroMQ distributed messaging 41
Borghes, Roberto; Kourousias, Georgios

On-Axis-View: a GUI library to enhance the sample environment control. 47
Roldan, Daniel; Andreu, Jordi;
Homs Puron, Roberto; Jover-Manas, Gabriel;
Becheri, Fulvio; Cuní, Guifré; Pascual-Izarra, Carlos

Graphical user interface and experiment control software at the MX beamlines at EMBL Hamburg 53
Karpics, Ivars; Bourenkov, Gleb;
Schneider, Thomas; Nikolova, Marina

MXCuBE 3 web application, on the way to next generation experiment control 59
Eguiraun, Mikel; Milán Otero, Antonio;
BOLMSTEN, Fredrik; NAN, Jie;
THUNNISSEN, Marjolein; Spruce, Darren;
HARDION, Vincent; GUIJARRO, Matias;
OSKARSSON, Marcus; BETEVA, Antonia;
DE SANCTIS, Daniele; MEYER, Jens; Gotz, Andy;
Leonard, Gordon

Programs and techniques based on ROOT package for acquisition and sorting of the list mode data of the neutron detectors 65
Litvinenko, Elena; Bogdzel, Andrey;
Churakov, Andrey; Rossa, Lutz; Sauer, Olaf-Peter; Wilpert, Thomas

"Manyo-Lib" Object-Oriented Data Analysis Framework for Neutron Scattering 72
Suzuki, Jiro; Inamura, Yasuhiro; Ito, Takayoshi;
Nakatani, Takeshi; Otomo, Toshiya

IROHA2: Standard instrument control software framework in MLF, J-PARC 76
Nakatani, Takeshi; Inamura, Yasuhiro;
Moriyama, Kentaro; Ito, Takayoshi

Recent Progress in the Development of MLF EXP-DB in J-PARC/MLF 80
Moriyama, Kentaro; Nakatani, Takeshi

Safety systems at the SAFARI-1 Neutron Diffraction Facility 86
Raaths, Christo; Van Heerden, Rudolph;
Marais, Deon; Venter, Andrew

DOI: 10.17199/NOBUGS2016.42

Community driven scientific software projects: lessons learned on tools and practices

C Pascual-Izarra *, C Falcón-Torres, Z Reszela, G Cuní, D Fernández-Carreiras, G Jover-Manas and M Rosanes-Siscart

ALBA-CELLS Synchrotron, Cerdanyola del Vallés, Spain

E-mail: `cpascual@cells.es, ctgensoft@cells.es`

Abstract. On the one hand, science is about openness and collaboration and, on the other hand, open and community-driven software projects have demonstrated to yield more generic and resilient solutions thanks to the diverse users' needs and feedback. Yet, the development of many scientific software projects (even free/open-source ones) is still managed in a closed way, typically by one or few people from a single institution. In some cases this is just due to the lack of knowledge or confidence on the already available tools and practices for collaborative development. In this work we share our experience on coordinating the transition of the Taurus and Sardana projects from being in-house developments to being driven by an international and diverse community. The initially established rules are constantly evolving aiming to reach continuous delivery while maintaining good software quality. The selected contribution workflows, testing strategies as well as the software and documentation delivery tools are described in detail, and the benefits of this kind of organization as well as potential pitfalls and lessons learned are discussed.

1. Introduction

It is generally accepted that software projects whose development is driven by a community (as opposed to a closed group of developers) tend to become more resilient (e.g. against the eventual leave of a main author), have better code quality (more bugs spotted and corrected), attract more contributions (making them more generic) and, as a consequence, get a wider exposure [1]. But this comes at some cost: first, one needs to provide tools for coordinating a geographically dispersed community of developers; second, time must be devoted to document procedures and policies and for supporting new developers in order to keep a coherent code quality and style; third, the contributions require to be peer-reviewed before being incorporated and, finally, a much stricter policy regarding the Application Programming Interface (API) is required to avoid breaking other people's developments.

While these are general good practices, closed-team development is more forgiving in taking shortcuts and exceptions to them in the name of short-term benefits. Also, some authors may fear the loss of control over their "pet" projects, or may not be familiar with the tools and practices available for collaborative development. This may explain why so many scientific software is developed in a non-open way (even if it uses free/open-source licenses).

In the rest of this paper we will try to encourage scientific software authors to consider a collaborative development model by showing the example of the Sardana [2, 3] and Taurus [4, 5] projects, which successfully transitioned from in-house to community-driven development.

2. Background
Sardana and Taurus are projects initially developed to satisfy the needs for the ALBA synchrotron beamline and accelerator control system [6]. Since their inception around 2009 (or earlier) they were licensed as Free/Open Source software, but their development was internal to ALBA and the code repository was only accessible from within the site.

In 2011, the code repositories for both Sardana and Taurus were moved to SourceForge [7] in order to facilitate adoption by third parties, but the development model continued to be essentially in-house and the support to other facilities was considered only a last priority.

Still, some other laboratories expressed their interest in using Taurus and Sardana, and as a consequence the move towards a community-driven project was proposed in order to share both the decision-making and the load of the development.

3. Opening the development
3.1. Seeding the Sardana Community
Following the example of the Tango Community [8, 9], which was initially based on a Memorandum of Understanding (MoU), some attempts were done for signing a formal agreement between the interested facilities. But the review and discussion on this MoU got delayed and the community still has no institutional agreement (nor seems to require one so far).

On technical grounds, the first efforts were put to formalize the processes on how to contribute to the project. For shared design decisions, the Sardana Enhancement Proposal (SEP) [10] process was agreed, inspired on Debian Enhancement Proposal (DEP) [11]. The SEP process defines the discussion workflow for promoting ideas into actual Sardana or Taurus features or policies. Regarding code contributions, a process for *public* code review was discussed and agreed [12]. It not only promotes the quality of the code but it also benefits the developers who can easily share their knowledge. Following the example of the Linux kernel project, we based the review process on patches sent and discussed by email (which only requires an email client, and does not require to log into any external service). But, in retrospective, we found two main issues: for the integrators it is cumbersome, especially when the contributions accumulate in the lists; for the contributors, it requires to learn and follow some conventions when emailing or discussing their contributions.

A biannual release cycle was selected according to the needs of the involved institutes. Each of the releases, preceded by extensive manual tests on various platforms, ensure a periodic checkup of the project. In addition, the pending issues may be assigned to a given release, improving the project transparency.

Two mailing lists, one dedicated to the developers and the other to the users were created. They are the main discussion and support channel, with lots of participants. As a negative note, the traffic generated by the code review process tends to bloat the developers list.

Additionally to the written communication, the community members organize open annual Sardana Meetings, where the technical aspects and roadmaps are discussed. Initially, video conferences were considered as a complement to the presential meetings but they are not so common for now.

3.2. Tools and methodologies adopted for community development
The Git [13] distributed version control system facilitates workflows for collaboration between disperse groups of developers. For this reason, both Taurus and Sardana migrated their code from a SVN repository into a Git one within the SourceForge platform and adopted the gitflow rules [14], which fit well with the biannual release cycle.

In the last years Github [15] became increasingly popular. Sardana and Taurus, similarly to the Tango community projects, will soon migrate from SourceForge to GitHub, attracted by its user-friendliness and the possibilities enabled by its tools and related services to better coordinate a community. The Github pull-request and the code-review workflows will make the contribution process more agile and hopefully bring more participants.

Performing manual tests of the overall project on each incoming contribution resulted to be time consuming and error prone. Therefore, the SEP5 [16] was proposed, defining some basic rules on how to develop consistent automated tests. Since providing full test coverage for already-existing projects such as ours is not practical [17], we focused on developing happy path tests for existing code, while encouraging Test Driven Development (TDD) for new features. Also, when working in a team, we try to alternate the person writing the tests and the one implementing the functionality.

The automated tests deliver their full benefits when using them together with the Continuous Integration (CI) practice. Employing disposable Docker [18] containers as the underpinning technology, facilitates the design and implementation of the testing infrastructure and helps with test isolation. Having a *public* CI service is crucial in collaborative projects. This could be achieved by exposing an internal CI server to the external users (in ALBA we use Mr. Jenkins [19] internally). However, we opted to set up the Taurus and Sardana CI on external platforms instead (Travis [20] and Appveyor [21]), mainly because: a) we reduce maintenance efforts related to supporting account creation, security, etc; b) it makes the communitary infrastructure less dependent on one specific facility; and c) they integrate smoothly with Github and Docker.

The up-to-date and verbose documentation is very important when it comes to sharing the project among many institutes. The Sardana and Taurus documentation is written using Sphinx [22] and was originally hosted on ALBA's internet servers. This setup required manual builds and deployments on every release, which was tedious and impractical. Consequently, we migrated the documentation to the Read the Docs (RTD) platform [23] which builds new docs after each commit to the Git repository and, at the same time liberated us from the infrastructure maintenance. However, while the benefits of the continuous documentation practice are unquestionable, the adequacy of RTD for projects such as Sardana or Taurus is being reconsidered for the following reasons: a) it forces us to implement and maintain mocks for the various non-pure-python dependencies; b) it is prone to spurious false-positive build failures and c) its environment is difficult to replicate locally when debugging is required. Alternatively, a Travis based solution may be implemented in the future for deploying the documentation.

3.3. Coming next

While employing CI brings benefits for the developers, it does not add a direct benefit for the user. The road of the software towards the production stage does not end after a successful build or unit test execution [24, 17]. In continuation of the CI step, the rest of the Continuous Delivery (CD) pipeline needs to be implemented providing a fully automated, reliable, repeatable and constantly improving process ended with a ready-to-deploy software package. GitHub features such as the automatic releases or staging areas together with Travis CI (with Docker) and Appveyor will be very useful in building such pipelines. However, since not all parts of the code are automatically tested, the less common use cases will still need to be tested manually. Optimally, these tests should be based on the user documentation, explaining how to interact with the system.

The user could benefit even more if the developers collaboration is extrapolated into the packagers collaboration i.e. the software would become available as high quality packages for the most popular platforms. Currently Sardana and Taurus projects are present in the official Debian [25] repositories, mostly thanks to the effort of a single packager. We believe that all the benefits of the collaborative development could be achieved in the packaging processes as well. Platforms such as Alioth [26] or OBS [27] are some examples on how to implement collaborative packaging. The artifacts produced by the CI step could indeed be distribution packages that could be used in the subsequent steps of the CD pipeline and finally become a software update.

Plenty of other ideas are being evaluated or planned to be implemented in our collaboration. The most interesting ones are the automatic code-style checks for the contributions, the unit test coverage metrics or the voting for the pending issues for the priority assignment. Due to the lack of space they are skipped in this paper.

4. Beyond the technologies

While the technologies and procedures are important, the success in building and coordinating a community of developers is also based on other factors. In our case, the following were determinant:

- Being responsive and encouraging towards users and contributors. Even prioritizing support to external members over those from our own facility (thinking in the longer-term goal of building a strong community).
- Having a "gradual strictness policy": we expect contributors to follow certain conventions, but we try not to scare first-timers by being too strict with their contributions, and rather educating them as they become more involved in the community. In the meanwhile, the integrators accept the burden of adapting the contributions to keep the quality standards.
- Using standard tools and services: when deciding on alternative tools or services, value what the potential contributors are already using and know. Avoid requiring them to join many online services, and of course make sure that all the applications that we recommend are freely available (i.e. Free/OpenSource software).
- Using well-known and documented workflows: just as with the tools, prioritize what the potential contributors already may be familiar with. Even to the point of preferring sub-optimal but well-known and well-documented workflows over perfectly-customized but unique ones that would require contributors to learn one more specific convention and also require ourselves to invest time documenting it.
- Making code modular: this is a good practice in general, but even more for distributed development since it allows less experienced contributors to collaborate without fear of breaking critical parts. It also enables parallel developments.
- Being transparent: use public channels (such as mailing lists, tickets, Enhancement Proposal documents, etc) for all development-related discussions. Even when the participants are on the same site or exchange private emails regularly. In this way, others may join the discussion or at least understand the motivations of previous decisions.
- Document everything. From the APIs (obvious) to the design goals and agreed roadmaps (see previous point). This facilitates outsiders to join the community.
- Ensuring the agreements are being followed: one or more developers get the sheriff role to keep an eye on CI tests status, code static analysis, unattended queues, etc., and remind the rest of the community to act in case of deficiencies.

5. Conclusions

Moving from an in-house development model into a community-driven one was a key for the success of the Sardana and Taurus projects, whose user base grew in the last three years from a few institutions to dozens of laboratories and several companies offering support for them [4]. The contributors community is also starting to grow (see 1) and we expect it to be further expanded with the adoption of more agile contribution workflows in the near future.

The transition required ALBA to invest a lot of effort into tasks whose results were not immediate or which were competing with ALBA's most urgent issues, but which were, nevertheless, deemed necessary on a longer-term perspective.

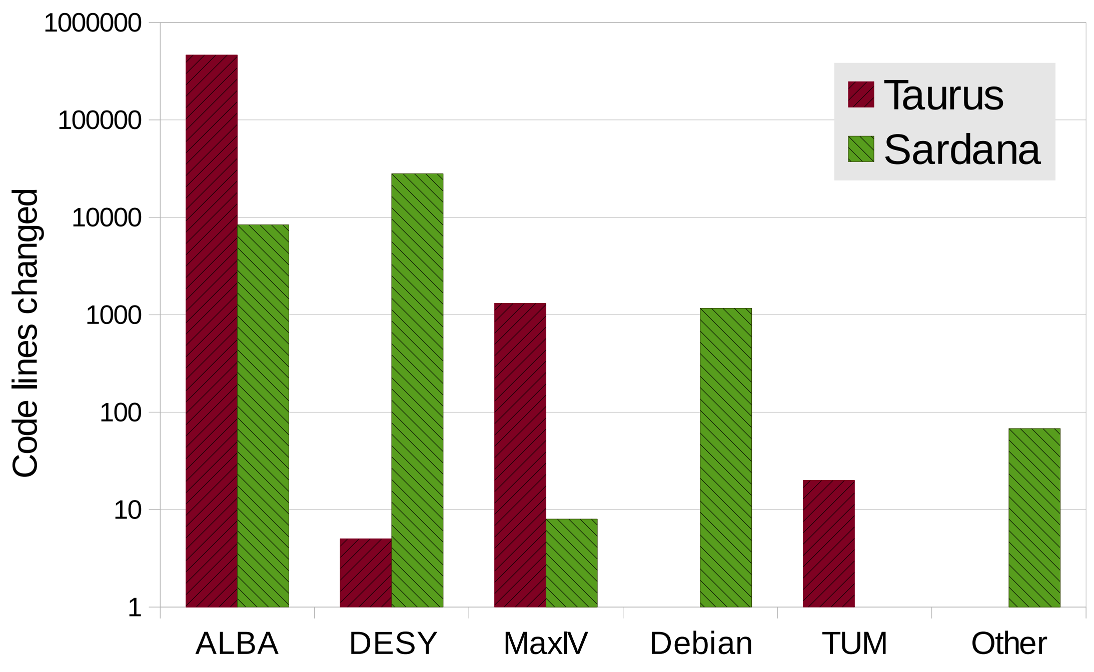


Figure 1. Contributions (measured as number of lines changed) in Taurus and Sardana projects for the period between Jul15 and Jul16 releases, grouped by author's institution

In order to facilitate the incorporation of external developers and to underline the community orientation of the projects, we favored solutions based on external providers (SourceForge, GitHub, Read the Docs, Travis, etc.) instead of on-site infrastructure. We also invested in learning and using standard practices and tools and in engaging, supporting and encouraging contributors.

Of all the decisions regarding tools and workflows, some worked excellently so far, while others already showed some weaknesses and yet some clearly failed but allowed us to learn in the process. We hope that our experiences can also be useful for other projects considering a similar transition.

Acknowledgments
We would like to thank the Sardana and Taurus community members and the ALBA Controls Group and, specially, to T Nunez, J Kotanski and T Kracht (DESY), T Coutinho, V Valls (ESRF), V Michel and A Milan (MaxIV), S Gara (Nexeya), P Goryl and L Zytniak (Solaris), F Picca (Soleil), J Krüger (TUM) and J Andreu, S Blanch, R Homs, J Moldes and D Roldan (ALBA) for their contributions to Sardana and Taurus.

References

[1] E.S. Raymond. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly Media, 2001.

[2] T Coutinho, G Cuní, D Fernández-Carreiras, J Klora, C Pascual-Izarra, Z Reszela, R Suñé, A Homs, E Taurel, and V Rey. Sardana: The software for building scadas in scientific environments. *ICALEPCS2011, Grenoble, France*, 2011.

[3] Sardana website. `http://www.sardana-controls.org/`.

[4] C Pascual-Izarra, G Cuní, C Falcón-Torres, D Fernández-Carreiras, Z Reszela, and M Rosanes. Effortless creation of control & data acquisition graphical user interfaces with taurus. *ICALEPCS2015, Melbourne, Australia*, 2015.

[5] Taurus website. http://www.taurus-scada.org/.
[6] David Fernández-Carreiras et al. The design of the alba control system. a cost-effective distributed hardware and software architecture. *ICALEPS2011, Grenoble, France*, page 1318, 2011.
[7] SourceForge website. https://sourceforge.net/.
[8] Andrew Götz et al. The tango controls collaboration in 2015. *ICALEPCS2015, Melbourne, Australia*, 2015.
[9] Tango website. http://www.tango-controls.org/.
[10] C. Pascual-Izarra. SEP0 website. https://sourceforge.net/p/sardana/wiki/SEP0/, 2013.
[11] Debian Enhancement Proposal website. http://dep.debian.net/deps/dep0/.
[12] C. Pascual-Izarra. SEP7 website. https://sourceforge.net/p/sardana/wiki/SEP7/, 2013.
[13] Git website. https://git-scm.com/.
[14] GitFlow website. http://nvie.com/posts/a-successful-git-branching-model/.
[15] GitHub website. https://github.com/.
[16] M. Rosanes-Siscart. SEP5 website. https://sourceforge.net/p/sardana/wiki/SEP5/, 2013.
[17] Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.
[18] Docker website. https://www.docker.com/.
[19] Jenkins website. https://jenkins.io/.
[20] Travis website. https://travis-ci.org/.
[21] AppVeyor website. https://www.appveyor.com/.
[22] Sphinx website. http://www.sphinx-doc.org/.
[23] Read the Docs website. https://readthedocs.org/.
[24] Z Reszela, G Cuni, CM Falcón Torres, D Fernandez-Carreiras, G Jover-Mañas, C Pascual-Izarra, R Pastor Ortiz, M Rosanes Siscart, and S Rubio-Manrique. Bringing quality in the controls software delivery process. *ICALEPCS2015, Melbourne, Australia*, 2015.
[25] Debian website. https://www.debian.org/.
[26] Alioth website. https://alioth.debian.org/.
[27] openSUSE Build Service website. https://build.opensuse.org/.

DOI: 10.17199/NOBUGS2016.26

Addressing the challenges of implementing the ESRF Data Policy

Andy Götz, Alex de Maria, Armando Solé, Roberto Homs-Regojo, Bruno Lebayle, Joanne McCarthy, Jens Meyer, Dominique Porte and Rudolf Dimper

ESRF, 71 ave des Martyrs, 38000, Grenoble, FRANCE

E-mail: andy.gotz@esrf.fr

Abstract. The ESRF, the European Synchroton, has recently adopted a Data Policy which will archive all data collected at the ESRF for 10 years and be made freely available as Open Data after an initial embargo period of 3 years (can be extended on request). Currently the ESRF produces 2 PBs of raw data annually. This means archiving at least 70 PBs of data over the next 10 years if one assumes a linear growth of data production. The Data Policy introduces a number of new challenges for the ESRF. These challenges include persistent user identities, user rights, metadata definition and standardisation, automated collecting of metadata, metadata catalogue, data containers, long term archiving, and finding and re-using data. This paper will describe how these challenges are being solved. The paper describes how it is possible for a mature synchrotron to adopt and implement a modern Data Policy largely built on existing standards like the ICAT metadata catalogue and the HDF5/Nexus data format/convention. Archiving such large quantities of data is largely due to the availability of off-the-shelf tape technology which continues to evolve and improve.

1. Introduction

This paper describes the challenges faced by the ESRF in implementing an open data policy at an existing site which previously left the issues of data management to the users.

2. The Data Policy

The main points of the ESRF Data Policy [1] are :

- ESRF Council officially adopted a Data Policy (1/12/2015)
- ESRF is custodian of data and metadata
- ESRF collects high quality metadata to facilitate reuse of data
- ESRF will keep raw (or reduced) data for 10 years + metadata forever
- Data will be registered in a data catalogue (ICAT [2]) + published with a Digital Object Indentifier (DOI)
- The experimental team has exclusive access to data during the embargo period (3 years but can be extended on request)
- Data will be made public after the embargo period under CC-BY licence
- Data Policy will be implemented on all beamlines by 2020

3. The Challenges

This section will present the different challenges which need to be addressed to implement the data policy.

3.1. Defining and adopting a Data Policy

The first challenge in adopting a Data Policy is to convince the major stakeholders that they need one. Once this step has been achieved the next challenge is to propose or adapt an existing data policy which fits the needs of the stakeholders and the community being served.

The need for having a Data Policy even for a mature institute like the ESRF which has been producing data without an Open Data Policy since 1992 has been driven by the Open Science movement. An increasing number of scientific institutions, universities, governments, and intergovernmental organisation encourage, expect and in some cases require science to be more open and traceable. There are many examples of charters and manifests signed by these bodies which call for Open Science. Open Science for scientific data means Open Data. This has become a requirement for all projects financed by the European Commission. This change in the political landscape convinced the ESRF stakeholders to study the Data Policy question again.

Once it was decided that a Data Policy was necessary the next step was to define a Data Policy. The ESRF participated in the Photons and Neutrons Pandata-Europe project (funded by FP7) where it led the Work Package on Data Policy. The deliverable [3] was a generic Data Policy which took into account the Photon and Neutron community. The Pandata Data Policy was thus close to what the ESRF needs were too. Some modifications were made based on the specific needs of the ESRF e.g. industrial users. The proposal was presented to the Scientific Advisory Committee who requested some changes and then presented to the ESRF Council for final approval. The Council gave their official approval on the 1 December 2015.

The Data Policy was presented to the Users at the ESRF User Meeting in February 2016. Feedback was mostly positive with only a few critical comments concerning the duration of the embargo period. Users were informed they can ask for an extension of the embargo period.

The challenges concerning adopting the ESRF Data Policy were : to convince the ESRF management and scientists and the users that the Data Policy was (a) necessary and (b) feasible. The first challenge was addressed by the changing scientific and politicial landscape with the move towards Open Science and Open Data. The decision by the EC to require Open Data for all H2020 projects since 2015 is a good example of the changing landscape. The users were convinced by the improved metadata, the better logical structure of their data and the additional services like long term archiving and DOIs (see below).

The second challenge about the feasability concerns mainly the cost. This was discussed in two stages - the cost of storing (a) metadata, and (b) raw data. An estimate showed that the cost of storing metadata indefinitely was neglible compared to the cost of available disk storage. It was therefore feasible to have (at least) a metadata policy. The second issue was the cost of long term archiving raw data. The projected storage needs for storing 10 years of raw data are 70 Petabytes assuming current data volume growth. The ESRF has the advantage that it has two high capacity tape libraries on site for backups. With new tapes of much higher density it turns out that 70 Petabytes can be stored with a modest increase in the budget of 100 000 € per year (over 10 years). This is much lower than the cost to produce the raw data.

3.2. Defining Metadata

One of the main goals and challenges of the Data Policy is to collect high quality metadata. The quality of the metadata will determine whether data can be re-analysed or understood by someone who was not part of the team who did the experiment which produced the data. High quality metadata also eases the task of the experimental team to (re)analyse their data. Metadata is often described as data about data or even as data not absolutely necessary for the analysis. In the case of the ESRF Data Policy metadata are defined as those data which describe the experiment.

The challenge of defining metadata is three-fold - what metadata to define, what standards to follow, and finding engineers and/or scientists who are interested in metadata. At the ESRF the decision has been taken to follow the Nexus conventions for defining metadata. Where definitions are lacking new local definitions have been adopted.

Only metadata which is related to data produced at the ESRF is stored. The Data Policy does not attempt to cover metadata produced by processes upstream (or downstream) related to the sample preparation or other processes which impact the experiment. Using the Nexus conventions and maintaining a common set of metadata for all beamlines which use the same technique makes the challenge of defining metadata manageable.

3.3. Collecting Metadata from Experiments

Once the metadata is defined the next challenge is how to collect it automatically on the beamlines. Due to the diversity of beamlines there are a large number of different types of experiments. This makes the challenge more difficult because there is no single solution for all the beamlines and all the experiments.

The solution adopted is to have a generic Metadata server which can be configured via one or more setups in the database to collect the appropiate metadata per experiment and store it in a metadata master file and send it to a metadata catalogue. The metadata server is written in Python as a Tango device server and is configured via properties stored in the Tango database. Tango [4] is the ESRF distributed control system framework. The only remaining difficulty is to introduce the concepts of sample and dataset in the data acquisition sequence. A sample refers to the user samples which are put in the beam. A dataset refers to that group of data files which were acquired on the same sample and can be reduced and/or analysed independently. This needs to be done in the generic data acquisition macros as well as the data acquisition macros which are often beamline specific. Where this is difficult to implement the user is given the possibility to manually decide when to change sample and when to create a new dataset in the data acquisition sequence.

The current solution (configuring the metadata and modifying macros) needs roughly 1 week to get the minimum parameters and up to 4 weeks for all parameters per beamline. This fits in with the timeline of the implementation.

3.4. Choosing a Metadata Catalogue

Another challenge in implementing a Data Policy is the choice of a metadata catalogue. The metadata catalogue is the database and modules which store and allow users to find the metadata and download the raw data. There are a number of catalogues for managing metadata and data. The CRISP FP7 project which the ESRF participated in included a work package 17 [5] for evaluating metadata catalogues. A number of catalogues were compared e.g. ICAT, Dspace, Fedora, Ckan, Invenio, Tardis, ISPyB, iRODS, SRB-MCAT, MS. Zentity. Two of the catalogues, ICAT and ISPyB, were developed in the photon and neutron communities. ICAT was developed at STFC (see talk [6] at this conference) based on a generic data model and ISPyB at the ESRF for Macromolecular Crystallography and BIOSAXS. The other metadata catalogues come from different domains and consequently their data model are quite different.

The challenge of implementing a data policy for all experiments at the ESRF requires a very generic data model which implements the common features shared by all experiments. ICAT with its scientific data model describes the common features of all experiments.

3.5. Defining a Data Format

A challenge for the Data Policy is to get experiments to adopt a common format. This has been a challenge for many sites and ESRF is not an exception. The Data Policy states that data will be made available in a well known format which can be read and interpreted for the next 10 years. In the past the formats were either very simple (ASCII based) or single image files in tiff or simple binary format. With the increasing complexity of experiments, huge data volumes and files and multiple techniques used on the same sample the simple approach is not adapted anymore. For example with this approach some experiments generate millions of files. This is very inefficient when transferring, backing up and/or exporting data. A data format which can store huge volumes of heterogeneous data in a few files is required. Without going into a long discussion the de facto standard for storing this type of data is HDF5 [8]. HDF5 is a binary format developed and maintained by the HDF Group and adopted by the majority of scientific applications which need to store large and diverse data. It allows any kind of data to be stored much like on a file system. Once the data format was chosen the next step is to define or adopt a convention for how to organise the data in the HDF5 file(s). Fortunately the Nexus convention already exists and defines a number of the standards needed. Where standards are missing (especially for beamline specific data and for new techniques) the ESRF has setup a metadata working group locally to endorse new definitions which are defined locally as they are needed.

3.6. Long Term Archiving

An important aspect of the ESRF Data Policy is to archive raw data for 10 years. This opens up a number of possibilities like remote data analysis, persistent identifiers (PIDs), and providing an Open Data policy. Due to the large volume of data (currently a few Petabytes are produced a year) this is quite a big challenge. The challenge is in finding a cost and energy efficient solution for large volumes of data. Tape storage is the most cost and energy efficient solution today. The ESRF has two high capacity tape libraries on site for doing backups. Recent advances in tape density make it possible to offer a solution based on tape only. This has led to upgrading the tape drives and tapes to the latest generation. The tape libraries are now equipped with T10000 tape drives and T2 tapes which have a capacity of 8.5 TB and an average read/write speed of 300 MB/s for large files. This brings the capacity of each tape library to 72 PBs. Data are duplicated in both tape libraries to ensure that two physical copies are kept in physically separate areas.

Tape archives are managed by a tape library and backup software (Time Navigator). In order to be efficient this imposes storing large ($>$ few GBs) files and limiting the number of entries in the backup database. Some basic arithmetic showed that in order to stay within the limits of the backup database over a period of 10 years we need to store on average < 1000 files per shift per experiment (assuming the ESRF continues to operate a similar number of hours as currently and maintains around 40 beamlines). As said above a large number of experiments (roughly 50 %) are above this limit today. Some are much higher ($> 10^7$ files) and we will need to move to using HDF5 before we can archive the data. This can be done in one of two places - at the data acquisition or when the data is archived. Although both solutions are envisaged there is a preference for the first.

3.7. Publishing PIDs

An important added value of the ESRF Data Policy is the generation of Persistent Identifiers (PIDs). A persistent identifier (PID) is a long-lasting reference to a document, file, web page, or other object. Persistent identifiers look like web addresses but with the difference that they are guaranteed to be there in 10 years time. A number of PID schemes exist (ARK, DOI, PURL, URN and XRI). The CRISP project did a survey and proposed to use the DOI system. DOI's consist of a landing page (web page) which contains high level metadata and a pointer to the data. The DOI are particularly useful for publications to be able to cite the data. It is possible to do the reverse and link DOIs to publications thereby tracking number of publications which cite ESRF data.

The main challenge with DOIs is not in implementing them but to decide at what level they should be generated i.e. per Investigation, per Sample or per Dataset. The cost of a DOI is low (cents) however it is important to make DOIs which can be generated with a minimum of human intervention. The solution currently proposed is to automatically generate one DOI per investigation and per sample. In the future users will be able to generate DOIs for a collection of datasets of their choice.

3.8. User Authentication + Authorisation

The archived data need to be kept under embargo for up to 3 years (or longer in special cases) with only members of the experimental team having access. This means all members have to be identifiable with individual ID's over a long period. This challenge is what is known as AAI - Authentification and Authorization Infrastructure. At the ESRF this is being solved with individual user accounts in the local LDAP server with the possibility of users being able to use their Umbrella credentials to authenticate. In the future other authentification mechanisms will be added such as eduGAIN.

3.9. Searching + Finding Data

Open Data are supposed to be FAIR - Findable, Accessible, Interoperable and Reusable [7]. The latter two are related to having high quality metadata and a common data format. The first two are addressed by the metadata catalogue, publications and the Open Data repositories. This challenge is currently addressed by the search mechanisms implemented in ICAT based on the Lucene indexing algorithm ([9]). Currently it is possible to search on Investigation, Sample and Dataset. Searching on Parameters (metadata related to specific dataset) is very basic due to the large variety of parameters and needs extending in the future. The next step is to register the ESRF metadata catalogue with external open data repositories so ESRF data can be found more easily. This is an expanding field and the repositories depend on what are the most widely used ones at the time of registration. One solution is to implement a data harvesting protocol like OAI-PMH which can be used for data searching. The agency managing the DOIs (Datacite) provide some of these features but they could be offered by ICAT directly in the future.

3.10. Exporting Data

A challenge for users is how to get their data home, especially when they have a lot of it. One of the main services offered by ICAT is a data download service called IDS. The IDS currently supports multiple protocols - http, ftp, WebDav, gridftp. IDS retrieves data from the tape archives or disk (if online) to a staging area where the user(s) can download using it one of the above protocols. The ESRF Data Policy does not automatically offer compute resources for re-analysing the data. This service is considered part of the DAAS (Data Analysis as a Service) service which will profit from the implementation of the Data Policy but which will need a separate budget.

3.11. Cost

The expected additional investment cost of implementing the ESRF Data Policy is 100 000 € mainly for the long term storage. An extra 1.5 persons per year is required in human resources to implement the Data Policy. More resources are required to implement the metadata collection on beamlines - up to 1 month per beamline, and to convert data analysis programs to read

HDF5. ESRF has started on this work for specific applications and is developing the **silx** toolkit, a software library which provides builtin support for HDF5 and converts legacy formats (SPEC files) to HDF5 (see silx poster [10] at this conference). Additional work required for implementing single user accounts and authentication has already started and is foreseen to be completed by 2017.

4. Timescale

The goal is to implement the ESRF Data Policy on all beamlines by 2020. This corresponds to roughly 10 beamlines a year. The first beamline to implement the full Data Policy (including DOIs and long term archiving) is expected to be at the end of 2016.

5. Conclusion

The adoption of an ESRF Data Policy has put data management on the front of the ESRF agenda. The first group of people to benefit from this are expected to be the users themselves. The Data Policy is only the start of data management, it opens a number of new possibilities like Remote Data Analysis as a Service, data citing, data re-use to mention a few. These services will be possible to implement in the future largely due to the Data Policy.

5.1. Acknowledgments

Thanks to the ICAT development team @ STFC for providing a robust metadata catalogue. Thanks to the system admins at the ESRF for providing a powerful infrastructure. Thanks to those scientists who have helped to define and improve metadata.

6. References

[1] ESRF Data Policy `http://www.esrf.fr/files/live/sites/www/files/about/organisation/ESRFdatapolicy-web.pdf`
[2] ICAT project web site `http://icatproject.org`
[3] PaNData Europe Data Policy `http://wiki.pan-data.eu/imagesGHD/0/08/PaN-data-D2-1.pdf`
[4] TANGO Controls web site `http://tango-controls.org`
[5] CRISP WP 17 `http://www.crisp-fp7.eu/research-programme/work-packages-descriptions/wp17/`
[6] Fisher, S et. al. *Growth of the ICAT family*, NOBUGS 2016 conference (Copenhagen)
[7] Wilkinson, M. D. et al. *The FAIR Guiding Principles for scientific data management and stewardship.* Sci. Data 3:160018 doi: 10.1038/sdata.2016.18 (2016).
[8] HDF5 home page `https://support.hdfgroup.org/HDF5/`
[9] Lucene project web site `https://lucene.apache.org/`
[10] Solé, V.A. et. al. *The silx toolkit*, NOBUGS 2016 conference (Copenhagen)

DOI: 10.17199/NOBUGS2016.57

Scientific data lifecycle at Elettra-Sincrotrone Trieste.

Milan Prica, Fulvio Bille`, Roberto Borghes, Valentina Chenda, Alessio Curri, Daniele Favretto, Georgios Kourousias, Roberto Pugliese, Martin Scarcia, Michele Turcinovich

Elettra-Sincrotrone Trieste S.C.p.A. Strada Statale 14 - km 163,5 in AREA Science Park 34149 Basovizza, Trieste, Italy

milan.prica@elettra.eu, roberto.borghes@elettra.eu, george.kourousias@elettra.eu

Abstract. Elettra-Sincrotrone Trieste comprises Elettra synchrotron and FERMI free-electron laser. Combined, the two facilities serve 34 beamlines. The Scientific Computing Team supports the full data lifecycle. Proposal submission and evaluation are handled in the Virtual Unified Office. Data acquisition and experimental control are built on top of the TANGO control system on all but the oldest synchrotron beamlines. Data reduction, on- and off-line analysis workflows and visualisation are supported by a common framework. Data is stored and catalogued with access through the web portal. Elettra implements the PaNdata-like data policy since 2014.

1. Introduction

Synchrotrons and especially free electron lasers facilities produce enormous quantities of scientific data during their normal operations. With advances in modern instrumentation and in particular with the newest, fastest, high resolution detectors, the problem of data deluge often arises. Besides, facilities handle a large quantity of sensitive user information and invaluable intellectual property contained in the research proposals. Managing this high volumes of scientific data and their ownership crosses different application systems, databases and storage media, often beyond the facility boundaries. The data lifecycle at Elettra facilities [1] is a complex process that begins with a research proposal and follows through a series of steps including laboratory access, data acquisition, meta-data correlation, pre-processing, storage, setting access rights, retrieval, analysis and publications.

2. A brief overview of the facilities

Elettra-Sincrotrone Trieste runs two advanced light sources: a third-generation synchrotron Elettra and a single pass Free Electron Laser (FEL), FERMI. Since the major upgrade of the facility in 2010, Elettra SRF operates in a top-up mode at two energies: 2.0 GeV for enhanced extended ultraviolet performance and spectroscopic applications (75% of user time) and 2.4 GeV for enhanced x-ray emission and diffraction applications (25% of user time). A vast range of research in physics, chemistry, biology, life sciences, environmental science, medicine, forensic science, and cultural heritage is carried out at 28 beamlines. A substantial upgrade of the Elettra machine is planned [2] in the near future.

The FERMI FEL design is based on an external seeding scheme that improves the output pulse coherence, central wavelength control and spectral bandwidth. The FEL output is tunable in power, wavelength, temporal duration, and polarization. With a peak brightness of about 6 orders of magnitude higher than third generation sources, and pulse lengths of the order of a picosecond or less, FERMI supports scientific investigations of ultra-fast and ultra-high resolution processes in material science and physical biosciences. FERMI is running at 50 Hz pulse rate since 2015. Currently six beamlines are operational.

3. User registration and proposal submission

Virtual Unified Office (VUO) is the main users' interface to the facilities. Constantly evolving since 1997, the web portal manages all the steps that together form an experiment, beginning with the user registration and a proposal submission and following through the elaborated results and publications. Besides Elettra and FERMI, the VUO portal manages also the facilities of the CERIC-ERIC [2] consortium.

The process starts with the creation of the user account. A user's login credentials will be valid not only on the web portal but also on the acquisition workstation, on the data storage devices, on the data analysis computational cluster, on the remote operations tools and for the Wi-Fi access. The authentication and authorization system dedicated to the scientific users is separated from the main company one. Umbrella authentication system is supported on the web portal. The proposal evaluation process is entirely managed in the portal, from proposal submission to the beamtime scheduling for the accepted ones.

Elettra was among the first light source facilities to adopt an official data policy in 2014. Users must agree to the data policy as part of the proposal submission procedure. The Elettra implementation is a soft version of the PaNdata proposal. A three-year embargo period is easily (automatically) extendable and while no data have been deleted to date, no hard guarantee is offered for long term data preservation. DOIs on the datasets are not issued at the present moment but that is likely to change in the future.

4. Data acquisition

The beamline end-stations of Elettra and FERMI are complex instruments, each consisting of a large number of interconnected components. Each experiment requires a long sequence of operations on most of these components that include among others valves and shutters, vacuum pumps, positioning motors, and a large variety of sensors, cameras and detectors. The data acquisition software should be capable of commanding all of these components while at the same time enabling the user to monitor the process through a clear graphical interface. A flexible, extensible and easily adaptable end-station control software based on the Tango distributed control system has been developed [3]. Its key components regard the automatization of the experimental sequence (Executer) and a fast acquisition system (FDAQ).

The Executer application is a highly flexible PyTango device that executes generic Python functions from scripts located in external files. These scripts can call external programs and functions written in other languages. Input variables and result viewers are handled as dynamic Tango attributes configurable via an XML configuration file. Each experimental sequence may be coded as an Executer's function.

The system for the data acquisition at FERMI had to be designed with the maximum regard for the high speed and the pulsed nature of a FEL source. The FDAQ application is PyTango device consisting of multiple threads (one for each data source) that collect and store shot-to-shot data and the related meta-data from a large number of Tango sources. Data and meta-data sources are configurable through an XML file. Elettra beamlines typically present much simpler, custom developed data acquisition systems. Except for the oldest beamlines, the end-station controls are Tango based. The experimental data and metadata are organized and saved in the HDF5 binary scientific data format. The HDF5 structure is custom at each beamline and no beamline has adopted NeXuS at the present moment. At FERMI, a typical experimental dataset is composed of multiple HDF5 archives containing data relative to contiguous FEL shots. FDAQ stores data in a fast local storage called scratch.

The main graphical user interface (GUI) to the data acquisition software presented to scientists has been developed in QTango, a C++ based framework. Just like the other key components of the system, it is highly customizable through an XML configuration file that defines both the content and the appearance of the front-end panel. The main section of the panel is a tabbed panel relative to the experiment. Each tab contains input and output variables and the command buttons of the corresponding Executer script. Another section is relative to the FDAQ application and contains a dynamic and checkable list of the available data sources. Finally, there is a common zone for data viewers and the additional command buttons. To access the panel, users must login into the acquisition system with their VUO credentials and choose a proposal to which the data collection is related.

4.1. Remote access

A number of tools are available for safe remote access to selected beamline controls and data analysis tools. Remote access allows for collaboration between researches present at the site and those following the experiments from their home institutes and guarantees prompt interventions from beamline scientist in case of necessity. However, the remote access presents high security risks and must be handled with utmost care. VUO integrates a Java-based tunneling tool with the authentication and authorization mechanism in support of RDP, NX and VNC.

4.2. End-user tools

The Scientific Computing team has built a collection of software tools [4] with a high level of customization reusable in a multitude of situations in support of user-oriented applications at the end-stations. These tools include an electronic logbook built on top of a HTML-based, WYSIWYG text editor that supports screenshot insertions and integrates a data server for upload of images and log messages from remote machines. Files are saved automatically and exportable to PDF format. Another important tool in the collection is a standard visualizer that supports a number of common scientific image formats like CBF, TIFF, MAR345, etc. It provides functionalities like region-of-interest zooming, line profiling, automatic or manual contrast adjustments and background subtraction. The visualizer tool is used on many beamlines of Elettra and FERMI for online data analysis of images from Dectris Pilatus detectors, mar345 imaging plates, Princeton Instruments CCDs, Basler and Andor cameras. A common necessity to investigate a new experimental techniques inspired the development of a tool for fast prototyping of data collection sequences and implementation of data analysis pipelines. Built as an extension module to Python Spyder IDE it integrates an interactive help system and numerous custom libraries developed and tested by the specialized staff. Plug-in based extensibility is common to all of the above tools.

5. Storage and data access

At some point, the collected data has to be moved from the fast scratch storage to the accessible online storage. This step may be performed manually from the VUO, but it is most often done automatically as part of the data acquisition pipeline. Some data preprocessing like the lossless compression on HDF5 files may be executed at the same time. Data is moved with *rsync* and the transfers are coordinated by the dedicated Tango devices. Tango devices take care of the creation of the necessary directories and setting of the POSIX access rights. Once moved to the online storage, the raw data files cannot be further modified. Selected users may access data in the online storage following a simple proposal / dataset / datafile scheme. The principal investigator and the responsible beamline scientists may enable anyone with a valid VUO account to access the data of the proposal. Data are accessible through WebDAV and through web browsers. Multiple files or entire datasets (folders) may be downloaded as a single zip file. The directory tree created provides a placeholder for elaboration results to which files may be uploaded. The E-logbook file generated during the beamtime is stored in the PDF format with the dataset. If and when necessary, data may be moved to long term (offline) storage at the CINECA supercomputing center. CINECA system is based on iRODS. Since the beginning of the PaNdata-ODI project, Elettra has been following the ICAT metadata catalog development. A move to ICAT will likely be necessary to implement open data access but the final decision is yet to be made. A purchase of some additional storage space for the online storage is imminent.

6. Conclusion

Implementing a complete data lifecycle in a large experimental physics facility, as is the case with Elettra, is a very complex and expensive task. A large number of heterogeneous systems have to work in concert while new pieces are constantly added to the puzzle. There are numerous attempts from international collaborations to provide solutions that address common challenges related to the Big Data and the open data access. The planned Elettra 2.0 upgrade will provide us an important window of opportunity to implement some of those advances.

References

[1] Billè F, Borghes R, Brun F, Chenda V, Curri A, Duic V, Favretto D, Kourousias G, Lonza M, Prica M, Pugliese R, Scarcia M, Turcinovich M Data Lifecycle In Large Experimental Physics Facilities: The Approach Of The Synchrotron ELETTRA And The Free Electron Laser FERMI *Proceedings of ICALEPCS2015,* Melbourne, Australia

[2] Karantzoulis E, Elettra 2.0 – The next machine, *Proceedings of IPAC2015,* Richmond, VA, USA

[3] Borghes R, Chenda V, Curri A, Kourousias G, Lonza M, Passos G, Prica M, Pugliese R, A common software framework for FEL data acquisition and experiment management at FERMI@Elettra, *Proceedings of ICALEPCS2013*, San Francisco, CA, USA

[4] Borghes R, Chenda V, Kourousias G, Lonza M, Prica M, Scarcia M A Flexible System for End-User Data Visualisation, Analysis Prototyping and Experiment Logbook, *Proceedings of ICALEPCS2015,* Melbourne, Australia

DOI: 10.17199/NOBUGS2016.45

The growth of the ICAT family

Stephen M Fisher[1], Frazer Barnsley[1], Wayne Chung[1], Sylvie Da Graca Ramos[2], Alex De Maria[3], Rebecca Fair[1], Andy Gotz[3], Tom Griffin[1], Rolf Krahl[4], Brian Matthews[1], Peter Parker[5], Kevin Phipps[1], Alex Potter-Dixon[1], Milan Prica[6], Christopher Prosser[1], Jianguo Rao[1], Shelly Ren[5], Brian Ritchie[1] and Jody Salt[1]

[1]Rutherford Appleton Laboratory, Didcot, OX11 0QX, UK
[2]Diamond Light Source Ltd, Diamond House Harwell Campus, Didcot, OX11 0DE, UK
[3]ESRF, 71 Avenue des Martyrs, 38043 Grenoble, France
[4]Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, Albert-Einstein-Straße 15, 12489 Berlin, Germany
[5]Oak Ridge National Laboratory, 1 Bethel Valley Rd, Oak Ridge, TN 37831, United States
[6]Elettra - Sincrotrone Trieste S.C.p.A., Strada Statale 14, Km 163.5, 34149 Basovizza, Trieste, Italy.

E-mail: `dr.s.m.fisher@gmail.com`

Abstract. The ICAT project provides a metadata catalogue and related components to support Large Facility experimental data and aspires to link all aspects of the research chain from proposal through to publication and can also be used to provide an implementation of a data policy as has been done at a number of facilities using ICAT. Over the last couple of years, the existing components of ICAT have seen improvements in functionality and performance. TopCAT, a GUI to work with multiple ICATs, has changed dramatically preserving only the original concept and new components have been added to provide flexible data delivery solutions and to make an ICAT installation easy to manage. These changes have been made in consultation with the ICAT community to ensure that the components are highly decoupled and that as far as possible backwards compatibility is maintained as more sites move their ICAT installations into production.

1. Introduction
ICAT [1] is based on three fundamentals: a data model, a data catalogue and a GUI to provide a good user experience. The CLRC Data Portal [2] was prototyped and reported on in 2001. A number of the ideas introduced at that time have been preserved though many of the technologies used today in the ICAT family did not exist then.

2. Early history
The CLRC Data Portal of 2001 made use of a data model that evolved into something that was published in 2004 as the CSMD (CCLRC Scientific Metadata Model: Version 2 [3]) and which has subsequently evolved [4]. The original catalogue was relational and had the idea of wrappers to extract metadata from the data dynamically but not to store it. This avoids duplication of data but is potentially very expensive. The metadata catalogue developed further and became an open-source project in 2008 [5, 6]. It was deployed as a SOAP based web service running

under Glassfish. The API, having specific operations to manipulate each table, became very large. The authorization scheme was ACL like and required the storage of a lot of extra data in the catalogue. A major rewrite was performed four years later in 2012 when the complex API was replaced by a generic interface with just a few calls related to the usual CRUD operations that were being performed on the underlying database.

The ICAT Data Service, IDS, was started in 2013 with the aim of being an interface to ICAT catalogued data files. It was inspired at some level by an earlier component: the Downloader. The Downloader was written to meet the needs of one facility and was only able to build zip files of data to download. Its successor, the IDS, has a clean plugin architecture to make it suitable for any facility and has many extra features as explained below.

Work on TopCAT as a GUI to view multiple ICATs started in 2011. Two years later, in 2013, work began on the ICAT Job Portal, IJP, a system allowing data stored in the IDS and catalogued in ICAT to be submitted to compute resources.

The ICAT community now has monthly teleconferences, an annual face-to-face meeting and is governed by a steering group which meets a couple of times a year.

3. Established components

The server components all run in a Java EE container such as Glassfish or WildFly.

3.1. ICAT Server

The ICAT server is a metadata catalogue to support primarily Large Facility experimental data, linking all aspects of the research chain from proposal through to publication. It provides SOAP and REST web service interfaces to an underlying database via easy to use APIs. It has powerful search features, a rule based authorization mechanism and it uses plugins for authentication.

3.1.1. Database and Schema The primary database is relational. The ICAT server should in principle work on any relational database which supports transactions and has a JDBC driver available. Best understood are MariaDB/MySQL and Oracle. Some of the information held in the relational database is also indexed in Apache Lucene [7] which is a fast text search engine. This gives the benefit of the relational model for keeping the data organised and at the same time allows users to find things when they are not sure where to look.

The schema is designed to be as regular as possible. All relationships are one to many and are cascaded in the one to many direction. This means for example that if you delete a Dataset then all its Datafiles are deleted too. Entities are identified by an object in the many to one direction and one or more naming fields. For example a Datafile is identified by its Dataset and a name. This also means that a Datafile cannot exist without a Dataset and that it can only be 'part of' one Dataset.

3.1.2. Authentication Plugins An authenticator implements a small interface which allows the ICAT server to authenticate and to find out information about the authenticator. Each plugin is deployed as a separate application in the Java EE container and is accessed by the ICAT server with remote calls. Each authenticator accepts a map of key names to key values where typical key names would be 'username' and 'password'.

3.1.3. Accessing the service Originally the ICAT Server only exposed a SOAP interface but now it has a REST interface as well.

The convenience of using the SOAP interface depends critically upon the level of support provided by the available libraries. Support in both Python and Java is good. The python-icat library is a Python package that provides a collection of modules for writing programs that access an ICAT service using the SOAP interface by extending Suds [8] with ICAT specific features and also providing a level of protection from server version dependency. In addition the ICAT Manager is a standalone Java client based on the Eclipse Platform for visualising and managing ICAT instances. It works for all recent versions of the ICAT server thanks to a JAX-WS dynamic client and allows display, editing and creation of any ICAT entity subject to the authorization rules.

Though the SOAP interface is easy to use, it is somewhat restricted and there is a tendency to bring back to the client side information which is not needed because it brings back whole rows from the database. The REST interface uses the HTTP(S) methods PUT, DELETE, POST and GET as appropriate. The REST interface is much more efficient because queries can be written in JPQL to return exactly what is wanted. Detailed documentation for each call is generated from the server code. Small client libraries are provided in Java and Python and mainly look after error handling. The Python API is the more convenient to use because instead of dealing with JSON strings you pass nested Python dicts and arrays.

3.1.4. Authorization Authorization is entirely rule based. Though rules can be related to a specific row of a table in an ACL style this is not the way they are normally expected to be used. The intention is that rules can be defined to implement a data policy, for example, to say that if you are related to an Investigation then you can see all the data related to that Investigation. You could also define a rule to say that all Datafiles older than some time are public. The authorization system is such that if *any* rule allows the user to perform the action then it will be allowed. There are no rules forbidding operations. The system is efficient because the rules are used in a way that allows the database to do most of the computation.

3.1.5. Calls There are very few calls and, apart from three REST calls for TopCAT to utilise the Lucene index, they are schema independent. The API is providing a generic approach to accessing a relational database which follows a schema with a few special constraints as described earlier. Some tables stored in the database are however special and can affect subsequent operations. In particular the Rules table which controls authorization is populated and queried like any other table but controls access to all ICAT operations.

REST calls support import/export of the contents of the ICAT database. These operations like all the others are subject to the authorization rules.

3.2. IDS Server

This component provides an 'ICAT friendly' interface to data storage. The IDS stores the data file itself and catalogues it as a Datafile object in the ICAT server. Calls are provided to store an individual data file and calls to get, query the status of, and delete groups of data files as specified by the IDs of Investigations, Datasets and Datafiles. TopCAT makes use of the IDS to download data.

It is also possible to configure the IDS to be used in 'Read Only' mode where data files are stored by some preexisting facility mechanism and the Datafile objects must then be created in ICAT to reference these files.

3.2.1. Two Level Storage The IDS can be configured to use a single level of storage where data are all available with low latency (e.g. disk) or it may be configured to use two levels of storage known as main and archive. The main storage should have low latency and the archive storage might have higher latency (e.g. tape). If the volume of data makes it not practical to hold all data on low latency storage then two level storage must be used. All of the calls to the IDS may be used irrespective of whether single or two level storage is used, however the archive and restore calls are ignored for single level storage, and the prepareData call (which ensures that data is brought back from archive to main storage) has no value.

When a file is written to the IDS it is first stored on main storage. Later it may be copied to archive storage if a two level storage model has been adopted.

Though the server provides explicit archive and restore calls, the movement of data between main and archive storage is normally handled automatically. The server configuration includes high and low watermark levels for the size of main storage. A background process notices if the high watermark is exceeded and requests the archiving of sufficient files to bring the main data storage size down to the low water mark. A restore operation will be queued if an attempt is made to access data that is not in main storage.

3.2.2. Plugin architecture Different facilities have different needs for external file structures. To cope with this, interfaces have been defined which must be implemented by a plugin written specifically for your facility. There are interfaces for main storage, archive storage and to define the entry names in a zip file when a group of files are downloaded in one call to the IDS.

3.2.3. ICAT Coupling ICAT session ids are passed as an argument to most calls. The server checks for READ access to the referenced data by a suitable call to ICAT. For put and delete operations the server checks for CREATE and DELETE access to the referenced data and makes the corresponding changes to ICAT to catalogue (create a Datafile entry in ICAT) or uncatalogue the file. The server is coded to maintain consistency with ICAT. In the case of a failure of software or hardware an orphan file may exist but there should never be an entry in ICAT for which no file exists.

3.2.4. Accessing the service The IDS server exposes a REST interface using the HTTP(S) methods PUT, DELETE, POST and GET as appropriate. Detailed documentation for each call is generated from the server code and comments in that code. IDS clients are provided in Java and Python.

3.2.5. Calls Many calls accept lists of IDs of Investigations, Datasets and Datafiles. The operation to store a file needs an existing Dataset in ICAT to link it to. From a server perspective the data is streamed in the body of the message. After the call the data stream will have been stored as a file and will be catalogued as a Datafile. When retrieving multiple files a zip file will be used to wrap them. As soon as the server has checked that the data are all available then streaming of the result will start. There is a 'getLink' call to provide efficient access to a data file if the file system hosting main storage is accessible to the user. It returns an absolute path which is a hard link to the file which has a server defined lifetime. It is expected to be used from a program which calls getLink and then immediately opens the file. Even if the hard link is deleted by the server the file remains accessible because of the open file handle. ACLs are set on the link to only allow read access to the user specified in the getLink call.

3.3. TopCAT

This is a web based GUI able to search across one or more ICAT server instances and download data via the corresponding IDS server. Figure 1 shows how those ICAT components mentioned so far might be used in conjunction with TopCAT. The browser communicates with TopCAT which searches in the ICAT server (here labelled simply as ICAT). It can provide links to the IDS to download specific files located by their metadata stored in the ICAT server. Each IDS sends messages to its own ICAT server to make sure that any requested operations are authorized. The IDS at the top of the figure is using an IDS plugin which communicates with a single storage

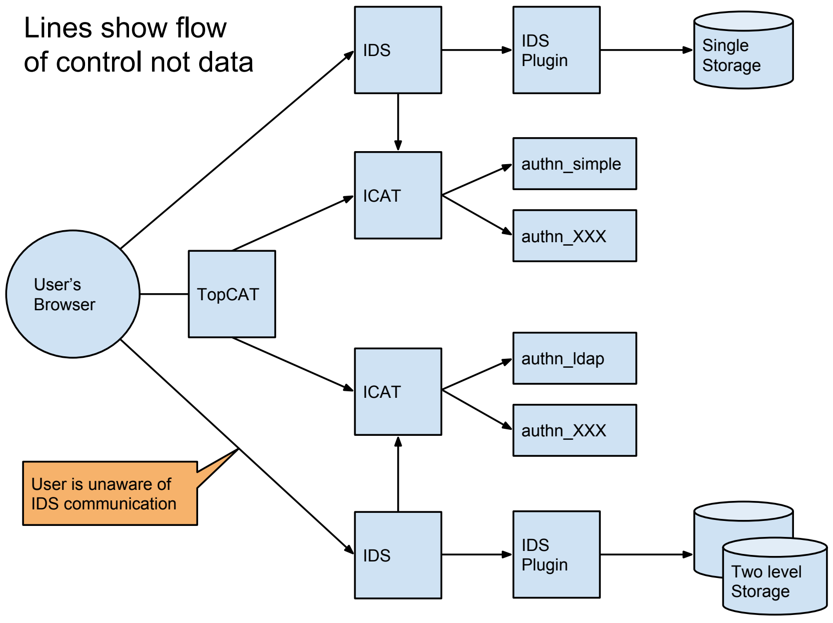


Figure 1. Block diagram showing some components

system. The IDS at the bottom is using a plugin able to communicate with two level storage: main and archive. Each ICAT server is shown with two authenticators.

4. Recent developments

4.1. TopCAT

TopCat has already been mentioned as an established component however it has recently had an overhaul so major that very few lines of old code remain. A major refactoring effort to replace the GWT [9] by AngularJS (version 1) and bootstrap began. This took into account a requirements gathering exercise mostly based on producing mockups and inviting comments. TopCAT now includes a configurable set of data delivery mechanisms in addition to http(s). One of these is the smartclient and the others are provided by PollCAT both of which are described below.

4.2. smartclient

The smartclient is a small server packaged as a standalone program. The user downloads this and starts the server which listens for requests from TopCAT to download some data and queues a list of Datafiles to be downloaded. It then uses a number of threads to process the queue so that it is doing a number of downloads in parallel from the IDS to local storage. It is a convenient way to transfer a large amount of data to your desktop machine.

4.3. PollCAT

When PollCAT is asked to perform a download it polls the IDS until the data is ready, then transfers it to a location determined by a plugin. Various plugins exist: for example to write to PanFS [10] or to a Globus Connect Server [11]. PollCAT is a third party transfer mechanism.

4.4. ICAT Job Portal

The IJP [12] was started in 2013. It makes use of the ICAT server to locate data to process and can then submit and manage compute jobs for interactive or batch processing. These jobs obtain their data from the IDS and write new data into the IDS. A job may record provenance information in the ICAT server for future reference. Recently it was decided that this should also migrate towards AngularJS and bootstrap. While prototyping this refactoring it was realised that the code would have a lot in common with TopCAT; TopCAT finds data and downloads it, the IJP finds data and submits it for processing. It was then decided that the most efficient way forward was to add a plugin mechanism to TopCAT, which has very recently been done. The plugin mechanism added to TopCAT also allowed a DOI (Digital Object Identifier) plugin to be written. This communicates with another service to obtain DOIs.

4.5. Dashboard

The Dashboard is a web based GUI to give an overview of ICAT usage. It subscribes to JMS messages transmitted by ICAT and IDS servers and also itself makes calls to ICAT. The information is stored in a database. The Dashboard has a number of displays to show information about users and where they are and about volumes of downloaded data. As a new component this was written from the beginning with a Java back end and an AngularJS front end and will soon be released.

5. Conclusion

The CSMD data model has evolved and has proved to be applicable to many different facilities. The ICAT components have been designed to be loosely coupled and to make use of plugins where extensibility is required so that they now meet the needs of a growing number of facilities. The ICAT project is very much alive with well attended monthly virtual meetings and considerable discussion outside the meetings in the ICAT forum.

References

[1] The ICAT Collaboration. The ICAT Project, accessed September 27, 2016. `https://icatproject.org` doi:10.5286/SOFTWARE/ICAT.

[2] Matthews B M, Ashby J V, Bicarregui J C, Boyd D R S, Kleese van Dam K, Lambert S C, and ONeill K D. The CLRC Data Portal. *ERCIM News*, (45):39–40, 2001.

[3] Sufi S and Matthews B M. CCLRC Scientific Metadata Model: Version 2. Technical Report DL-TR-2004-001, Daresbury Laboratory, 2004.

[4] Fisher S M and Matthews B M. CSMD: the Core Scientific Metadata Model, accessed September 26, 2016. `http://icatproject-contrib.github.io/CSMD/csmd-4.0.html`.

[5] Flannery D et al. ICAT: Integrating data infrastructure for facilities based science. In *5th IEEE International Conference on e-Science*. IEEE, December 2009. `http://purl.org/net/epubs/manifestation/9367`.

[6] Matthews B M, Sufi S, Flannery D, Lerusse L, Griffin T, Gleaves M, and Kleese van Dam K. Using a Core Scientific Metadata Model in large-scale facilities. In *5th International Digital Curation Conference*, December 2009. `http://purl.org/net/epubs/manifestation/4837`.

[7] Lucene, accessed October 3, 2016. `https://lucene.apache.org`.

[8] Gospodnetic J. Lightweight SOAP client (Jurko's fork), accessed September 27, 2016. `https://pypi.python.org/pypi/suds-jurko`.

[9] GWT, accessed September 27, 2016. `http://gwtproject.org`.

[10] PanFS, accessed September 28, 2016. `http://www.panasas.com/products/panfs`.

[11] Globus Connect Server, accessed September 28, 2016. `https://www.globus.org/globus-connect-server`.

[12] Fisher S M, Phipps K, and Rolfe D J. ICAT Job Portal: a generic job submission system built on a scientific data catalog. In Tamas Kiss, editor, *IWSG 2013: 5th International Workshop on Science Gateways*. CEUR-WS, June 2013. `http://ceur-ws.org/Vol-993/paper6.pdf`.

DOI: 10.17199/NOBUGS2016.65

Building a prototype Data Analysis as a Service : the STFC experience

Frazer Barnsley, Brian Matthews, Tom Griffin, Jody Salt, Derek Ross, Catalin Condurache, Alexander Dibbo, Shirley Crompton
STFC Rutherford Appleton Laboratory, Chilton, Didcot, OX11 0QX, UK

E-mail: frazer.barnsley@stfc.ac.uk

Abstract. In this paper, we describe the motivations behind the design of the Data Analysis as a Service platform, under development within STFC. This is a service aimed at providing a platform for users to access compute and software resources suitable for analyzing experimental data arising from the use of large analytic facilities, for example ISIS and Diamond Light Source. We give some requirements for the use of such a platform, and then go on to discuss its initial architecture. We then describe the current state of the implementation of this architecture with an initial example of its use. We conclude with a discussion on the future direction of this development,

1. Introduction

Modern instruments and detectors are capable of capturing large amounts of data in one scan, and experiments are becoming more sophisticated, with multiple techniques applied concurrently or dynamic structures such as chemical reactions being observed. Data volumes have now grown so large that in many cases it is simply not practical for users to transport the data to their home institution. In other cases, the analysis chains are complex, with a combination of data analysis and simulation, requiring access to high performance computing, large memory machines and a complex software stacks for effective and timely processing of data. These resources and expertise may not be consistently available to all users. As a consequence, many facilities are exploring how to best provide additional computing resources to enable users to access and analyse their data remotely. At STFC Rutherford Appleton Laboratory, there is co-location of large-scale facilities with a dedicated data-centre hosting large-scale data archives, computing capabilities and specialist expertise. This gives the opportunity to coordinate into one service access to in- and post-experiment computing support for facility users to aid those users to interpret their experimental data.

STFC's Scientific Computing Department is working with the RAL based facilities (ISIS, Diamond, CLF) to implement and deploy a *'Data Analysis as a Service'* system (DAaaS) to support this service. Such a system provides facility users with easy to use access to compute resources, collocated with the experimental data archives, to efficiently and easily process their data, within a managed, secure virtual environment. Commonly used software packages will be systematically made available via a deployment and configuration system, and the environment offered to users will be customised according to the nature of the experiment and requirements of the experimental team. The system will further support a number of interfaces, allowing both easy access to users for routine tasks as well as a more interactive environment for more specialised usage.

In this paper we describe some requirements that such a DAaaS platform should be expected to satisfy. We then go on to describe the architecture of our approach, and our experience of configuring and deploying a prototype DAaaS for facilities at STFC's Rutherford Appleton Laboratory. Finally, we discuss our plans to extend and develop this system to provide a production environment, covering a range of analytic techniques to users within one service

2. Requirements

The high-level requirements the DAaaS platform should satisfy include the following.

1. Users should have access to their experimental data and compute resources for data analysis during their visit and after they return to their home institution.

2. User access should be secure; they should be able to access the data for their experiments and only their experiments. Other users should not be able to monitor a user's activity within their data analysis environment.

3. Users should be provided with a suitable software environment. That is, current analysis, visualisation, workflow and other software appropriate to their scientific approach should be made available to them systematically within the environment, with suitable licencing. Users should be able to upload their own software packages; however, the quality of such software should not impact other users.

4. Users should be able to access the compute resource appropriate for the current task. Thus, if they are in-visit it may be appropriate to access instrument workstations or facilities clusters, while post-visit they should be able to access data-centre compute resources. If the compute task becomes too large for one system, they should be able to submit jobs onto a HPC cluster.

5. As users access different compute resources, the data available to them should be consistent – that is, the current working data should move as the user accesses different platforms transparently to the user (subject to latencies in moving the data).

6. Users should be provided with user interfaces (virtual desktops, specialised clients, web-clients) suitable to their science domain, community knowledge base and level of expertise.

7. The system providers should be able expand the service to additional experimental user groups at marginal cost – that is, it should be a software deployment and configuration task to introduce new groups rather than a complete redevelopment. Thus the system can be efficiently delivered across many communities.

These requirements are demanding; we propose that the most appropriate basis to build a DAaaS system satisfying these requirements is on a loosely-coupled and expandable cloud-based platform.

3. Design of the DAaaS

We discuss the overall design of our prototype DAaaS. A cloud-based solution can provide the basic platform; this would provide an environment which can be remotely accessed, be securely insulated from other users, and be provisioned with access to the relevant data and software. The existing capability within Scientific Computing Department can provide such a platform, which we are using to develop the DAaaS. An outline of the DAaaS architecture is given in Figure 1.

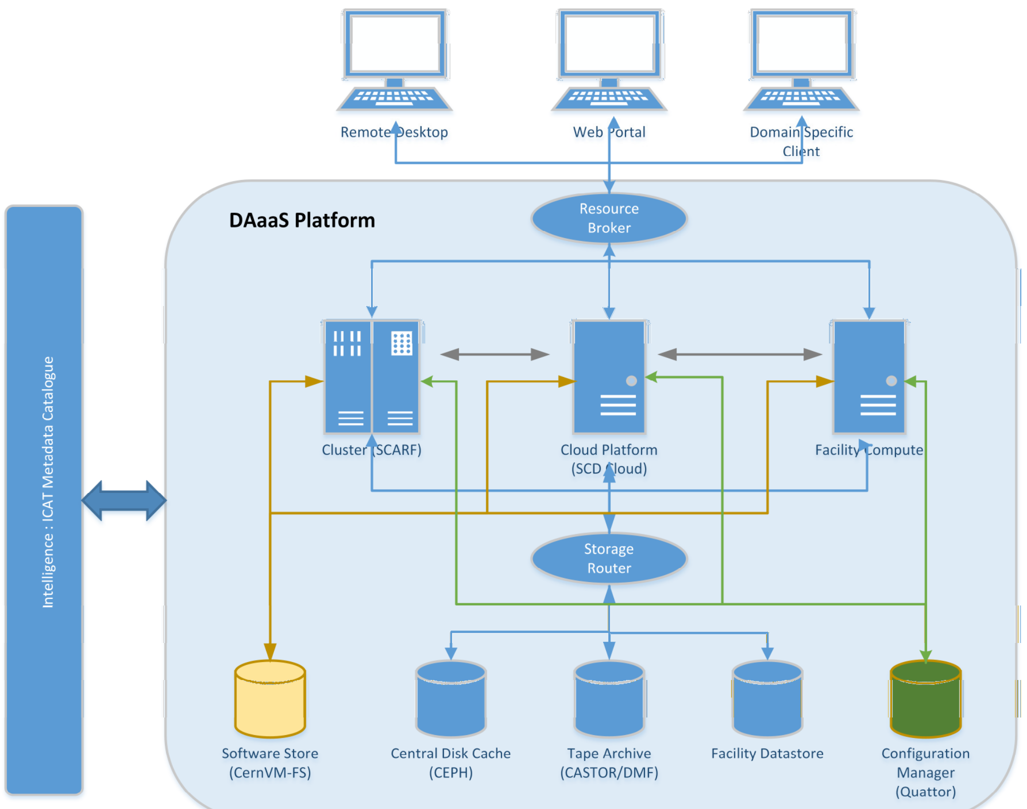

Remote Desktop
Web Portal
Domain Specific Client
DAaaS Platform
Resource Broker
Intelligence : ICAT Metadata Catalogue
Cluster (SCARF)
Cloud Platform (SCD Cloud)
Facility Compute
Storage Router
Software Store (CernVM-FS)
Central Disk Cache (CEPH)
Tape Archive (CASTOR/DMF)
Facility Datastore
Configuration Manager (Quattor)

Figure 1 : Outline architecture of Data Analysis as a Service

The architecture can be split into three main layers:

Storage Layer: the "data backplane" with the persistent data storage available to all the tools, services and software within the DAaaS, on disk or tape, located centrally or within facilities (e.g. on instruments). Data should be maintained consistently between them, and made available to the compute services.

Compute Layer: the compute services which are available to the users, with a number of different compute platforms. The user should be able to move between them easily and have their data made available on local caches, consistent with their last use of the data.

User Interface Layer: a number of different user interfaces should be made available to the users, depending on their familiarity with the systems and the nature of the interactions which they would require with the Platform. Thus we can provide a direct access via a remote desktop, giving the user highly flexible and customisable access to the platform. Further, we can provide more controlled environments via web portals or domain specific user clients.

The Platform would also integrate services which customised and configure the platform to the needs of the particular user domain, and also allow the deployment of common software packages, so that users have access to consistent versions of software within the different compute platforms, with appropriate libraries and configuration.

The Platform includes a notion of a controlling source of “intelligence”. That is, it uses information on the users, users’ experiments, and experimental domains to contextualise the service. This controls access to users to their experimental data, but also guides the system to provide access to appropriate services. For example an experimental group in imaging would be given access to suitable reconstruction software, with licensing appropriate to the user, and use of resources such as high-performance clusters suitable to the scale of reconstruction, and access to workflows and visualisation tools. The system will also maintain the information, collecting provenance data as the users process their data to provide an audit trail for tracking and restarting progress through the analysis, for replaying workflows and for validating results. The source of intelligence within the STFC DAaaS is the ICAT metadata catalogue [1], a catalogue of facilities experiments and associated data.

4. The Technology Stack of the DAaaS prototype

The technology stack we are using to prototype the DAaaS system is guided by our existing experience with Cloud systems, distributed storage systems such as Ceph, software distribution and packaging using CernVM-FS, and our experiences of different methods for providing remote desktop like services. We give an overview of the status of the components used in the DAaaS.

Compute Layer: The platform currently has two main compute resources - a cloud compute cluster, based on OpenNebula [7] and SCARF, a batch system based on IBM's Platform LSF [8]. The cloud has 896 processing cores and 3.5TB of memory and provides users with interactive analysis environments via customised virtual machines (VM). The SCARF batch system has approximately 6000 cores coupled with 300TB high speed storage based on PaNaSaS [9]. It has over 70 different applications available and more than 500 users. Jobs can be submitted through an API provided by Platform LSF. The default remote entry point to the DAaaS system is via the OpenNebula cloud platform.

User Interface Layer: The platform provides a web portal for users to interact with the platform. It allows them to launch interactive analysis environments customised to specific science domains. Users currently have three methods to access interactive environment via remote desktop – NoVNC [2], SSVNC [3] and RDP [4]. NoVNC is a browser based client for VNC connections. It makes use of the HTML5 canvas element and WebSockets. It requires no setup or configuration for the user and can provide a full-screen remote desktop session in browser. RDP is a remote desktop technology, available on Windows systems by default and there are clients for Linux and OSX as well. It provides better performance than NoVNC for visually intensive work or connections across large geographic areas but it requires more steps from the user to set it up. The final method is using SSVNC. This is a secure VNC client that works across Windows, Linux and OSX. Again, it provides better performance than NoVNC but requires that the user installs a client.

Storage Layer: The platform provides three different method for users to access their data - through the browser, WebDAV [10] and Globus / GridFTP [11]. The browser based data access allows users to upload and download data into the analysis environments. WebDAV is supported out-of-the-box on Windows, Linux and OSX and allows users to map a network drive. For large scale data transfers, we provide a Globus Online endpoint. This technology is based on GridFTP that is able to handle the transfer of very large data volumes in a stable way.

Software: Software is made available via a mixture of CernVM-FS and a configuration management system, Quattor. CernVM-FS is a network file system based on HTTP that is designed to deliver scientific software onto virtual machines and physical worker nodes [5]. Some of its features include aggressive caching, digitally signed repositories and a user defined hierarchy for the software. It is capable of supplying multiple versions of software over different operating systems and architectures and has the ability for automatic file de-duplication. CernVM-FS is useful for scientific applications that have been designed in a portable way where dependencies are packaged into the software. When software is placed in a CernVM-FS repository it is then usable by both the VM analysis environments and the SCARF batch system. For software that is not portable, we use Quattor [6]. This is a system administration toolkit providing a portable and modular set of tools for the automated installation, configuration, and management of clusters, farms, grids and clouds. This is used to configure and install the software. Quattor is also used to configure the different types of analysis environments.

5. An example: tools for Macro-Crystallography.

As an initial example of the use of the DAaaS platform, SCD has been working with the Collaborative Computational Project on Macromolecular Crystallography (CCP4) to host tools developed by CCP4 in the prototype cloud platform. This is described in detail in a separate presentation at NoBugs 2016 [12]. A screen shot of the DAaaS remote desktop with the CCP4 tools is given in Figure 2.

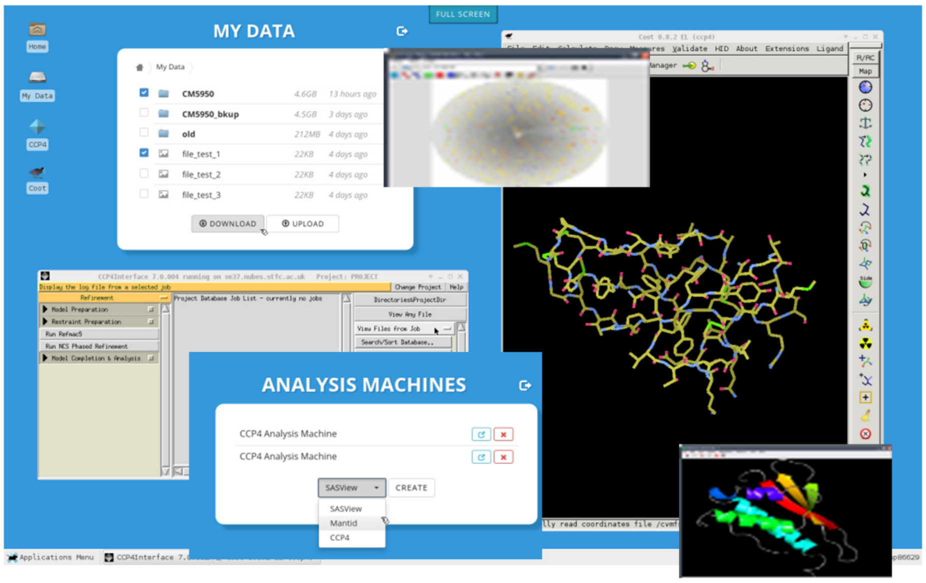


Figure 2 : the DAaaS remote desktop with CCP4 tools.

Registered CCP4-DaaS users can access CCP4 virtual machines via a remote desktop or a browser client. The CCP4 VMs are configured at the contextualisation stage which mounts the user's persistent file share and provides a set of standard software tools including the CCP4 integrated suite of programs

6. Future plans

The current prototype DAaaS platform provides a useful demonstrator and a testbed to validate and fine tune our initial requirements. Work continues to develop some of the more complex functionality of the system, as well as expand the range of software environments and user interfaces the system offers to different experimental user communities.

A particular challenge is the storage router. The router needs to co-ordinate the flow of the users' experimental data from the (disk and/or tape) archives to the cache of the compute service that the user is currently using. Then current data needs to be moved between compute platforms as the user changes their focus. To do this seamlessly and invisibly to the user requires a careful analysis of how data is used and what is required to keep the state of the individual data caches consistent with the local state of the data. Data then needs to be stored in persistent storage and then the caches flushed.

The use of the CernVM-FS to maintain consistent versions and images available to all platforms also can be integrated into a continuous integration software engineering process. The CernVM-FS can be supplied by continuous integration platforms such as Jenkins. In this case, as new software versions are released, they can be rapidly deployed onto the DAaaS platform.

The intelligence core of the system, embodied within the ICAT needs developing and expanding. In particular the integration of the ICAT with customisation services to provide tailored user environment needs to be developed. Further use of tracking and provenance to monitor and guide the usage of the system within particular analysis workflows needs to be explored.

Acknowledgements

We would like to thank our colleagues in SCD particularly in the ICAT, WLCG Tier 1 and CCP4 teams, and in ISIS and DLS for their help and advice in the development of the DAaaS. The work is supported by ISIS and SCD within STFC. We would also like to thank the wider PaNData community for providing an arena to share and discuss ideas in this emerging area.

References

[1] Stephen M Fisher et. al. The growth of the ICAT family. *NoBugs 2016, Copenhagen, Denmark, 17-19 October, 2016.*

[2] VNC client using HTML5 (WebSockets, Canvas) with encryption (wss://) support. https://kanaka.github.io/noVNC/

[3] SSL/SSH VNC viewer http://www.karlrunge.com/x11vnc/ssvnc.html

[4] An open source remote desktop protocol(rdp) server. http://www.xrdp.org/

[5] CernVM File System (CernVM-FS) https://cernvm.cern.ch/portal/filesystem

[6] Quattor http://www.quattor.org/

[7] OpenNebula - Flexible Enterprise Cloud Made Simple http://opennebula.org/

[8] IBM Spectrum LSF http://www-03.ibm.com/systems/spectrum-computing/products/lsf/

[9] Panasas - High Performance Parallel Storage http://www.panasas.com/

[10] Web-based Distributed Authoring and Versioning http://www.webdav.org/

[11] Research data management simplified | globus https://www.globus.org/#

[12] Shirley Crompton et. al. Virtual Large Facility for Experiment-based Structural Biologists. *NoBugs 2016, Copenhagen, Denmark, 17-19 October, 2016.*

DOI: 10.17199/NOBUGS2016.21

SIMEX: Simulation of Experiments at Advanced Light Sources

C Fortmann–Grote[1], A A Andreev[2,3], R Briggs[4], M Bussmann[5], A Buzmakov[6], M Garten[5,7], A Grund[5,7], A Huebl[5,7], S Hauff[1], A Joy[8], Z Jurek[9,10], N D Loh[11], T Rüter[1], L Samoylova[1], R Santra[9,10,12], E A Schneidmiller[13], A Sharma[3], M Wing[8,13], S Yakubov[13], C H Yoon[14], M V Yurkov[13], B Ziaja[9,10,15], A P Mancuso[1]

[1]European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany
[2]Max Born Institute, Berlin, Germany
[3]ELI ALPS, Szeged, Hungary
[4]European Synchrotron Radiation Facility ESRF, Grenoble, France
[5]Helmholtz–Zentrum Dresden–Rossendorf, Germany
[6]Institute of Crystallography, Russian Academy of Sciences, Moscow, Russia
[7]Technische Universität Dresden, Germany
[8]University College London, UK
[9]Center for Free Electron Laser Science, Deutsches Elektronen Synchrotron, Hamburg, Germany
[10]The Hamburg Center for Ultrafast Imaging, Hamburg, Germany
[11]Department of Physics, National University of Singapore, Singapore
[12]Department of Physics, University of Hamburg, Germany
[13]Deutsches Elektronen Synchrotron, Hamburg, Germany
[14]Linac Coherent Light Source, SLAC National Accelerator Laboratory, Menlo Park, USA
[15]Institute of Nuclear Physics, Polish Academy of Sciences, Krakow, Poland

E-mail: [1]`carsten.grote@xfel.eu`

Abstract. Realistic simulations of experiments at large scale photon facilities, such as optical laser laboratories, synchrotrons, and free electron lasers, are of vital importance for the successful preparation, execution, and analysis of these experiments investigating ever more complex physical systems, e.g. biomolecules, complex materials, and ultra–short lived states of highly excited matter. Traditional photon science modelling takes into account only isolated aspects of an experiment, such as the beam propagation, the photon–matter interaction, or the scattering process, making idealized assumptions about the remaining parts, e.g. the source spectrum, temporal structure and coherence properties of the photon beam, or the detector response. In SIMEX, we have implemented a platform for complete start–to–end simulations, following the radiation from the source, through the beam transport optics to the sample or target under investigation, its interaction with and scattering from the sample, and its registration in a photon detector, including a realistic model of the detector response to the radiation. Data analysis tools can be hooked up to the modelling pipeline easily. This allows researchers and facility operators to simulate their experiments and instruments in real life scenarios, identify promising and unattainable regions of the parameter space and ultimately make better use of valuable beamtime.

1. Introduction

Simulations of photon science experiments that cover the complete path of the photon from the source through beamline optics to the sample, its interaction with and scattering from the sample, and its detection have recently been used to model coherent diffraction imaging experiments of single biomolecules at Ångstrom resolution [1]. Single particle imaging (SPI) experiments (schematically depicted in Fig. 1a) are performed at x-ray free-electron lasers (XFELs) that meet the stringent requirements for ultra–short and intense x–ray pulses. Such start–to–end simulations allow us to investigate the feasibility and potential outcome of complex experiments under real–world conditions. Every part of the experiment is simulated such that imperfections in the sources spectral, temporal, and spatial structure, in the x–ray optical components, radiation damage to the sample, and detector limitations are all taken into account, allowing us to study how they affect the experimental observable and how these error sources correlate with each other.

The computer program `simex_platform` [2] was derived from the SPI simulation suite `simS2E` [1]. Besides SPI, `simex_platform` supports simulation of various types of experiments, involving various light sources (synchrotron sources, free-electron lasers (FELs), optical lasers), various beamline optics, samples, and diagnostic techniques. A python library provides standardized user interfaces to the simulation codes and their parameters. This approach was adopted from the Atomic Simulation Environment (ASE) [3, 4] which provides python objects (so called *Calculators*) that act as interfaces to ab–initio electronic structure codes. Users can embed their own codes into our simulation environment and run them under more realistic conditions compared to running them isolated with idealized parameters and initial conditions.

Some simulation codes (called *Backengines*) are readily shipped with `simex_platform`, in particular for SPI simulations. For other codes, we have developed standardized data formats (data interfaces) to integrate them into simulation workflows. Our software has modern and flexible deployment options including cmake, binary packages, and docker containers packed for various operating systems [5] and runs in parallel on high–performance computing clusters.

2. Concepts and structure of `simex_platform`

2.1. Simulation baseline

Currently, `simex_platform` supports simulation of photon experiments that follow a sequential pattern involving five steps:

(i) Photon source: generation of radiation e.g. by accelerated charged particles or optical resonators.

(ii) Photons propagate through a beamline consisting of apertures, slits, mirrors, gratings, and lenses.

(iii) The photons interact with the sample.

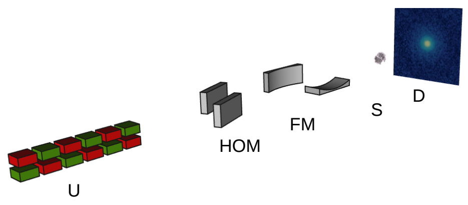
(iv) Scattered photons travel towards the detector.

(v) A detector registers incoming photons and produces a digital signal.

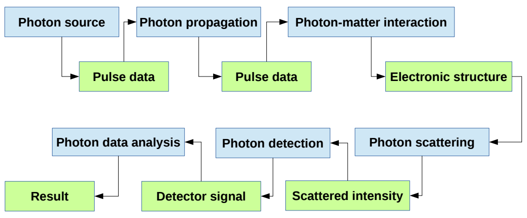
SPI, for which a schematic setup is shown in Fig. 1a is one example for a baseline application. The data flow between *Calculators* and intermediate data containers for which we have defined standardized data hierarchies and formats is shown in Fig. 1b.

2.2. Calculators

Each of the five simulation tasks in a baseline application is represented by a suitable *Calculator*. *Calculators* are organized in a lightweight abstraction scheme consisting of three abstraction layers as depicted in Fig. 2



(a) Schematic of a XFEL single–particle imaging experiment. Photons emitted from the undulator (U) of the XFEL are deflected by offset mirrors (HOM) and focussed by focussing mirrors (FM) onto the sample (S). Diffracted radiation is captured in the detector (D).



(b) Baseline workflow in `simex_platform`. Blue boxes represent *Calculators*, green boxes are data interfaces.

Figure 1.

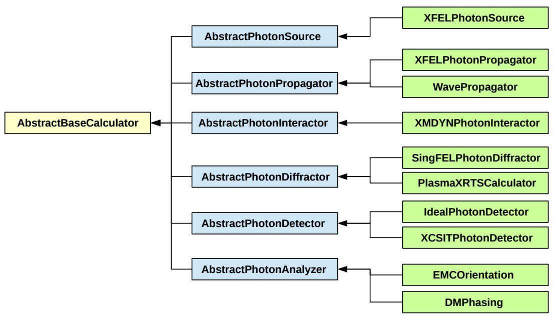


Figure 2. SIMEX *Calculators* and their relationships

The top level class *AbstractBaseCalculator* is a pure virtual object. It declares virtual members that a deriving *Calculator* must implement; in this way we ensure basic functionality of the *Calculator*, in particular that *Calculators* can exchange data and status information with each other. These virtual members are a data container, where the data manipulated by the *Calculator* is stored, together with corresponding query and assignment methods, furthermore methods for *Backengine* execution as well as input-output (IO) utilities. The intermediate level (blue boxes) provides abstract base classes for each block of the simulation workflow: AbstractPhotonSource, AbstractPhotonPropagator, AbstractPhotonInteractor, AbstractPhotonDiffractor, and AbstractPhotonDetector. A sixth abstract *Calculator*, AbstractPhotonAnalyzer is provided to ease integration of post–processing data analysis codes.

These classes are mainly responsible for declaring a list of variable names expected from the previous *Calculator* in the workflow and a list of variable names provided for the next *Calculator* in the workflow. Specialized *Calculators* adjust these lists according to their needs.

The third and final abstraction layer (green boxes) implements the interfaces to the simulation codes (*Backengines*) that perform the actual numerical work on the input data. They convert parameters and input data into suitable formats for the *Backengine*, before issuing the system or library calls, gather the results after the calculation has finished, and write them to an interface data file.

2.3. Interfaces

Start–to–end simulations in simex_platform require that any two subsequent *Calculators* employed in the simulation pipeline can communicate data amongst each other. The data source has to write the data in a format that the ensuing data sink can handle and interpret correctly. In simex_platform, we chose the Hierarchical Data Format (hdf5) [6] as the underlying format for all simulation data files. Data consistency in the workflow is realized through the aforementioned mechanisms of exchanging information about the required and provided data sets among the *Calculators*. Recently, we have started to adopt a more general approach of defining inter–calculator interfaces based on the open standard for particle–mesh data openPMD [7]. In particular, the transparent handling of physical units in openPMD makes it possible to implement generic unit conversion mechanisms.

3. Applications

As a first development milestone simex_platform supports simulation of three different baseline applications: single particle imaging at the European XFEL, coherent x–ray scattering from high power laser excited plasmas and x–ray probing of warm dense matter produced by high energy laser shock compression.

3.1. Single particle imaging

Femtosecond coherent x–ray pulses are generated in the SASE1 beamline of the European XFEL. The self-amplification of spontaneous emission (SASE) process is simulated with the code FAST [8], which gives the electric field distribution at the exit of the undulator section in a plane perpendicular to the beam direction as a function of time (discrete time slices). FAST is not public, we use precomputed datasets publicly available from a web database [9], instead.

The pulses are then propagated through the beamline and the SPB–SFX instrument's [10] focussing optics [11] to the sample interaction point. Propagation is modelled with the python library WPG [12, 13] with realistic parameters for the various optical components in the beamline. The output is again the electric field distribution as a function of time in the sample interaction plane transverse to the beam direction.

The ultra–short and intense x–ray pulses interact with the sample primarily via photo–ionization processes. Secondary Auger electrons quickly launch an ionization cascade creating a nano–plasma that starts to expand on timescales of 10–100 fs after the first photons hit the sample. These processes are calculated by means of Molecular Dynamics (MD) simulations with the code XMDYN [14], coupled to a Monte Carlo (MC) engine that describes the photon–matter interaction processes in a stochastic manner. Rates and cross sections are read from tabulated Slater–Hartree–Fock calculations using the code XATOM [14, 15]. We save the entire sample trajectory (atom positions and scattering form factors). We note here in passing that earlier simulations of SPI described in Ref. [16, 17] completely neglect radiation damage.

Scattering from the biomolecule is simulated in the far field approximation with the code SingFEL [18]. SingFEL utilizes the output from the photon–matter interaction code to calculate the instantaneously scattered light intensity by multiplying the intensity at the sample with the appropriately weighted form factors and geometry factors such as the detector solid angle. The instantaneous diffraction is integrated over the pulse duration and saved as a 2D array representing the pixel area detector.

The detector response to the incoming diffracted radiation can be calculated with the x-ray camera simulation toolkit (X-CSIT) software [19]. It describes particle generation, charge transport, and electronic signal processing. This can be used to simulated various detector effects like variations in pixel performance, non–linear gain, electronic noise, and counting statistics.

The detector simulation concludes the simulation pipeline. In SPI, diffraction patterns are submitted to a sequence of data post–processing algorithms to reconstruct the electron density from the measured diffraction data. `simex_platform` provides interfaces and *Calculators* for orientation and phasing algorithms described in [20].

3.2. Scattering from hot dense plasmas
Hot (few hundreds of eV) and dense (near solid density and beyond) plasmas are generated during the interaction of ultra–intense ($> 1 \times 10^{17}\,\mathrm{W/cm^2}$) ultra–short (of the order 10 fs) optical laser pulses with solid targets (e.g. metal foils). The plasma is characterized by strong density and temperature gradients and plasma instabilities. These features can be characterized by coherent x–ray scattering [21]. The photon source and propagation are described in the same way as for the SPI example above (Sec. 3.1). The short–pulse optical laser–plasma interaction is modelled with `PIConGPU` [22, 23], an open source, explicit, relativistic 3D Particle In Cell (PIC) code employing finite difference time domain techniques [24] to solve the Maxwell equations coupling the laser field and the plasma particles. For x–ray scattering calculations, we describe the radiation by a photon distribution, which we convert from the field distribution delivered by the propagation *Calculator*. The simulation tracks the photons through the plasma volume simulated by `PIConGPU` using the software `paraTAXIS`. The distribution of scattered photons across the detector plane can then be fed into the detector simulation as described in Sec. 3.1.

3.3. X–ray absorption and radiography in warm dense matter
High energy (few tens of J) pulses of optical laser light impinge on a solid target surface creating a shock wave travelling through the target. Pulse shaping and multiple shock compression can create density, pressure, and temperature conditions similar to planetary interiors [25], also referred to as warm dense matter (WDM). The target is probed by x–ray absorption spectroscopy and radiography to measure the thermodynamic conditions before, during, and after traversal of the shock wave.

In this case, the laser–matter interaction is modelled with radiation–hydrodynamic codes, solving the partial differential equations of radiation transport and hydrodynamic plasma motion. Two codes are currently interfaced, the 1D code “Esther” [26] and the 2D code “MULTI2D” [27]. The most important variables considered in these codes are those associated with the extreme conditions generated by the high–power lasers: pressure (density), temperature, and velocity. Feedback from these hydrocode outputs are crucial in the design and implementation of laser shock experiments with x–ray interactions.

By measuring the x–ray absorption as a function of position (radiography) and photon energy (x-ray absorption finestructure spectroscopy (XAFS)), the shock front structure, shock propagation dynamics, and ionic structure of the target can be monitored as a function of time by varying the delay between optical pump pulse and the x–ray probe.

The XAFS signal is a measure of the crystalline structure or liquid near order in the shock compressed target making it possible to identify structural phase transitions induced by the shock. XAFS can be simulated by combining first principle electronic structure methods with linear response theory. A well known implementation is the non–open source code FEFF [28].

Radiography can be used to image e.g. the shock front during dynamical compression. The Oasys [29] framework has the capability to calculate radiographs from simulated density profiles.

4. Summary and Outlook
We have presented in this work an open–source platform for simulation of experiments at advanced laser light sources. Our python library provides user interfaces and data format conventions that enable simulations of entire experiments from source to detector. *Calculator* interfaces to various simulation codes, both open and closed source, already exist, particularly for SPI simulations. For other applications, x–ray scattering from short–pulse laser excited

plasmas and warm dense matter diagnostics, we have collected the simulation codes and user interfaces will be provided in the next development steps.

Data format definitions are derived from the openPMD standard to facilitate communication between subsequent *Calculators*. These data interfaces will also enable us to perform simulations that go beyond the baseline scheme. Examples are simulation of coherent light sources based on laser–plasma accelerated electron beams and the generation of ion beams through laser–plasma interaction, their interaction with matter, and medical applications.

Acknowledgements
CFG, RB, AH, and SY acknowledge support from the European Cluster of Advanced Laser Light Sources (EUCALL) project which has received funding from the *European Unions Horizon 2020 research and innovation programme* under grant agreement No 654220. MW acknowledges support from the Alexander–von–Humboldt Foundation.

References

[1] Chun Hong Yoon et al., in: *Scientific reports* 6 (2016), p. 24791.
[2] URL: `https://github.com/eucall-software/simex_platform`.
[3] S. R. Bahn et al., in: *Comput. Sci. Eng.* 4.3 (2002), pp. 56–66.
[4] URL: `https://wiki.fysik.dtu.dk/ase/`.
[5] Sergey Yakubov et al., in: *NOBUGS*, 2016.
[6] The HDF Group, http://www.hdfgroup.org/HDF5/, 1997.
[7] Axel Huebl et al., 2015, URL: `http://www.openpmd.org`.
[8] E. L. Saldin et al., in: *Nucl. Instrum. Methods Phys. Res., Sect. A* 429.1 (1999), p. 233.
[9] URL: `https://in.xfel.eu/xpd/`.
[10] Adrian Mancuso et al., Technical Report, European XFEL GmbH, 2013.
[11] Richard J Bean et al., in: *Journal of Optics* 18.7 (2016), p. 074011.
[12] Liubov Samoylova et al., in: *J Appl Cryst* 49.4 (2016), pp. 1347–1355.
[13] URL: `https://github.com/samoylv/WPG`.
[14] Zoltan Jurek et al., in: *J Appl Cryst* 49.3 (2016), pp. 1048–1056.
[15] Sang-Kil Son et al., in: *Phys. Rev. A* 83.3 (2011), p. 33402.
[16] Kartik Ayyer et al., in: *Structural Dynamics* 2.4, 041702 (2015).
[17] S. Serkez et al., in: *ArXiv e-prints* (2014), p. 1407.8450.
[18] URL: `www.github.com/eucall-software/singfel`.
[19] A. Joy et al., in: *Journal of Instrumentation* 10.04 (2015), pp. C04022–C04022.
[20] Ne-Te Duane Loh et al., in: *Physical Review E* 80.2 (2009), p. 026705.
[21] T. Kluge et al., in: *Physics of Plasmas* 23.3 (2016), p. 033103.
[22] M Bussmann et al., in: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, New York, USA, 2013, p. 1.
[23] URL: `https://github.com/ComputationalRadiationPhysics/picongpu`.
[24] Kane Yee, in: *IEEE Transactions on Antennas and Propagation* 14.3 (1966), pp. 302–307.
[25] Y Ping et al., in: *Phys. Rev. Lett.* 96.25 (2006), p. 255003.
[26] J. P. Colombier et al., in: *Phys. Rev. B* 71.16 (2005), p. 165406.
[27] R. Ramis et al., in: *Computer Physics Communications* 180.6 (2009), pp. 977–994.
[28] John J. Rehr et al., in: *Phys. Chem. Chem. Phys.* 12 (2010), pp. 5503–5513.
[29] M. Sanchez del Rio et al., in: *Proc. SPIE 9209* (2014), p. 9209X.

DOI: 10.17199/NOBUGS2016.4

The unified software package Sonix+. Analysis of development experience

A.S.Kirilov

Frank Laboratory of Neutron Physics, Joint Institute for Nuclear Research, 141980 Dubna, Moscow Region, Russia.

E-mail: akirilov@nf.jinr.ru

Abstract. The software package Sonix+ (Sonix) [1] developed as a unified control software for neutron instruments. It has been in operation since 1995. Currently it is used at instruments of the IBR-2 (FLNP, JINR), as well at some instruments at other centers of the Russian Federation (totally about 20 installations). Though the main ideas of the Sonix + are largely similar to the decisions taken at other centers of the NOBUGS community, the consideration of specific needs and traditions prevailing at the FLNP, had substantial influence at the structure of the complex, and some implementation decisions. This applies, for example, the choice for the operating system of instrument control computer, implementation details of so-called "database" - parameter storage with fast access for inter module communication, a graphical user interface, choice of the spectra recording format, etc. During the long period of exploitation the complex was developed, it has been added new components and has been changed to the new requirements. This led, in particular, to the adjustment of a number of initial decisions, and to a certain eclecticism. Some implemented features have unclaimed, and others, on the contrary, have required further development.

1. Few words about history

The prototype of the Sonix was maintained at the Neutron Spectrometer of High Resolution NSHR) at beam 7a of the pulsed reactor IBR-2 of the FLNP JINR in March 1995. A VME based computer running Os-9 was used for the instrument control. The GUI was based on the X11. The Varman database [2] has been used for the inter client communication. Further this software has been adapted for several other instruments and was named the Sonix (SOftware for Neutron Instruments on the X11 base). The adaptation was not very easy because the initial software did not intended as universal one. For example, there was no possibility to organize hierarchical structures of execution modules, the custom made script was rather poor, there was no visualization of spectra for data from PSD, etc.

In the early 2000s it was decided to replace the expensive control VME computers with cheaper PCs, and the Unix-like Os9 operating system with the Microsoft Windows. This stimulate to re-create our control software due to obtained exploitation experience and recent trends. The new version is called Sonix+. Currently the Sonix + is the set of modules, united by the common philosophy, which allows to organize control system for arbitrary instrument rather simply.

2. Main features of the Sonix+

The Sonix+ is a unified, universal modular set of modules. It has many features both in structural principles and implementation ideas more or less similar to most of modern control software [3]. There are some important differences.

Sonix + is mostly local system. Computation power of modern computers usually is sufficient to do the whole job, so this decision simplifies the system maintenance. For those applications that require synchronization with external computers so called "Cchannel" (command channel) module is provided. In our practice this need is very seldom. For remote user access several possibilities are available - the Windows Remote Desktop, the VNC, the WebSonix service [4], etc.

The Microsoft Windows is used as an operating system on control computers. This choice was made on the urgent wishes of our users, many of whom are perceived the Unix-like Os9 operating system very negatively. Another reason for this choice was the advantage in the software development systems, compared with the Linux at that time.

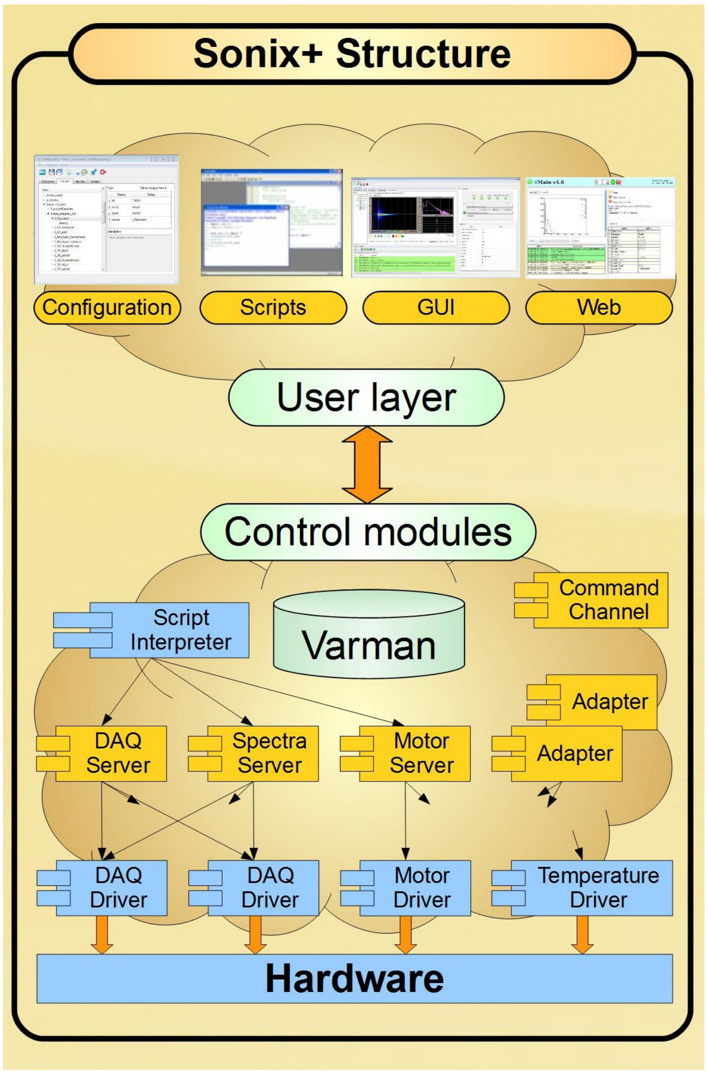


Figure 1. The Sonix+ structure

Unfortunately, there is separate data format for each IBR-2 instrument up to now. So, the Nexus standard [5] for storing measurement results were not chosen because our users do not interested in it.

In this situation we decided to our own format as common internal format for all instruments. It comprises two files, one of which contains the actual spectra and the second - the Varman database snapshot. This image contains all measurement parameters registered in the Sonix+. Finally data are transferred to format specific to each spectrometer.

We have proposed and implemented a universal approach to GUI [6]. This means that the same set of clients are sufficient to satisfy requirements of an arbitrary instrument without change.

3. Sonix+ structure

The structure is presented in the figure 1. In a modular system every component (module) is responsible for some device or function. In practice, a set of modules for each instrument has a non-linear structure. Some modules serve hardware devices – they are at a lower level of the hierarchy. The other modules are used to maintain work of the others. We have chosen these components and tried to create them as universally as we can expect. We called these modules servers. If one has to port Sonix+ to another instrument, servers can be used without redesign.

Of course, for communication of servers with other modules specialized protocols are needed. Actually the following protocols have been established:

- universal protocol for device control and inquiring;
- DAQ protocol;
- motor protocol;
- remote inquiring and control protocol via sockets.

All of these were proved by the long practice with the exception of motor server. It will be eliminated in the nearest future.

4. Variable manager (Varman)

During the experiment, all modules communicate through the special storage. It is called Varman database. This storage provides very fast access to parameters. At every moment the contents of the Varman database completely represents the current state of the measurement process at the instrument with the exceptions of spectra.

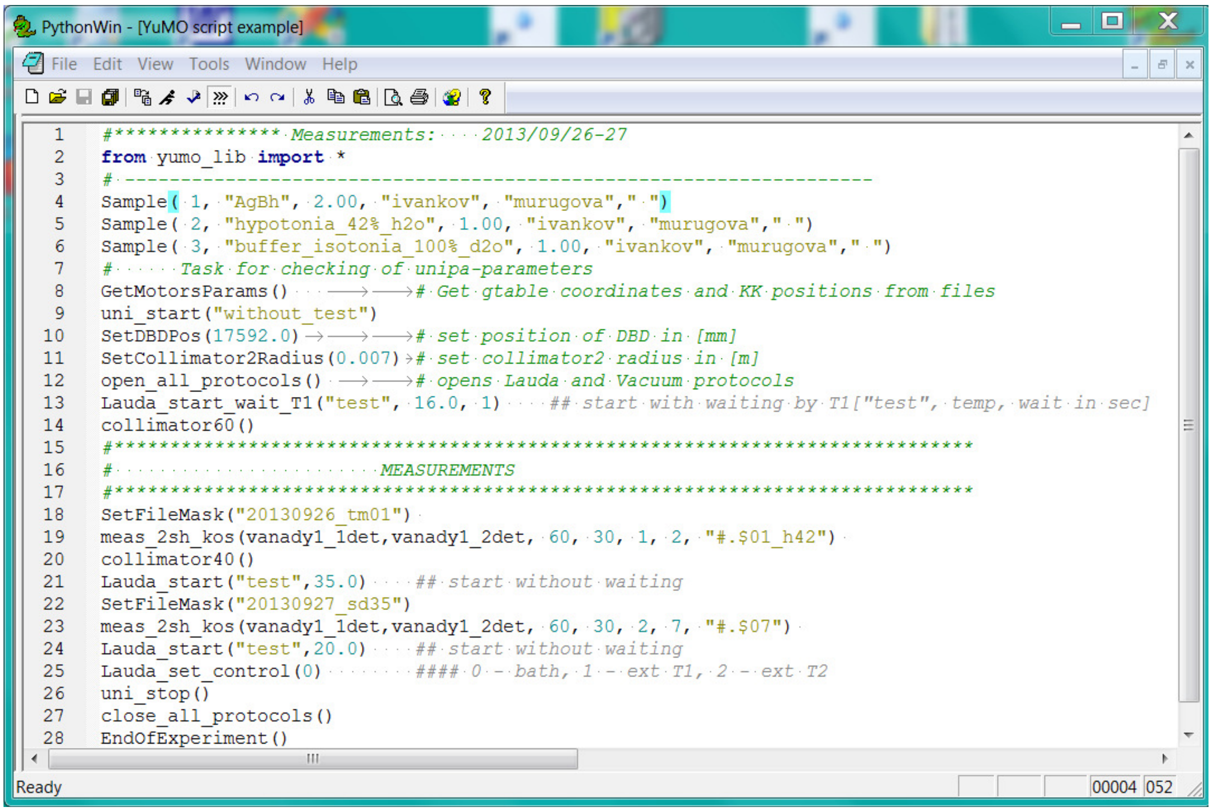
The idea and initial soft were taken from IRI TU Delft [2]. Redesign for the Windows was made by V.Yudin [7]. Initially all communications were made through simple variables like "motor position", etc. Now operations mostly deal with "structures" of variables like "device status" which consist of significantly complex structures. So some initially implemented Varman features like "interests" - reaction on appropriate events - are now out of use. Another feature which seemed very important for Varman authors - differentiation access to variables for various user categories - did not even implemented in PC version. It lost practical significance at this moment. Unfortunately, another problem - protection of the control computer against external attacks - is quite more urgent.

5. Script utilization

The Python language is used as a script language in the Sonix+ and all nice features and packages of Python are available to control experiments and for preliminary data processing.

This allows one to reasonably separate the specificity of instrument measurements into the most flexible and easy-to-change component. This fact is very useful from the viewpoint of unification and for facilitating the transfer of Sonix+ to new instruments.

It is also important that using the Python as a script language opens a wide door for the user to program experiments. Unfortunately, during these years there were only two user's attempts to take this opportunity.


PythonWin - [YuMO script example]
File Edit View Tools Window Help
Ready
00004 052

```
#*************** Measurements:    2013/09/26-27
from yumo_lib import *
# ------------------------------------------------------------------
Sample( 1, "AgBh", 2.00, "ivankov", "murugova"," ")
Sample( 2, "hypotonia_42%_h2o", 1.00, "ivankov", "murugova"," ")
Sample( 3, "buffer_isotonia_100%_d2o", 1.00, "ivankov", "murugova"," ")
#      Task for checking of unipa-parameters
GetMotorsParams()               # Get gtable coordinates and KK positions from files
uni_start("without_test")
SetDBDPos(17592.0)              # set position of DBD in [mm]
SetCollimator2Radius(0.007)     # set collimator2 radius in [m]
open_all_protocols()            # opens Lauda and Vacuum protocols
Lauda_start_wait_T1("test", 16.0, 1)    ## start with waiting by T1["test", temp, wait in sec]
collimator60()
#**********************************************************************
#                        MEASUREMENTS
#**********************************************************************
SetFileMask("20130926_tm01")
meas_2sh_kos(vanady1_1det,vanady1_2det, 60, 30, 1, 2, "#.$01_h42")
collimator40()
Lauda_start("test",35.0)    ## start without waiting
SetFileMask("20130927_sd35")
meas_2sh_kos(vanady1_1det,vanady1_2det, 60, 30, 2, 7, "#.$07")
Lauda_start("test",20.0)    ## start without waiting
Lauda_set_control(0)        #### 0 - bath, 1 - ext T1, 2 - ext T2
uni_stop()
close_all_protocols()
EndOfExperiment()
```

Figure 2: The YuMO instrument script example

The Sonix+ script is the pure Python code. To control the script interpretation a special module is developed. As in other cases, some possibilities inherent in it were not in demand in practice by users. With instruments evolution and to satisfy new users requests actual script is becoming more and more difficult. To simplify user's life each spectrometer is equipped with a library of standard operations. It provides the ability to validate the script before it execution also.

6. User interfaces

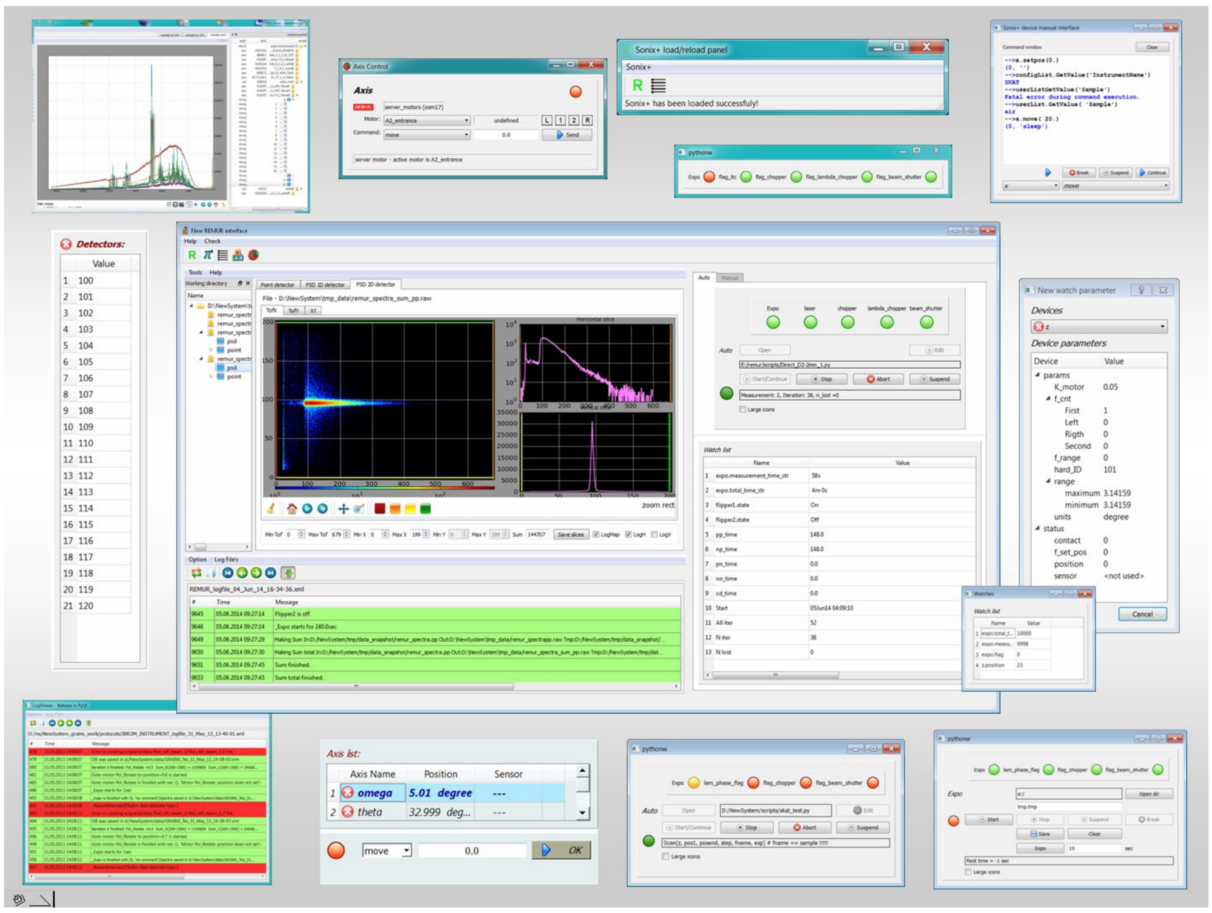
There are generally three kinds of Sonix+ interfaces: command line, GUI and web interface.
Command line propose to enter commands as a Python string using any of standard shell (mostly PyhonWin client).

Sonix+ GUI also has undergone a long evolution from separate window for the each controllers to a single common window. There are universal GUI and some specialized. The universal GUI is available at all instruments. It is organized according to principle – *each sub window is dedicated to one of the main user's needs*. There are three main needs to watch: current state of the instrument, the measurement history (log file), the picture of spectra (spectrum). The fourth need is to control the measurement process. Thus, four programs (windows, widgets) are generally sufficient to conduct an experiment. There are additional programs as well (the Load control panel, the Configuration editor and some else).

Specialized GUI are created by special request, for instance, for instrument tuning. Some specialized interfaces were created for instruments at other centers, if their users are known to be so far from modern computers.

7. GUI components set

The reasons for the development of a specialized set of widgets to control the experiment, as well as the rationale for its specific implementation considered in the work [8]. The set of visual components (widgets), which facilitates the development of graphical user interface (GUI) to control the measurements on neutron instruments were implemented in PyQt. The set components correspond to the basic functions of the user interface for managing the measurement and visualization of spectra, as well as provide program upload / download for the package components and a number of other functions. Most of these widgets could be used as independent client or as a part of more complex window.



	Value
1	100
2	101
3	102
4	103
5	104
6	105
7	106
8	107
9	108
10	109
11	110
12	111
13	112
14	113
15	114
16	115
17	116
18	117
19	118
20	119
21	120

	Axis Name	Position	Sensor
1	omega	5.01 degree	---
2	theta	32.999 deg...	---

Figure 3: The universal GUI and the widget set.

8. Spectra visualization

Spectra visualization includes a set of widgets for visualization of mono detectors, 1D PSD and 2D PSD data. Spectra can be read from files (including zipped files) or from the DAQ controllers directly. The matplotlib library panel is used to display the graph. Besides curve drawing the panel implements typical operations like scaling, shifting, etc. Additional operations implemented in the widget are:

- linear/log scaling;
- automatic/manual definition of limits in the display window;
- some others concerning dimensionality of the data.

9. Instrument tuning

The tuning program (ICE) is developed for reflectometers of the IBR-2. The ICE program is implemented as an add-on the control complex Sonix + and is designed to adjust the instrument before the main measurement. Program have been written using the PyQt and the graphics library matplotlib.

10. Future plans

The main direction of our future efforts is organization of centralized repository for measurement data with corresponding services.

11. Conclusion

Overall the Sonics/Sonix+ project success is proved by its practice. At the same time, as would be expected, some of the previous decisions have to be revised, in accordance with emerging needs and opportunities.

12. Acknowledgements

For a long period of development and use of Sonix many people in the FLNP and abroad contributed to the project with their work, ideas and criticism. The author is deeply grateful to all of them.

References

[1] http://sonix.jinr.ru/wiki/doku.php?id=en:index
[2] Sckipper M.N. 1997 *Proc. of DANEF'97* E10-97-272 (Dubna) pp 288-294
[3] http://www.nobugsconference.org/Topics:ESS
[4] I.A.Morkovnikov and A.S.Kirilov 2013 Upgrading WebSonix - Remote Instrument Control System Experiment on the IBR-2 Reactor *Proc. of NEC'2013* (Dubna) pp 181-185
[5] http://www.nexusformat.org
[6] A. S. Kirilov, S. Veleshki, S. M. Murashkevich and T. B. Petukhova The unified GUI for neutron instrument control based on PyQt 2013 *Proc. of NEC'2013* (Dubna) pp 158–160
[7] A. S. Kirilov and V. E. Yudin 2003 The Implementation of realtime database for control of the experiment in the MS Windows environment *Prep. of JINR* (Dubna) R13-2003-11
[8] Kirilov A. S. and Petukhova T. B. The Set of Components to Create GUI for Neutron Instrument Control Systems on the Base of PyQt *Comm. of JINR* (Dubna) P10-2015-38

DOI: 10.17199/NOBUGS2016.36

DonkiOrchestra: a scalable system for data collection and experiment management based on ZeroMQ distributed messaging

Borghes Roberto, Kourousias Georgios
Elettra Sincrotrone Trieste, ITALY

E-mail: roberto.borghes@elettra.eu

Abstract. Synchrotron and Free Electron Laser beamlines consist of a complex network of devices. Such devices can be sensors, detectors, motors, but also computational resources. The setup is not static and is often upgraded. The data acquisition systems are constantly challenged by such continues changes and upgrades, so a constant evolution of software technologies is necessary. DonkiOrchestra is a TANGO based framework developed at Elettra Sincrotrone Trieste that takes full advantage of the ZeroMQ distributed messaging system and supports both data acquisition and experiment control. In the DonkiOrchestra approach, a TANGO device referred to as Director, provides the logical organization of the experiment as a sequential workflow relying on triggers. Each software trigger activates a set of Players that can be hierarchically organized according to different priority levels. This allows for concurrency and map-reduce strategies. Data acquired by the Players is tagged with the trigger number and sent back to the Director which stores it in suitably structured HDF5 archives. The intrinsic asynchronicity of ZeroMQ maximizes the opportunity of performing parallel operations and sensor readouts. This paper describes the software architecture behind DonkiOrchestra, which is fully configurable and scalable, so it can be reused on multiple endstations and facilities. Furthermore, experimental applications, performance results and future developments are presented and discussed.

1. Introduction

The research centre Elettra hosts two advanced particle accelerators: the electron storage ring Elettra and the free-electron laser (FEL) FERMI hosting 34 experimental stations.

Elettra is a third-generation synchrotron light source that provides state-of-the-art techniques to lead experiments in physics, chemistry, biology, life sciences, environmental science, medicine and cultural heritage. Currently 28 beamlines utilize the radiation generated by the Elettra source. It can operate at two different electron energies: 2.0 GeV for enhanced extended ultraviolet performance and spectroscopic applications; 2.4 GeV for enhanced x-ray emission and diffraction applications. A substantial facility upgrade, named Elettra 2.0, is currently planned [1].

FERMI is a seeded free electron laser (FEL) facility operating in the ultraviolet and soft x-ray range. FERMI has been developed to provide fully coherent ultrashort 10-100 femtosecond pulses with a peak brightness ten billion times higher than that made available by third-generation light sources. It is a single-pass FEL that comprises two separate coherent radiation sources: FEL-1 operates in the wavelength range between 100 and 20 nm via a single cascade harmonic generation, while the FEL-2 is designed to operate at shorter wavelengths (20-4 nm) via a double cascade configuration.

The Scientific Computing team manages a set of core services spanning from beamline control and data acquisition systems to Cloud computing and storage resources, plus a set of web based services for e-Science and Scientific Business management. The data acquisition and experiment management system of all the FERMI endstations is based on FermiDAQ [2], a common software framework highly configurable and scalable. The use of a single software infrastructure showed a remarkable enhancement of the team efficiency, but it is deeply dependent on the FERMI pulsed source architecture. So we focused on a new version of the software, reusable on a wider spectrum of scientific endstations and facilities.

The next sections provide the reader with a technical overview of the new framework, called DonkiOrchestra, its experimental applications and future developments.

2. DonkiOrchestra architecture overview

Synchrotron and Free Electron Laser beamlines consist of a complex distributed network of devices, like sensors, detectors, motors, but also computational resources. Each scientific setup is unique, not static and is periodically upgraded with new instrumentation. At the base of the design of the new framework DonkiOrchestra there are few fundamental goals:

- to maximize scalability and customizability;
- to enhance synchronized parallel operations;
- to minimize data communication overheads.

DonkiOrchestra draws on experience gained with the FEL data acquisition system FermiDAQ. Any experiment at FERMI is based on the acquisition of a train of light pulses with a frequency of 50Hz. Experimental instrumentation is synchronized with the arrival of the light pulses through an electric trigger signal which is tagged with an incremental index. Such a system forces the parallelism of the the control system and acquired data may be easily correlated through the photon pulse index.

This approach of organizing a scientific experiment as a train of independent phases has been inherited by DonkiOrchestra, with the main difference that the synchronization trigger is not a hardware signal but a software event shared through the instrumentation network.

As in a symphonic orchestra the system is composed by a Director and multiple independent Players. Each Player belongs to a Priority group and has a specific task to execute. The Director conducts the experimental sequence by sending a train of software triggers to the Players. For each step of the experimental sequence, a trigger signal is sent to the Players with the highest Priority 0, then to the group of Players with Priority 1 and so on. Each Player executes its task upon the arrival of the trigger and sends back to the Director an acknowledge event. A Player associated to a scientific sensor acquires the data, puts a tag on it with the trigger number and sends it back to the Director for storing it in suitably structured archives. In Figure 1 is schematically shown how a standard 2D scan experiment may be implemented with the described approach.

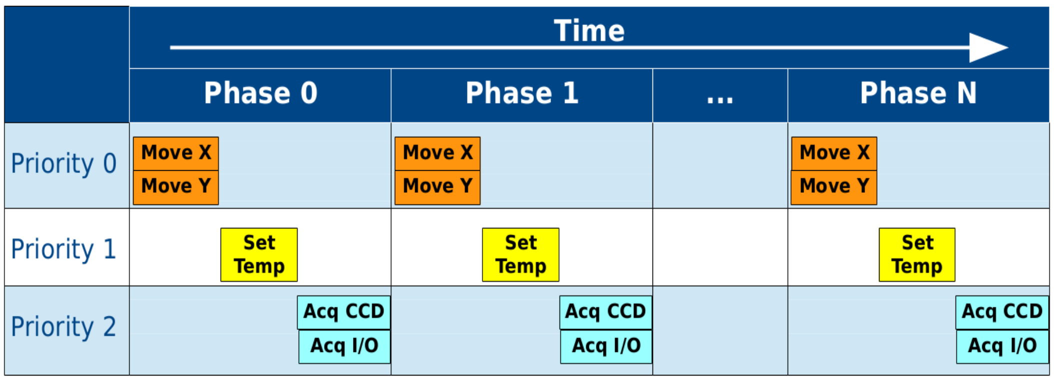


Figure 1. A 2D scan experiment organized as a triggered sequence.

3. DonkiOrchestra technical design

DonkiOrchestra has been written using TANGO [3], a robust and easy to use distributed control system toolkit. TANGO is operating system independent and supports C++, Java and Python for all its components.

DonkiOrchestra Director and Players are independent TANGO devices distributed on different computers connected through an ethernet network. As core language of the new framework we decided to use Python that adds dynamicity and portability. This choice affects only the Director device, since the Player devices may be developed using also the C++ or the Java TANGO binding.

ZeroMQ [4] messaging system has been chosen for its asynchronous I/O model that fits the need of having a scalable distributed application. It has a score of language APIs and runs on most operating systems. Besides this, the TANGO framework recently adopted ZeroMQ for its event system.

One of the project starting points was the use of ZeroMQ messages for scientific data sharing, so we performed a few preliminary reliability tests of the protocol and a set of efficiency tests to evaluate its performance at the increase of the data chunk size. Our particular interest was the measurement of the time spent by a Player for publishing a big chunck of data. Figure 2 shows the results of a series of tests that put the time spent for a ZeroMQ push() operation in relation with the data message size. The comparison of the different curves clearly shows that a TANGO User-Event has the same performance of the basic ZeroMQ API function. The test has been performed using Tango 8.1.2.c, ZMQ 3.2.3 on two x64 Linux machines connected through a 10Gb ethernet connection.

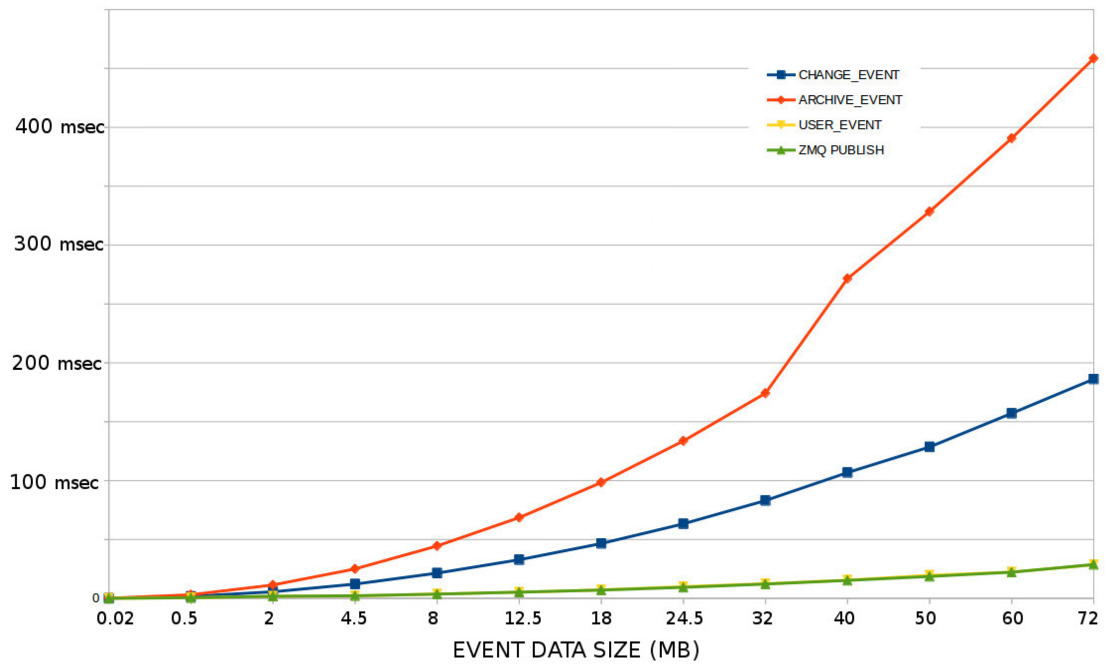
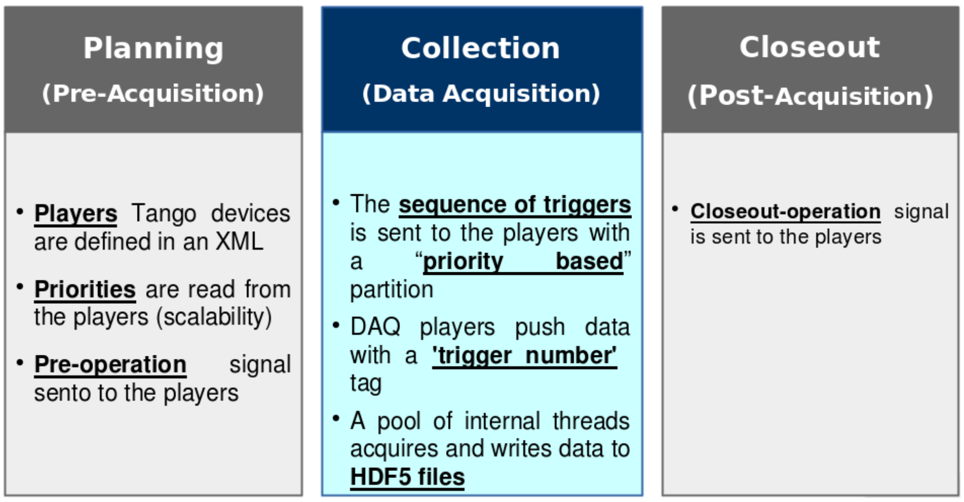


Figure 2. Time spent for a ZeroMQ publish() function vs message data size.

The Director device has been designed with a configurable, scalable and reliable structure. Schematically, an experiment sequence is composed by three consecutive phases shown in Figure 3.

- In the planning phase the Director reads the address of the Player devices from an XML file, retrieves their priorities and sends them a preparation signal. During this preliminary phase each Player sets up the sequence of operations to be performed upon the arrival of the event triggers, e.g. a motion player may prepare the sequence of target positions for a scan.

- During the collection phase the Director sets up and conducts the train of ZeroMQ trigger events. Each trigger message is composed by an index number and a priority tag. For each main experimental phase, the Director sends a trigger to the group of Players with the highest priority, waits for all the acknowledge messages, then proceed with the next priority group.
- In the closeout phase the Director sends to the Players a specific signal in order to perform any needed closing procedure.



Planning
(Pre-Acquisition)

- **Players** Tango devices are defined in an XML
- **Priorities** are read from the players (scalability)
- **Pre-operation** signal sento to the players

Collection
(Data Acquisition)

- The **sequence of triggers** is sent to the players with a "**priority based**" partition
- DAQ players push data with a **'trigger number'** tag
- A pool of internal threads acquires and writes data to **HDF5 files**

Closeout
(Post-Acquisition)

- **Closeout-operation** signal is sent to the players

Figure 3. DonkiDirector experiment phases.

Concurrently to the experiment execution, a pool of internal threads acquires and saves asynchronous data messages coming from the acquisition Players. The structure of a data message is composed by the experimental data and a trigger index information, so data messages coming from different sources may be easily correlated. The Director device stores scientific data in HDF5 [5] binary archives. Experimental data may be integrated with environment information by defining some metadata sources in the Director XML configuration file. A simplified schema of the explained architecture is shown in Figure 4.

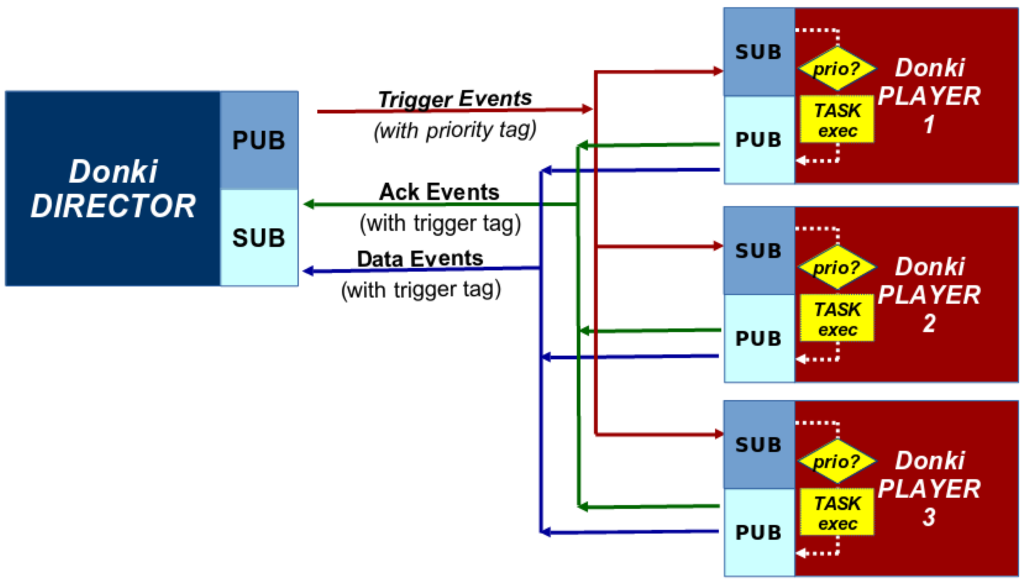


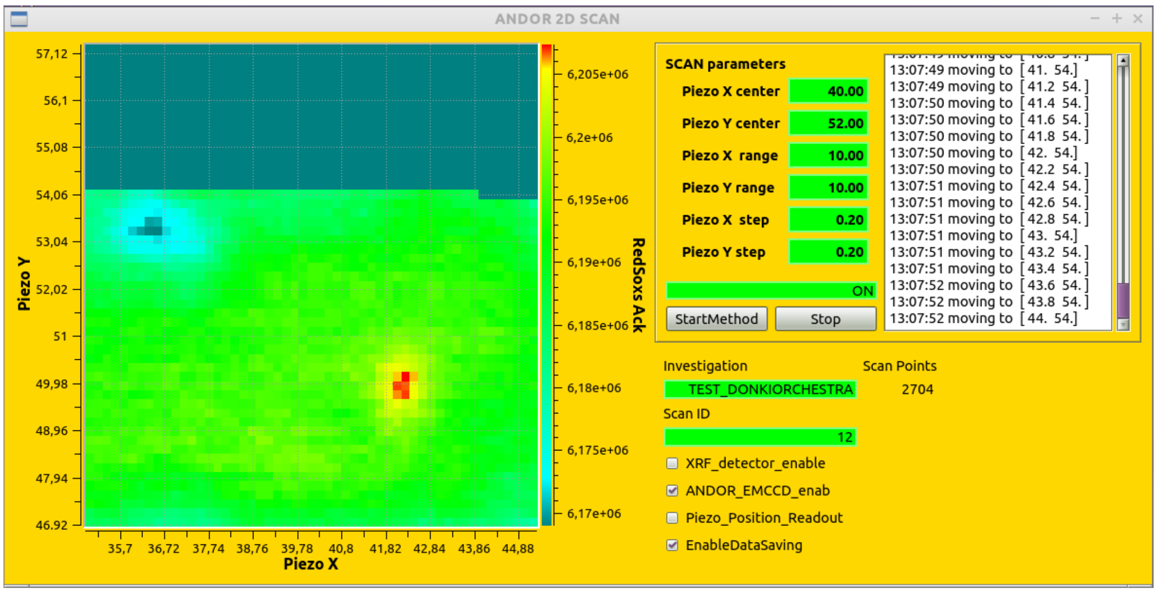
Figure 4. DonkiOrchestra schematic architecture.

4. DonkiOrchestra at Twinmic beamline

European scientists with highest expertise in X-ray microscopy, diffractive X-ray optics, X-ray contrast technologies and detection, started in 2001 to integrate the advantages of complementary scanning and full-field imaging modes into a *single* instrument, which they named 'TwinMic'. Since its opening to the users in late 2007, the TwinMic beamline at the Elettra synchrotron has been attracting the interest of a broad life science community. Using simultaneously the x-ray transmission and emission detection systems it is possible to perform elemental, absorption and phase contrast imaging that provides complementary chemical and morphological information for the sample under investigation.

The extreme versatility of the Twinmic beamline allows to plan a wide range of different scientific experiments. TwinMic beamline is a prefect field test for DonkiOrchestra, so we installed and adapted it in order to control the TwinMic instrumentation:

- a Physics Instruments piezoelectric sample stage;
- a Princeton X-ray CCD 1300x1340;
- an Andor IXON DV860 camera;
- a XGLab low-energy X-ray fluorescence detector.



SCAN parameters

Piezo X center	40.00
Piezo Y center	52.00
Piezo X range	10.00
Piezo Y range	10.00
Piezo X step	0.20
Piezo Y step	0.20

ON

```
13:07:49 moving to [41.  54.]
13:07:49 moving to [41.2  54. ]
13:07:50 moving to [41.4  54. ]
13:07:50 moving to [41.6  54. ]
13:07:50 moving to [41.8  54. ]
13:07:50 moving to [42.  54.]
13:07:50 moving to [42.2  54. ]
13:07:51 moving to [42.4  54. ]
13:07:51 moving to [42.6  54. ]
13:07:51 moving to [42.8  54. ]
13:07:51 moving to [43.  54.]
13:07:51 moving to [43.2  54. ]
13:07:51 moving to [43.4  54. ]
13:07:52 moving to [43.6  54. ]
13:07:52 moving to [43.8  54. ]
13:07:52 moving to [44.  54.]
```

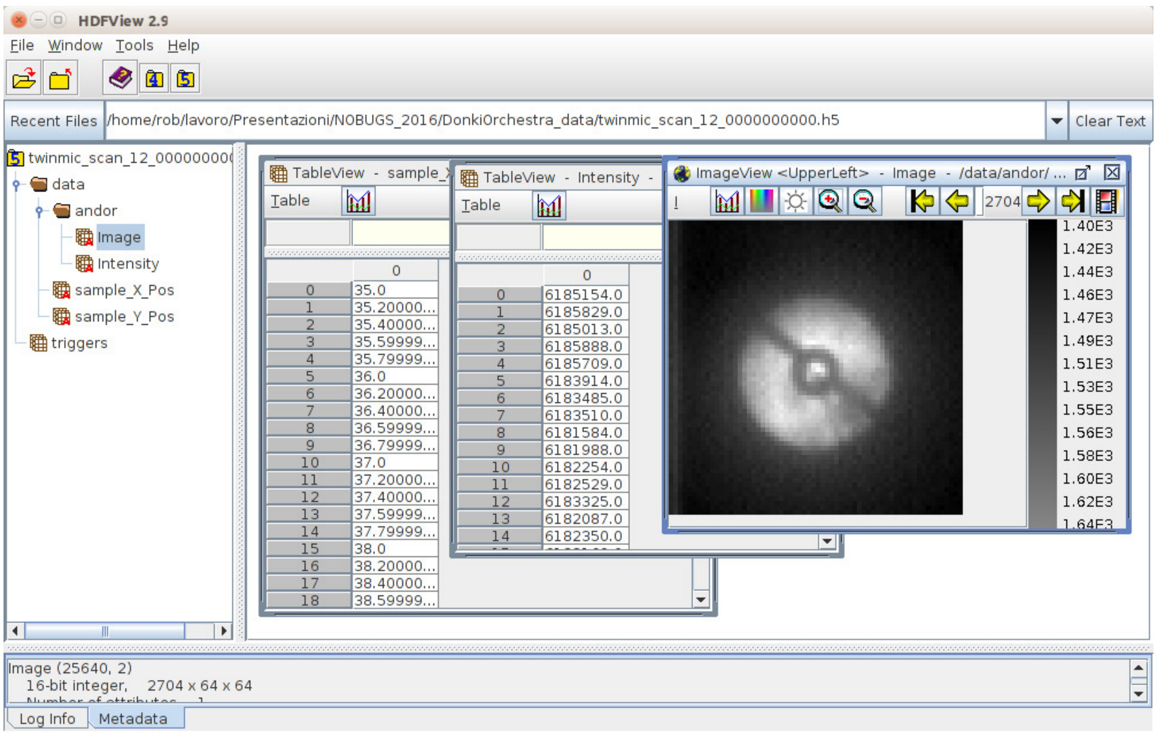
Investigation: TEST_DONKIORCHESTRA
Scan Points: 2704
Scan ID: 12

- ☐ XRF_detector_enable
- ☑ ANDOR_EMCCD_enab
- ☐ Piezo_Position_Readout
- ☑ EnableDataSaving

Figure 5. Two dimensional scan performed with DonkiOrchestra at Twinmic beamline.

As first experiment, DonkiOrchestra has been used to implement a two dimensional raster scan with concurrent acquisition of CCD images and XRF spectrums. For such application we have developed a dedicated graphical user interface (Figure 5) that uses the event-based infrastructure of DonkiOrchestra. Similarly to the Director, the GUI receives the ZeroMQ data messages from the Players. Each message contains experimental data (stage positions and image intensity) and a trigger index that is used to correlate the data and create the scan plot .

The results of the scan are stored in a HDF5 archive (Figure 6) that contains the datasets of stage positions, CCD images, XRF spectrums and some environment metadata. Data belonging to different datasets may be easily correlated by a post-processing analysis software in order to produce scientific results.



/home/rob/lavoro/Presentazioni/NOBUGS_2016/DonkiOrchestra_data/twinmic_scan_12_0000000000.h5

	0
0	35.0
1	35.20000...
2	35.40000...
3	35.59999...
4	35.79999...
5	36.0
6	36.20000...
7	36.40000...
8	36.59999...
9	36.79999...
10	37.0
11	37.20000...
12	37.40000...
13	37.59999...
14	37.79999...
15	38.0
16	38.20000...
17	38.40000...
18	38.59999...

	0
0	6185154.0
1	6185829.0
2	6185013.0
3	6185888.0
4	6185709.0
5	6183914.0
6	6183485.0
7	6183510.0
8	6181584.0
9	6181988.0
10	6182254.0
11	6182529.0
12	6183325.0
13	6182087.0
14	6182350.0

```
Image (25640, 2)
   16-bit integer,    2704 x 64 x 64
```

Figure 6. TwinMic HDF5 archive produced with DonkiOrchestra.

5. Conclusions

The present paper describes the software architecture behind DonkiOrchestra, a framework which supports both data acquisition and experiment control. The DonkiOrchestra approach is fully configurable and scalable, so it can be reused on different endstations and facilities. The strength of this software product resides in its design choices: a powerful messaging system like ZeroMQ that maximizes the opportunity of performing parallel operations and sensor readouts, a reliable distributed control system like TANGO, a dynamic and portable language like Python and a fully configurable structure that permits a high degree of customization.

Acknowledgement

The authors, members of the Scientific Computing team of Elettra, thank the IT Group and the Twinmic beamline staff of Elettra Sincrotrone Trieste. Moreover they acknowledge the importance of advanced technologies and open source software like that of the distributed control system TANGO and ZeroMQ messaging system.

[1] E Karantzoulis E, “Elettra 2.0 – The next machine”,Proceedings of IPAC2015, Richmond, VA, USA
[2] R Borghes, V Chenda, A Curri, G Kourousias, M Lonza, M Prica, and R Pugliese. “A Common Software Framework for FEL Data Acquisition and Experiment Management at FERMI.” Proceedings of ICALEPCS2013, San Francisco, CA, USA, 2013, 6–11
[3] www.tango-controls.org/
[4] http://zeromq.org/
[5] https://support.hdfgroup.org/HDF5/

DOI: 10.17199/NOBUGS2016.46

On-Axis-View: a GUI library to enhance the sample environment control

Jordi S Andreu, Fulvio Becheri, Guifré Cuní, Roberto J Homs, Gabriel Jover-Manas and Daniel Roldan

ALBA Synchrotron Light Source, Carrer de la llum 2-26, 08290 Cerdanyola del Vallès, Barcelona, Spain

E-mail: `jandreu@cells.es, ctbeamlines@cells.es`

Abstract. Many X-Ray experiments performed in a synchrotron facility (like Macromolecular Crystallography, Non-Crystalline Diffraction or Powder Diffraction experiments) require precise control of the sample position and orientation (centering procedures), the proper positioning of the X-Ray beam and its morphological analysis and also, a friendly integration of all the different optical and motorized instruments into the main control interface. The *On-Axis-View* library provides such functionalities, enhancing the development of customized Graphical User Interfaces according to the specific user requirements and also a better user experience. In addition, the usage of the library greatly reduces the required time to develop new solutions and eases its maintenance.

1. Introduction

A mandatory feature of any synchrotron beamline is the remote monitoring and operation of all its devices: motors, cameras, valves, detectors, etc. From the control system side, there are two important issues to handle prior to operate. The first issue is the fine-tuning, alignment, monitoring and characterization of the target sample at the final experimental position. The second is the correct characterization and alignment of the incident beam at the position of the sample. In order to visualize the scene, an area scan camera is usually mounted with its optical axis aligned with the incident beam trajectory and focused on the position of the sample. This video image provides a visual control of the sample *i.e.* its position and orientation during basic operations like mounting/unmounting or during the centering operations. With no sample mounted, the same camera can be also used to visualize the X-ray beam by placing a Yttrium-Aluminium-Garnet (YAG) crystal film at the position of the sample. The images obtained with the YAG crystal are used to characterize the X-ray incident beam by calculating its position, width and height.

The ALBA Control System [1, 2] is based on the Sardana package [3, 4], a software for Supervision, Control and Data Acquisition (SCADA) and the Taurus framework [5, 6] for creating and supporting Graphical User Interfaces (GUI) and command line applications, dedicated to experiment control and data acquisition of both the accelerator and beamlines. Sardana is built on top of the Tango framework [7] extending it with a powerful python-based macro development environment. This allows to plug in custom procedures and complex macros via its *MacroServer*. It also provides a comprehensive access to the hardware through the *Device*

Pool and based on common and dynamic interfaces. The Taurus framework was originally conceived as an in-house solution for connecting client side applications to Tango device servers. It provides a user interface code for the Sardana suite based on the *Model-View-Controller* approach. Based on the Sardana and Taurus frameworks, the aim of the library is to embed in a single product the tools required by the scientists for sample and beam operation, providing a high-level and user-friendly interface integrated with the ALBA control system. To achieve this goal, the library has to act as a reliable link between the different beamline equipment involved in those procedures and the GUI delivered to the final users. The effort to integrate the OAV library to other control systems can be greatly reduced thanks to the capability of taurus to support models other than Tango.

The On-Axis-View (OAV) library could be conceptually divided in three different parts. The first one is the library core containing the abstract classes. These classes provide the engine to update the representation of the different beamline objects on the displayed image (canvas). The second one is a specific Taurus widget which serves to display the video image via the guiqwt library [8] and related image tools encapsulated in a menu bar. Third and last, a configuration module to define how the real equipment is integrated in the control system. According to a given configuration, the library creates specific objects representing all the elements involved in the sample/beam management and characterization. This improves the transferability of the library to different beamlines.

2. Technical Details

The OAV library has been written in python 2.7 [9] and organized as follows. First, a base core composed by two abstract python classes which provide a link between the real object in the experimental setup and its representation in the canvas. One class is the *ZoomAware* class which provides the set of transformations used for repainting any real object represented in the canvas according to the camera zoom. These methods keep track of the current zoom value and implement the coordinates transformation between the real coordinates respect to the laboratory coordinate system and the canvas coordinates. The other abstract class included in the library core is the *ObjectAware* class, which inherits from the ZoomAware class. It provides an additional set of transformations used for recalculating the canvas object position according to any change of the position of a certain real object. The second part is the front-end widget, which is a specialization of a TaurusWidget class. This class contains an ImageDialog class (from guiqwt) embedding the video image and a configurable toolbar menu as main access point to the tools. In our case, the ImageDialog object contains the video image supplied by the *video_last_image* attribute from a LImA [10] Tango device server, responsible for managing the OAV camera at the position of the sample. Since this widget inherits from a TaurusWidget class, it can be easily added to any other Taurus-based application. Finally, a configuration module which is used for setting up the library at each beamline according to the specific integration of the real equipment.

The library comes with a set of tools extending the default guiqwt ones and completing the user interface for the scientist to operate their experiments with control, confidence and precision (see Table 1). The group of tools currently provided with the library could be separated in three categories:

- **Base tools:** Specialization of default guiqwt tools which make no use of the OAV core classes.
- **Simple tools:** Customized default guiqwt tools which inherits from any of the OAV core classes.
- **Complex tools:** Customized simple OAV tools with a higher events flow complexity.

Table 1: The current OAV tools catalog is organized in three different categories: base*, simple† and complex‡ tools. Any of these tools can be added to the OAVWidget by adding its name to the list of tools passed as optional argument.

Tool	Description
`cross`*	Automatically draws a cross-hair at the optical axis position, which is assumed to be provided by an external source.
`save`*	Saves to a file the current video image including the canvas objects.
`centering`†	Calculates the distance between a point set by a mouse click (usually the sample position) and the beam position, and moves the selected point-object to the beam position.
`n-click centering`†	Calculates and centers the sample making the target point invariant over translations and sample rotations.
`circle`†	Draws a circle element in the canvas and displays its diameter on screen. The circle can be drawn in sample aware mode, which implies that it will move as attached to the sample.
`ruler`†	Draws a ruler on the canvas which can be used to measure any element in the image. The measured value is shown on screen. The ruler can be drawn in sample aware mode, which implies that it will move as attached to the sample.
`scale`†	Draws a reference scale on the left-bottom corner showing the length value of each axis. The values are dynamically updated for each change of the zoom value.
`beam`†	Draws a rounded shape according to the beam position and size.
`select position`‡	Selects a set of points in the canvas and stores the corresponding motor sample positions which can be retrieved for future operations.
`raster scan`‡	Define a grid of points whose positions are used to generate a 2D-scan. The grid of points defined by the tool can be used to execute a complex collection of data at each point via an external program (in our case, a Sardana macro executed by the MacroServer device). The tool also provides a callback method which changes the spot color background associated to each grid position (which represents the incident beam) according to a given color scale.

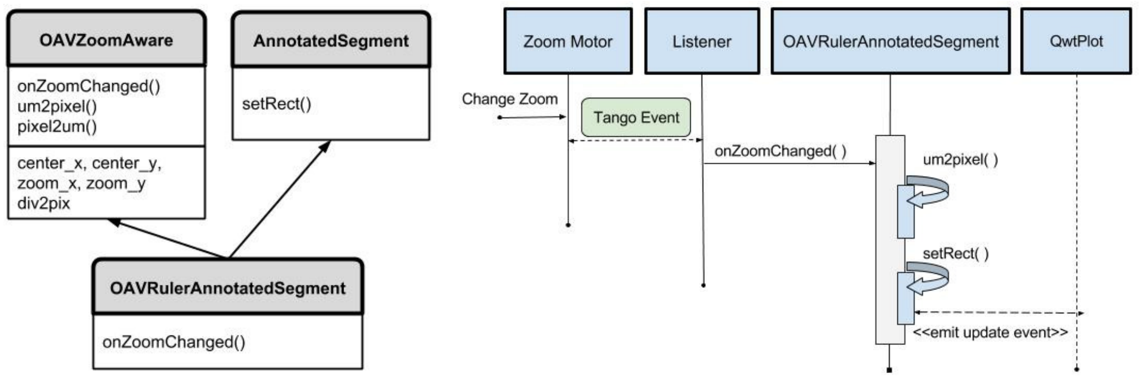
3. Tools implemented at ALBA beamlines

The OAV library is used on three of the seven beamlines at ALBA: the Powder Diffraction (MSPD-BL04), the Non-Crystalline diffraction (NCD-BL11) and the Macromolecular Crystallography (XALOC-BL13) beamlines. As we have previously mentioned, the OAVWidget class inherits from the TaurusWidget class and then, the addition of the OAVWidget to any other GUI is straightforward. Thanks to this, a common set of generic tools is shared across those beamlines, ready to be used by just configuring the tools through the configuration module.

Although the OAV library is shared across all beamlines, some tools need to be re-implemented according to the different equipment integration. For example, this is the case of the *n-click centering* tool, used at MSPD and XALOC, which has a different implementation due to their differences in the motorization of the sample. In other situations, the requirements are strongly dependent on the beamline workflow and equipment, driving us to design beamline-specific tools. An example is the *raster scan* tool used at XALOC, which provides the control over the whole experiment and shows the results on the GUI. A similar case is the *select position* tool used during a microdiffraction experiment at MSPD. Although the tool does not provide the control of the experiment, the user is able to select a set of points which can be used later on by a Sardana macro to measure the diffraction.

3.1. A simple tool: the ruler

One basic requirement from any beamline scientist is to measure the length of the objects found at the sample position i.e. the sample, the incident beam, etc. The *ruler* tool provides this functionality by enabling the user to draw a straight line between two different points in the canvas. The main object encapsulating this tool is the *OAVRulerAnnotatedSegment* class, which inherits from the ZoomAware class and the AnnotatedSegment class (Figure 1a). When the camera zoom changes, a Tango event is generated carrying the new zoom value and it is received by a listener contained in the OAVRulerAnnotatedSegment object (Figure 1b). Since the shape object owns the necessary methods to calculate its length in the real space (*i.e.* the distance between the two points in the real world corresponding to the two canvas points) the ruler object is redrawn and the new distance is reported as a text label next to the canvas object. Once the ruler object is created, the user can also modify the object by editing its edges and the displayed value gets updated automatically (Figure 3a).



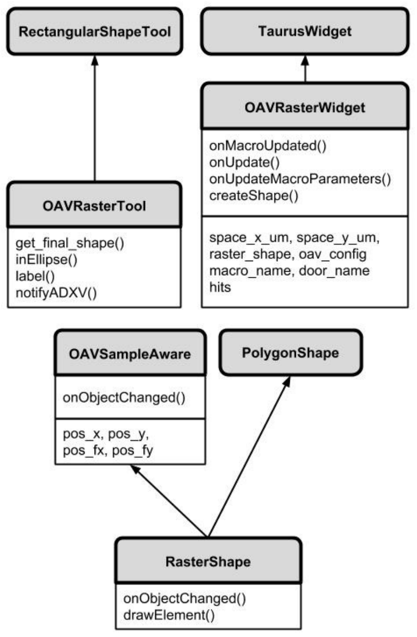
(a) UML class diagram (b) UML sequence diagram

Figure 1: UML class and event sequence diagrams corresponding to the ruler tool.

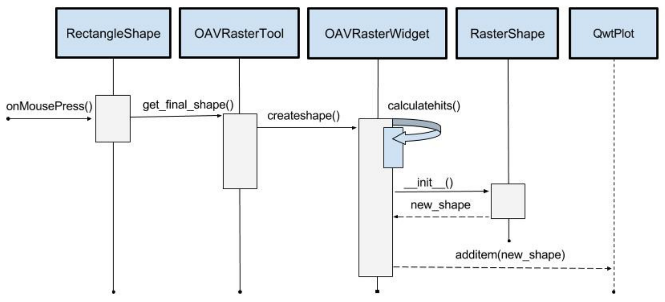
3.2. A more complex tool: the raster scan

Among the set of experiments conducted in a beamline, most of them can be carried out by executing a python code from a Command Line Interface (CLI). However, for some of them, a specific GUI greatly simplifies the experiment setup, monitoring and data acquisition. For instance, the proper characterization of the sample in any experiment of macromolecular crystallography is a key factor to decide the best sample region from where to collect the diffraction images. The raster tool helps the scientist by automatically define a series of collections to be performed at different points of the sample.

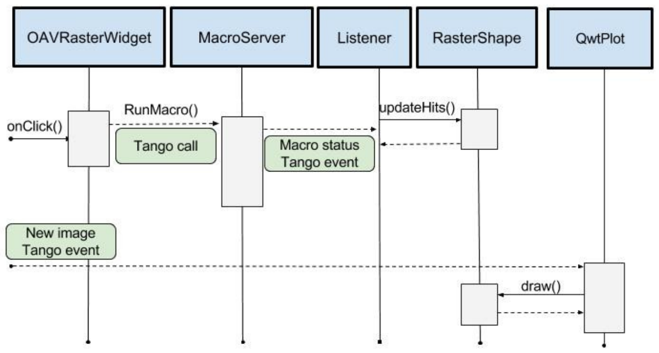
We show in Figure 2a the UML class diagram for the raster tool. The tool is composed of three different objects: the tool itself, containing the objects to link with the ImageDialog class; the *RasterShape* class, a specialization of the shape class to provide the custom behavior for the shape, in this case, a grid of rounded shapes; and finally, the *RasterWidget*, which is used as entry point to modify the current shape (*i.e.* adding or substracting scan points) and to start the scan process from the GUI.



(a) UML class diagram

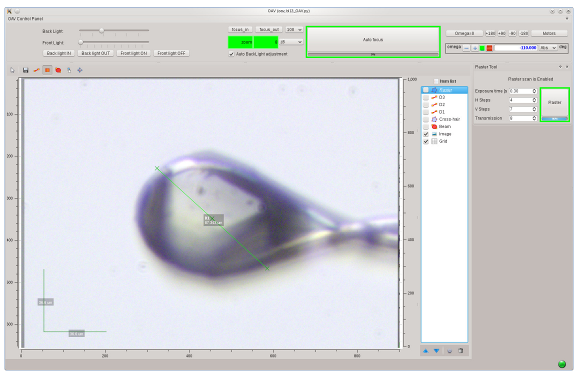


(b) Grid creation events diagram

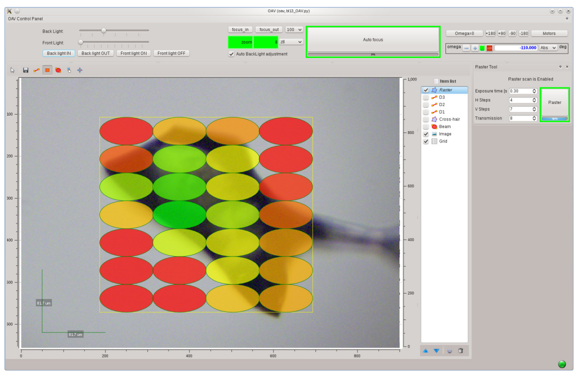


(c) Results display events diagram

Figure 2: UML class and event sequence diagrams of the raster scan tool.



(a)



(b)

Figure 3: Two snapshots of OAV-Raster application from XALOC beamline (BL13). **Left:** The ruler tool helps to measure the size of the sample in a pin. **Right:** During a raster scan the target spots are colored according to a color scale indicating the best spots for diffraction (in green).

Initially, the user selects a rectangular region over the sample image which is filled with rounded objects representing the incident beam. This generates a regular grid of points each of them representing a target position for a data collection (see Figure 2b). Once the grid is drawn, the user can add or substract target points manually by using the OAVRasterWidget controls. The scan process is an external procedure which returns the results of the scan via Tango events. Once the event is received, the results can be managed by the raster object. In our case, the results are represented coloring the spots according to a figure of merit function. The ellipses objects are redrawn with the proper color triggered by the Tango event responsible for updating the video image (see Figure 2c). In this way, the scientist can visualize which is the most convenient spot from where to take the complete collection of images. In Figure 3b we show a snapshot of the XALOC-BL13 beamline GUI used to perform the raster scans thanks to the this tool.

4. Conclusions

The OAV library provides a framework for developing either custom simple tools or more complex solutions which can be easily embedded in any other application. The common API and the configuration files gives the developer an easier and faster way to integrate the library. Apart from the generic tools shared across different beamlines, we have also shown how the library is used to provide beamline-specific solutions. Several tools has been designed and implemented for Powder Diffraction experiments, Non-Crystalline Diffraction or Macromolecular Crystallography taking advantage of the OAV library, obtaining clean and user-friendly GUIs while fulfilling the requirements of the final users.

Acknowledgments

The authors wish to acknowledge the work done by the whole ALBA controls group. Also acknowledge the support and feedback provided by the scientist from Powder Diffraction (MSPD-BL04), Non-Crystalline Diffraction (NCD-BL11) and Macromolecular Crystallography (XALOC-BL13) beamlines, an important part of the refinement design process of the library and tools. The authors also thanks R. Pastor for reviewing this manuscript.

References

[1] Fernández-Carreiras D *et al. The design of the ALBA control system: a cost-effective distributed hardware and software architecture* Proc. ICALEPCS 2011 Grenoble, FRBHMUST01.
[2] ALBA website: http://www.albasynchrotron.es
[3] Reszela Z, Cuní G, Fernández-Carreiras D, Klora J and Pascual-Izarra C *Sardana - A python based software package for building scientific SCADA applications* Proc. PCaPAC 2014 Karlsruhe, WCO206.
[4] Sardana website: http://www.sardana-controls.org
[5] Pascual-Izarra C, Cuní G, Falcón C M, Fernández-Carreiras D, Reszela Z, Rosanes M and Coutinho T M *Efforless creation of controls & data acquisition graphical user interfaces with taurus.* Proc. ICALEPCS 2015 Melbourne, THHC3003
[6] Taurus website: http://www.taurus-scada.org
[7] Tango website: http://www.tango-controls.org
[8] guiqwt website: http://pythonhosted.org/guiqwt
[9] Python website: http://www.python.org
[10] Homs A, Claustre L, Papillon E and Petitdemange S *LImA: a generic library for high throughput image acquisition.* Proc. ICALEPCS 2011 Grenoble, WEMAU011

DOI: 10.17199/NOBUGS2016.91

Graphical user interface and experiment control software at the MX beamlines at EMBL Hamburg

Ivars Karpics, Gleb Bourenkov, Marina Nikolova, Thomas R. Schneider

European Molecular Biology Laboratory (EMBL) Hamburg Unit c/o DESY, Notkestrasse 85, 22607, Hamburg, Germany

E-mail: `karpics@embl-hamburg.de`

Abstract. The EMBL Hamburg outstation, located on the DESY Campus (Hamburg Germany), operates two macromolecular crystallography beamlines P13 and P14 on the PETRAIII storage ring. Both beamlines are equipped with high-end instrumentation such as adaptive X-ray optics, configurable compound refractive lenses, beam conditioning units, kappa goniostats, PILATUS detectors, and automatic sample changers to fulfil the needs of the MX community. For the high-level control of the beamlines and data acquisition the MxCuBE (Gabadinho *et al.*, 2010) interface is used. The growing complexity of experiments requires easy and fast adaptation of the control software. The data model and overall architecture of MxCuBE allows implementing new data collection methods and strategies (large scale grid scans, serial crystallography experiments, etc.). EMBL Hamburg is part of the MxCuBE collaboration (http://mxcube.github.io/mxcube/) and actively participates in the software co-development. The overall infrastructure of graphical user interfaces (as running on the Qt4 library), experiment control, on-line data processing and the ISPyB experiment information system at EMBL Hamburg MX beamlines will be presented.

1. Introduction

The macromolecular crystallography (MX) beamlines P13 and P14 at EMBL Hamburg are in user operation since late 2012. With a minimum focus size of 30 x 20 μm^2 (divergence <0.15 μrad) and X-ray energy tunability between 4.0 and 17.5 keV P13 is well suited for crystallographic phasing experiments (Cianci *et al.*, 2015; Cianci *et al.*, 2016). Using an adaptive focusing system the beam can be defocused to a size of 140 x 70 μm. Additionally apertures are used to adjust the beam size. Crystals are positioned and rotated in the X-ray beam with an MD2 diffractometer (Cipriani *et al.* 2007) equipped with a mini-kappa goniostat. The micro-focus beamline P14 is capable of offering a 5 x 5 μm beam with a divergence below 0.3 μrad. In addition to the micro-focused beam a collimated, homogeneous, and quasi-parallel beam with a maximum size of 300 μm can be produced by using Compound Refractive Lenses (CRLs) as mounted in a transfocator (Vaughan *et al.*, 2011). Toggling between micro-focused and collimated conditions takes less than 20 seconds in either direction. P14 is tunable between 6 and 20 keV. For crystal handling an MD3 diffractometer with a vertical spindle axis is installed. By using a crystallization plate holder replacing the mini-kappa goniostat the diffractometer can be used with CrystalDirect^{TM} (Cipriani *et al.*, 2012; Marquez *et al.*, 2014) plates for *in situ* data collection. Both beamlines are equipped with PILATUS 6M-F detectors (DECTRIS AG, Baden, Switzerland) and sample mounting robots MARVIN (in-house development EMBL Hamburg). To ensure a robust network connectivity between the beamline components a subnetwork infrastructure clearly separated from the institute's main network is deployed. Before an experiment users can submit information about their samples and experiment conditions to the ISPyB system (Delageniere *et al.*, 2011). During the experiment for rapid transfer of data collected at a beamline an InfiniBand connection between PPUs (Pilatus Processing Units) acting as a primary data storage and a secondary data storage system is used. After a user visit experimental data are kept for several weeks. Processing results and information about the experiment are available also by remote access in ISPyB. Figure 1. describes the connection between beamline components accessible by the user.

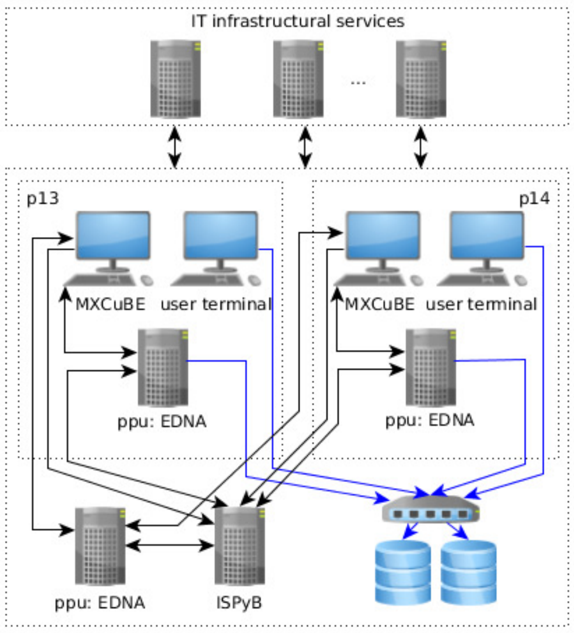


Figure 1: Beamline components and network topology for EMBL beamlines P13 and P14. At both beamlines the interaction of the user with the beamline takes place via a computer running MxCuBE. On the fly data processing is performed on a local PPU, while manual processing takes place on an additional PPU that can also serve as a hot spare in case one of the other PPUs fails. All network connections between these components are 1 Gbit/s (black arrows). High data rate InfiniBand connections connect the PPUs to the centralized TByte high performance (aggregated 1 GB/s write speed) storage (blue arrows). The ISPyB services run on a separated dedicated server. For data processing and structure solution dedicated multi-core systems with high-end graphics cards are deployed on each beamline user terminal.

For instrument control the TINE control system is used. TINE (Three-fold Integrated Networking Environment, P. Bartkiewicz, P. Duval, 2007) is a multi-platform, multi-protocol, and multi-architecture control system initially developed at DESY for the control of particle accelerators. It is based on a client-server principle with plug-and-play capabilities. For integration TINE includes client and server APIs for a wide range of programming languages and other instrument control environments (C, Java, Visual Basic, python, Labview, Matlab). All beamline devices are accessed via TINE device servers (Pazos *et al.*, 2008; Ristau *et al.*, 2014; Cianci *et al.*, 2016).

2. Experimental control and data management interfaces

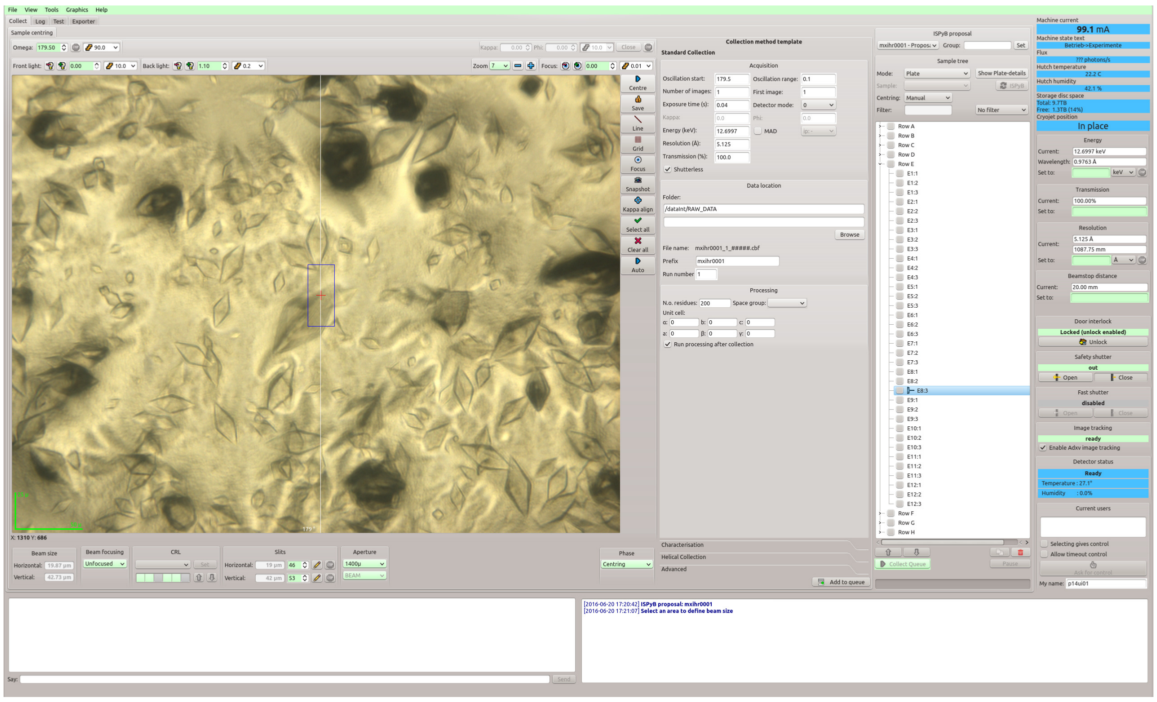
To provide users with an intuitive and robust interface to complex beamline functionalities, the MxCuBE interface for experimental control is used. Initially developed and supported by the European Synchrotron Radiation Facility (ESRF) MxCuBE is widely used in synchrotrons across Europe since 2005.

2.1. Middleware TINE services

While in earlier implementations of MxCuBE high-level beamline control operations, e.g. data acquisition sequences were implemented via SPEC (Certified Scientific Software, ESRF) commands or macros, on P13 and P14 such tasks requiring coordinated operation of more than one device are implemented into a set of middleware TINE services. For example, the TINE Energy Server simultaneously controls the undulator gap, the Bragg axis of the monochromator, and the second monochromator crystal to perform a change of X-ray energy. A TINE Transmission Server controls the configuration of attenuators for automatically setting the transmission as a function of energy. A dedicated TINE server combining both energy tuning and transmission adjustments provides an integrated functionality for the automatic acquisition of energy scans and X-ray fluorescence spectra. Crystallographic data acquisition is performed via the TINE Detector Server which coordinates the actions of the safety interlock, the Energy and Transmission Servers, the diffractometer, detector positioning, 2D X-ray detector, and a mechanical crash protection system. Further TINE services control the data management tools (in case of PILATUS detector the Furka and Grimsel tools as supplied by DECTRIS), and the rapid data delivery to the diffraction image display (ADXV, Arvai, 2015). The implementation of most middleware services follows the general architecture described in Franke *et al.* 2012.

2.2. MxCuBE implementation

MxCuBE is written in Python and accessible as an open source project through GitHub (https://github.com/mxcube/mxcube). The architecture of MxCuBE is divided in two major parts: a hardware control layer implementing the actual connections to the beamline devices and a user interface layer (Guijarro *et al.*, 2004). In the hardware control layer a hardware repository acts as a container of individually configurable hardware objects and is used as a data dispatcher. APIs to major control systems like EPICS, Tango, Sardana as well as to TINE middleware services are available. With respect to the user interface layer a Qt4-based version of MxCuBE is currently deployed at EMBL Hamburg. The Qt interface allows a fast and rapid implementation of new graphical objects and is easily adaptable to evolving MX experimental strategies. With respect to the previous version some changes were made to the overall layout to improve the ergonomics (Figure 2). The image from the MD2/3 on-axis viewing system (OAV, Cipriani F.,



Collection method template
Standard Collection

Acquisition
Oscillation start: 179.5
Oscillation range: 0.1
Number of images: 1
First image: 1
Exposure time (s): 0.04
Detector mode: 0
Kappa: 0.0
Phi: 0.0
Energy (keV): 12.6997
MAD
Resolution (Å): 5.125
Transmission (%): 100.0
✓ Shutterless

Data location
Folder: /dataInt/RAW_DATA
File name: mxihr0001_1_#####.cbf
Prefix: mxihr0001
Run number: 1

Processing
N.o. residues: 200
Space group:
Unit cell:
α: 0 b: 0 c: 0
a: 0 β: 0 γ: 0
✓ Run processing after collection

Phase: Centring

Machine current: 99.1 mA
Machine state text: Betrieb->Experimente
Flux: ??? photons/s
Hutch temperature: 22.2 C
Hutch humidity: 42.1 %
Storage disc space: Total: 9.7TB Free: 1.3TB (14%)
Cryojet position: In place

Energy
Current: 12.6997 keV
Wavelength: 0.9763 Å

Transmission
Current: 100.00%

Resolution
Current: 5.125 Å 1087.75 mm

Beamstop distance
Current: 20.00 mm

Door interlock: Locked (unlock enabled)
Safety shutter: out
Fast shutter: disabled
Image tracking: ready
✓ Enable Adxv image tracking
Detector status: Ready
Temperature : 27.1°
Humidity : 0.0%
Current users
Selecting gives control
Allow timeout control
My name: p14ui01

```
[2016-06-20 17:20:42] ISPyB proposal: mxihr0001
[2016-06-20 17:21:07] Select an area to define beam size
```

Figure 2: Graphical layout of MxCuBE Qt4 at EMBL Hamburg.

Castanga J. C., 2005) is displayed on the left-hand side of the MxCuBE main window. Tools to control sample position and orientation, change of focusing plane, lighting, zoom levels, etc. are grouped with this display. Controls for choosing different beamline optics configurations and for defining beam shapes are located underneath the OAV display. In the central part of MxCuBE the user can choose the data collection method and assign it to the respective sample. On the right-hand side status information about the beamline and synchrotron is displayed.

2.3. Serial helical line scans

Evolving macromolecular crystallographic experiments require more complex data acquisition and processing techniques, and more elaborate graphical user interfaces. At EMBL Hamburg we recently introduced the serial helical line scan data collection method (Gati *et al.*, 2014). In a serial helical line scan (SHLS) a rectangular region of interest in a sample consisting of crystals and matrix material in which the crystals are embedded is systematically exposed with an X-ray beam. Each helical scan involves a simultaneous translation and rotation of the sample with respect to the X-ray beam along a straight line (Flot *et al.*, 2010). While a crystal is traversing the X-ray beam during a helical scan a small rotation data set (Arndt U. W., 1968) is collected. By performing a series of helical scans any crystal present in the region of interest is hit by the X-ray beam and diffraction data is collected. Analysis of the images acquired in terms of whether or not diffraction patterns are detected and whether they originate from the same or from different crystals is performed subsequently. In our implementation of MxCuBE a serial helical line scan data collection starts with defining the region of interest that is mapped into the coordinate system of the diffractometer (Figure 3a and 3b). The user then chooses the data

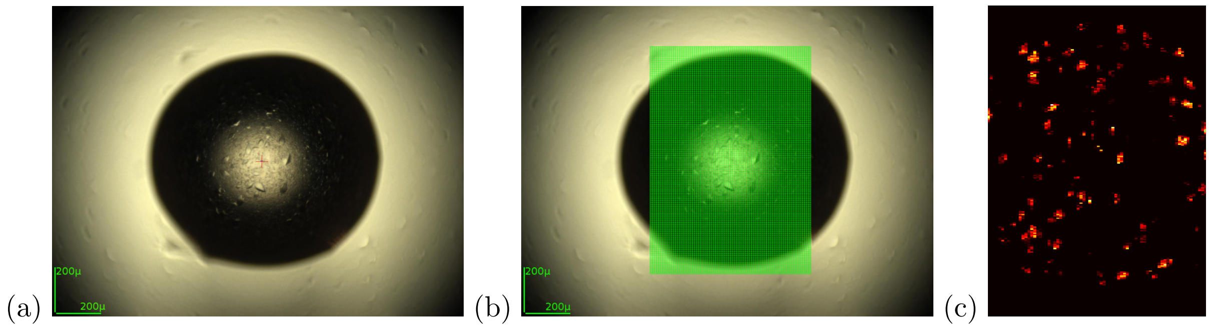
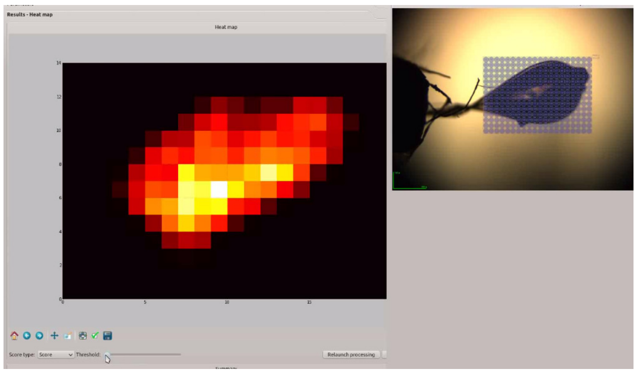


Figure 3: Serial helical line scan data collection. (a) Crystallization drop in a CrystalDirect^{TM} plate containing crystals as seen in the OAV image. (b) Definition of the region of interest to be covered by SHLSs. (c) Heatmap representing the diffraction quality observed across the region of interest defined in (b).

collection parameters including the number of parallel helical lines, the number of frames to be collected during each helical line, exposure time per frame, rotation range per helical scan, X-ray energy, and X-ray transmission. All data collection parameters are then sent to the the TINE detector server that executes the collection sequence. The general sequence involves validation of the input parameters, adjustment of the X-ray energy, transmission and resolution if requested, re-calibration of the detector if necessary, preparation of the diffractometer and the detector, for example, setting it up for multiple external triggers, opening the detector cover, starting the collection, and waiting for the detector and the diffractometer to finish. After the collection starts MxCuBE launches remote EDNA (Incardona *et al.*, 2009) parallel processing based on the DOZOR (Popov A. N., Bourenkov G. P., 2015) plugin to analyse the frames collected in terms of diffraction signals in real time. Processing results are sent back to MxCuBE via an xml-rpc server proxy. The results of the analysis are mapped onto a two-dimensional grid implemented as a hardware object (graphics manager) for assembling a heatmap of diffraction quality. The heatmap is presented to the user in MxCuBE in real time (Figure 3c). Once an SHLS is completed the heatmap becomes interactive for controlling a number of actions:

- Selecting different properties (total DOZOR score, number of diffraction spots, resolution, etc.) as a base for the heatmap.
- Inspecting a frame related to a specific element of the heatmap by clicking on the element of the heatmap.
- Physically moving the sample to a center position corresponding to an element of the heatmap by double-clicking.
- Creating a centring point for a subsequent data collection based on the location on the heatmap.
- Filtering scan results and automatically creating centring points for further data collections (Figure 4).

(a) 

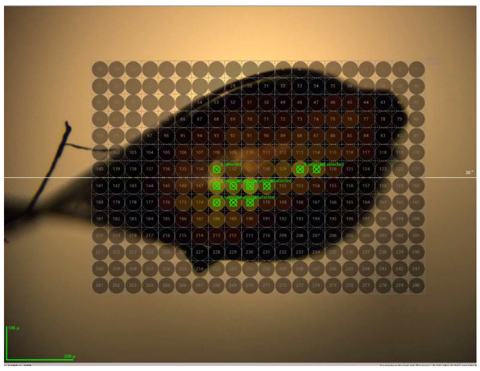
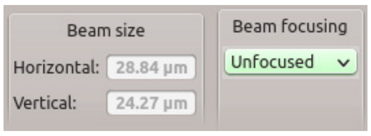
(b) 

Figure 4: Heatmap-based interaction with MxCuBE. (a) Relation between heatmap and physical sample; (b) Creation of centring points based on mesh scan results.

2.4. Beam focusing tool

Apart from the data acquisition capacities MxCuBE includes various beamline alignment, customization, and testing tools. For example, at beamline P14, a widget allows switching the beam focusing between micro-focused and unfocused mode within 20 seconds (Figure 5).

(a) 

Beam size
Horizontal: 28.84 µm
Vertical: 24.27 µm

Beam focusing
Unfocused

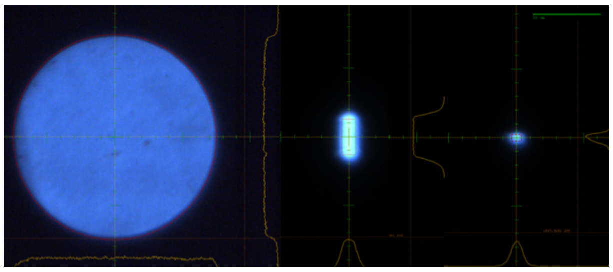
(b) 

Figure 5: Beam options configuration. (a) Widget for controlling beam size and focusing configuration; (b) Unfocused, vertically and micro-focused beams as imaged on the scintillator built into the MD3 diffractometer; the diameter of the unfocused beam is approximately 150μm.

The underlying hardware object acts on a collection of 12 motors in 4 groups (detector translation, beam control unit, experimental table and KB mirrors). TINE motor servers provide data (current position, status, velocity, etc.) about each group and depending on the requested configuration MxCuBE assembles the commands for the accurate positioning of the KB-mirrors.

3. Conclusion and future perspectives

EMBL Hamburg operates two macromolecular crystallography beamlines equipped with high-end instrumentation using a combination of the TINE and MxCuBE instrument control systems. MxCuBE as an experimental control interface is well suited and capable of including new data collection methodologies. The tight and fruitful MxCuBE collaboration encourages the standartization of beamline user interface across many beamlines and provides the same look-and-feel to the users at different facilities. Future developments are geared towards (1) increasing the robustness of the system both in terms of avoiding undefined states of the beamline and in terms of not permitting operator induced errors (2) improving the ergonomy of the user interface especially in terms of implementing intuitive and precise control of complex experimental scenarios (3) making the interaction with laboratory information management system both on the sample and on the data side seamless (4) improving diagnostic tools and procedures for trouble-shooting.

4. References

[1] Arndt, U.W., Wonnacott, A.J, The Rotation Method in Crystallography, North Holland, Amsterdam (1977)

[2] Bartkiewicz P., Duval. P. *Measurement Science and Technology*, **18** (2007), p. 2379

[3] M. Cianci, M. R. Groves, D. Barford and T. R. Schneider (2015) Acta Cryst. D**72**, 403-412

[4] Cianci, M., Bourenkov, G., Karpics, I., Kallio, J., Pompidor, G., Roessle, M., Cipriani, F., Fiedler, S. & Schneider, T. R. (2016). Submitted.

[5] Cipriani, F., Felisaz, F., Lavault, B., Brockhauser, S., Ravelli, R., Launer, L., Leonard, G. & Renier, M. (2007). *AIP Conference Proceedings* **879**, 1928-1931.

[6] Cipriani F, Rower M, Landret C, Zander U, Felisaz F, et al. (2012) *Acta Cryst.* D **68**: 1393-1399

[7] Delageniere, S., Brenchereau, P., Launer, L., Ashton, A. W., Leal, R., Veyrier, S., Gabadinho, J., Gordon, E. J., Jones, S. D., Levik, K. E., McSweeney, S. M., Monaco, S., Nanao, M., Spruce, D., Svensson, O., Walsh, M. A. & Leonard, G. A. (2011). *Bioinformatics* **27**, 3186-3192.

[8] Flot, D., Mairs, T., Giraud, T., Guijarro, M., Lesourd, M., Rey, V., van Brussel, D., Morawe, C., Borel, C., Hignette, O., Chavanne, J., Nurizzo, D., McSweeney, S. & Mitchell, E. (2010). *J. Synchrotron Rad.* **17**, 107-118

[9] Franke D, Kikhney AG, Svergun DI. *Nucl Instrum Methods A.* 2012;689:5259.

[10] Gabadinho, J., Beteva, A., Guijarro, M., Rey-Bakaikoa, V., Spruce, D., Bowler, M. W., Brockhauser, S., Flot, D., Gordon, E. J., Hall, D. R., Lavault, B., McCarthy, A. A., McCarthy, J., Mitchell, E., Monaco, S., Mueller-Dieckmann, C., Nurizzo, D., Ravelli, R. B., Thibault, X., Walsh, M. A., Leonard, G. A. & McSweeney, S. M. (2010). *J Synchrotron Radiat* **17**, 700-707.

[11] Gati C, Bourenkov G, Klinge M, Rehders D, Stellato F, Oberthur D, Yefanov O, Sommer BP, Mogk S, Duszenko M, Betzel C, Schneider TR, Chapman HN, Redecke L. (2014) *IUCrJ* **1** (Pt 2):87-94

[12] Guijarro, M., Berruyer, G., Klora, J. & Rey-Bakaikoa, V. (2004). *Nobugs 2004 Conference*, Villigen PSI, Switzerland, paper 00065

[13] Incardona, M.-F., Bourenkov, G. P., Levik, K., Pieritz, R.A., Popov, N. & Svensson, O. (2009). *J. Synchrotron Rad.* 16, 872-879

[14] Marquez JA, Cipriani F. Structural Genomics, vol 1091. *Methods Mol. Biol.* **1091**:197-203

[15] Pazos, A., Ristau, U. & Fiedler, S. (2008). *PCaPAC08*, edited by PCaPAC-CERN. Ljubljana, Slovenia

[16] Popov, A. N. & Bourenkov, G. P. (2015). DOZOR. European Synchrotron Radiation Facility, Grenoble, France

[17] Ristau, U., Kolozhvari, A. & Fiedler, S. (2014). *PCaPAC2014*, edited by PcaPAC-CERN.

[18] Vaughan *et al.*, *J Synchrotron Radiat.* 2011 Mar 1; 18(Pt 2): 125-133

[19] Zander, U., Bourenkov, G., Popov, A. N., de Sanctis, D., Svensson, O., McCarthy, A. A., Round, E., Gordeliy, V., Mueller-Dieckmann, C. & Leonard, G. A. (2015). *Acta Cryst.* D**71**, 2328–2343.

DOI: 10.17199/NOBUGS2016.104

MXCuBE 3 web application, on the way to next generation experiment control

M. Eguiraun[1], A. Milan-Otero [1], F. Bolmsten[1], J. Nan[1] , M. Thunnissen[1] , V. Hardion[1], D. Spruce[1] , M. Guijarro[2] , M. Oscarsson[2] , A. Beteva[2] , D. de Sanctis[2], G. Leonard[2], J. Meyer[2], A. Gotz[2]

[1] MAX IV Laboratory, Fotongatan 2, 225 94 Lund, Sweden.
[2] ESRF The European Synchrotron, 71 Av des Martyrs, 38000 Grenoble, France.

E-mail: `mikel.eguiraun@maxiv.lu.se`

Abstract.
Macromolecular Xtallography Customized Beamline Environment (MXCuBE) is a software platform that provides users of beamlines at synchrotrons an easy to use graphical environment. From one side it hides the complexity of the beamline hardware, facilitating normal operation, while on the other side provides routines for automated complex data collection strategies. The third evolution of this software is under development as part of the MXCuBE collaboration. A prerelease version has already been used in experiments at MAXIV and ESRF, the facilities leading the development. The main evolution compared to the previous versions is the transition to a web based environment, which is expected to facilitate remote data collection and on-line data analysis, among other things. This article explains the main features and technical details of MXCuBE v3.

1. Introduction

The MXCuBE project started in 2005 at ESRF [1, 2], with the objective of providing a unified and user-friendly software to the macromolecular crystallography (MX) beamlines. The software was designed to pace the increase in performance and automation at the MX ESRF beamlines. From the initial task of collecting single crystal diffraction data, MXCuBE evolved to include more advanced functionalities, enabling the assessment of diffraction characteristics of samples, complex and automatic data collection, and recording X- ray emission spectra and subsequent analysis. Furthermore it offered the possibility to control the beamline remotely; directly from the home laboratory (Remote Access).

In 2010, a collaboration for the development of MXCuBE started among the major synchrotron facilities in Europe with the aim of further developing MXCuBE. Today it is actively supported by the following partners: ESRF, Soleil, MAX IV, HZB, EMBL, Global Phasing Ltd, DESY and ALBA. And several new partners are considering to join the collaboration.

The current stable version of MXCuBE (release v 2.2) is developed in Python using the PyQt toolkit and nowadays represents the state-of-the art in terms of GUI for MX diffraction experiments. It enables the implementation of new data collection methods [3, 4] and hands-off automatic data collection. However, the continuous evolution of structural biology beamlines and data collection protocols necessitated the development of novel control software, that not only could keep up both with scientific drivers and user needs, but that could evolve with new technology. From this basis the project of MXCuBE v3 started.

One of the main reasons to implement MXCuBE 3 as a web application is to take advantage of the recent advances in user interface design coming from web technologies. As any web application, MXCuBE 3 runs in a web browser making maintenance and deployment of the client easy and facilitates user access. The distributed nature of web applications along with simple client installation makes MXCuBE 3 ideal for remote access usage. MXCuBE already supports the web-based LIMS (Laboratory Information Management System) for protein crystallography, ISPyB [5]. A seamless integration can further be achieved thanks to the use of a common platform, resulting possibly in a better performance and smoother operation. Taking advantage of this new technology the user interface has been completely redesigned, with the aim of enhancing the user experience by means of a feedback gathering from users, an improved experiment queue operation and sample management, and a better integration with LIMS system.

The main technologies in use are python-flask web framework [6] for tL he backend, and React JavaScript library [7] for the front-end, enhanced with several third party libraries for both components. Low-level control is achieved via Tango, Sardana and custom protocols, by means of the so-called *Hardware objects*, which are self-contained pieces of software which links to the underlying instrumentation control. These libraries are being reused from the previous versions of MXCuBE, hence compatibility between different versions is guaranteed.

MXCuBE v3 development is currently lead by MAXIV and ESRF. The first milestones defined in 2015 have already been achieved, when the first data collection experiment was successfully performed in June 2016 during MAX IV new MX beamline Biomax [8] commissioning. The next months will be dedicated to increase the stability of the application and to add new features that will be needed when the user operation starts. MXCuBE v3 will not only be the experiment control environment at Biomax beamline in MAXIV and the ESRF MX beamlines in the near future, but the experience acquired will serve for future software developments.

The structure of this paper is as follows: first the MXCuBE v3 development plan and the main goals of the project are described, the next section deals with the technologies adopted, and finally some conclusions will be listed.

2. MXCuBE v3 Development

A preliminary feasibility study was requested by the MXCuBE collaboration board, and the conclusions and experience gained from that study defined the technology stack to be used for the development. The real kick-off of the project happened in September 2015, at that time, MAXIV and ESRF defined together the development plan, specifying the main features and milestones, in addition, each milestone was subdivided in a set of work packages.

MAX IV decided to directly use MXCuBE v3 for the BioMax beamline from the very beginning of the beamline operation. Although this might look like a risky approach, the fact that the schedule of the installation of the beamline components was distributed over several months, and the commitment of both institutes allowed a successful development. Moreover, the old MaxLab as well as the ESRF beamlines have been used extensively for testing the developments (avoiding disruption of user operation).

In June 2016, shortly before the inauguration of the MAX IV facility, the first diffraction data was collected at the BioMax beamline. For such first experiment, the most important feature to implement was the so called 'Standard Data Collection', which is the most basic data acquisition. It implies configuring the beamline in order to get a suitable photon beam, as well as configuring the data collection by centering the crystal in the beam path and the oscillation to be performed by the diffractometer. For a more comprehensive list and description of the main features and milestones see the github page of the project [9].

Frequent meetings are held for planning and estimating the work, following a kind of distributed scrum (Agile methodology), where the review, planning and estimation of work is done every month. Technical discussions and face to face meetings for specific topics are also organised when needed. The project relies, apart from physical meetings, on Github and Skype for communication between the teams.

3. From Qt to Web

Although making MXCuBE available for the web is a major development effort, the whole existing Hardware Object layer is kept unchanged and is common among the different versions. In this way the application reuses the existing libraries for beamline control. The focus is on bringing the user interface to the web making the transition from MxCuBE v2 to MxCuBE v3 straightforward.

This section explains the main characteristics of the Hardware Objects library, as well as the main technologies in use in MXCuBE v3.

3.1. Hardware Objects

The MXCuBE Hardware Repository holds the description of the devices and equipments of the beamline as a set of XML files. Each file represents at least one Hardware Object, which is a self-contained piece of software that links the graphical layer to the underlying instrumentation control software. The most common communication protocols for accelerator control are available, e.g. Tango, EPICS, Tine, Sardana, Spec or EMBL Exporter. Hardware Objects can be composed in order to represent more complex equipment, like a diffractometer. And and they can also call methods from another Hardware Object, for example an object that handles the configuration of the beamline could retrieve motor positions from another Hardware Object. The asynchronous communication has been implemented using the concept of signals and slots.

3.2. User Interface

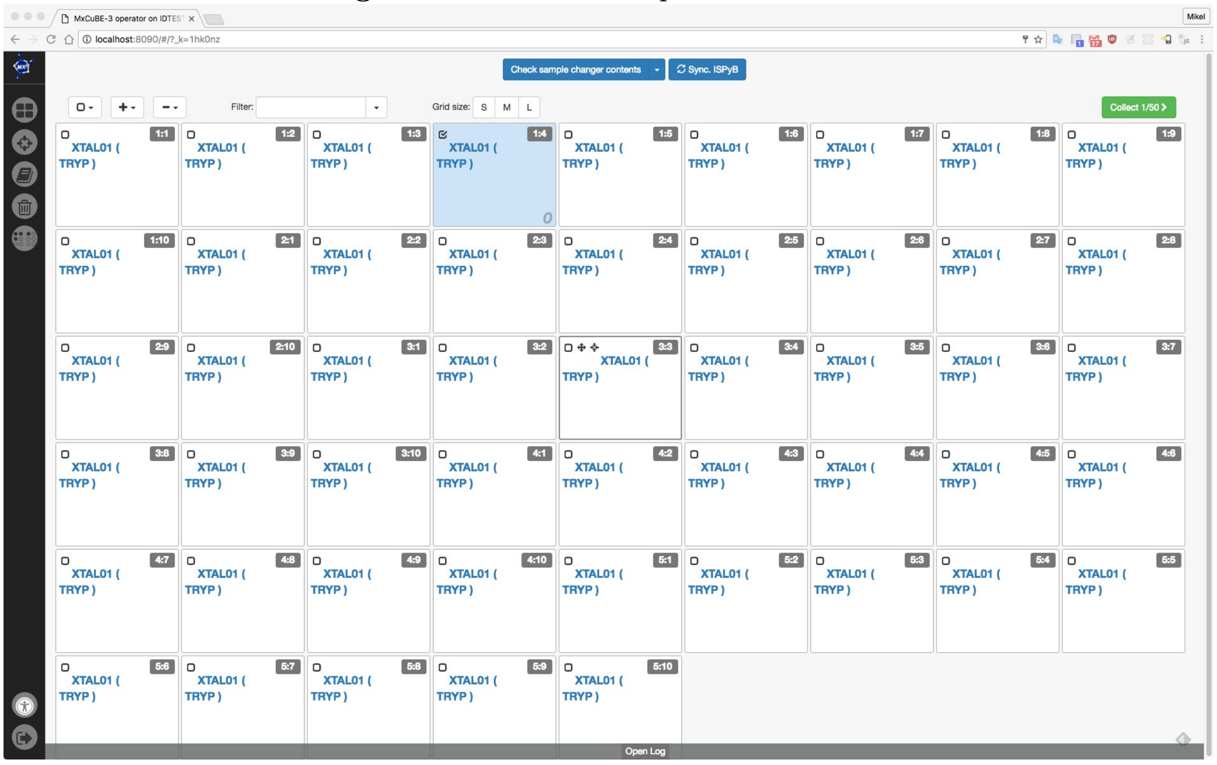
Since this new development moves from using Qt libraries into a web environment one of the first decisions (and discussions) was the user interface. The previous interfaces fulfil very well their mission, however a questionnaire was sent to nearly two hundred beamline users in order to detect any potential usability problems as well as to listen to suggestions. The overall feedback was positive, however there were clear needs to make improvements, such as the usability considering different user profiles, but also the advanced configuration and management of the experiment.

There are two main modes of operation with MXCuBE: automatic and manual mode. The first one consists of selecting a set of samples and selecting the appropriate workflow that will be applied to all of the samples, i.e. an automated, complex and sequential predefined data collection. For this purpose, the interface should present an easy way of inspecting all available samples, and associate a workflow only to the ones the user is interested in.

On the other hand, in a manual operation the scientist usually have a close view of the sample, he/she manipulates the diffractometer, centres the sample and manually selects the most appropriate parameters for the data collection. Several collection types are also available, and characterisation routines could give the user an initial proposal for the diffraction experiment. In any case, one could pre-configure all the experiment and then manually proceed with each sample.

Hence, since there are two main operation modes, the interface layout is split into different views, decoupling the sample lists and experiment configuration, see Figure 1, from the sample view. The new interface is intended to cope with increased sample throughput (several hundreds of samples each day), representing samples as cards, which is considered more appropriate compared to a hierarchical information display in a big tree as it was before. A card would allow

Figure 1. MXCuBE Sample Grid Interface.



to condensate information of sample details and performed analysis in a small space, and offers more possibilities regarding user interaction.

The Figure 2 shows the sample view together with the experiment tree for the current sample, basically the one that is mounted and a glance of what is coming. Ideally, the switching back and forth between the two views should be minimised, additionally, the design should avoid duplication of information whenever is possible. At the present stage of development the two abovementioned views have been implemented, which provide the most basic and important capabilities needed in the beamline operations. Additional views are currently in development.

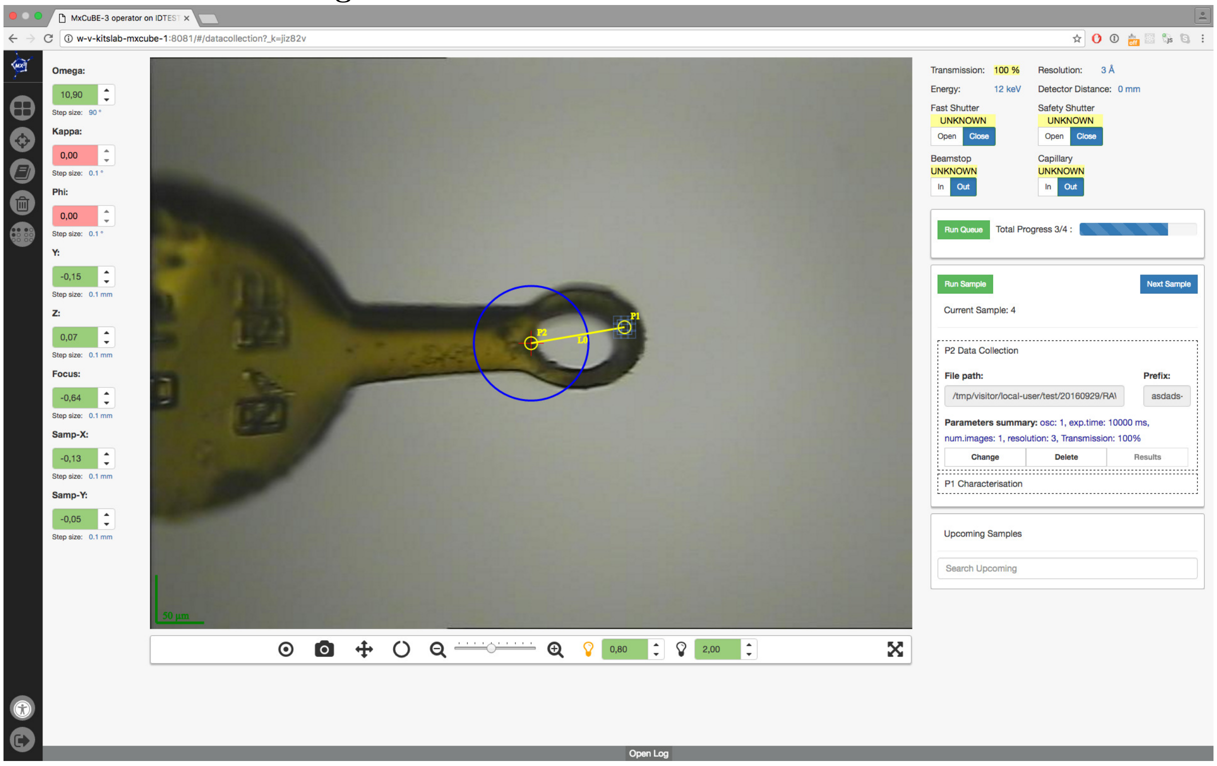
3.3. Backend

The backend server is based on Python Flask web server [6]. It is a Web Server Gateway Interface (WSGI) *microwebframework*, in which the core of the sever is designed to be simple but extensible, thus the developer needs to implement many services commonly found in heavy web servers. There are many flask extensions available, for example for dealing with databases, user authentication, form validation, templates, and so on. The backend also uses the gevent-flask extension to enable asynchronous network communication, in order to respond to concurrent requests in a very efficient manner.

Flask has been proven to be simple and effective to use, and integration within any WSGI-compliant application container can be done with little effort. Although at the moment, in the development phase, the Flask builtin web server is used.

The backend has been implemented following a Rest-like HTTP API calls. Where calls requested by the client are made via the standard HTTP methods (GET, POST, PUT, DELETE), and in the response apart from the result of the operation some data is included.. The different URLs are defined in a way that the functionality is easily understandable. For bidirectional asynchronous communication, MXCuBE v3 uses *socketio*, which is a library

Figure 2. MXCuBE Data Collection Interface.



MxCuBE-3 operator on IDTEST

w-v-kitslab-mxcube-1:8081/#/datacollection?_k=jiz82v

Omega: 10,90 Step size: 90 °
Kappa: 0,00 Step size: 0.1 °
Phi: 0,00 Step size: 0.1 °
Y: -0,15 Step size: 0.1 mm
Z: 0,07 Step size: 0.1 mm
Focus: -0,64 Step size: 0.1 mm
Samp-X: -0,13 Step size: 0.1 mm
Samp-Y: -0,05 Step size: 0.1 mm

Transmission: 100 % Resolution: 3 Å
Energy: 12 keV Detector Distance: 0 mm
Fast Shutter UNKNOWN
Safety Shutter UNKNOWN
Beamstop UNKNOWN
Capillary UNKNOWN

Total Progress 3/4 :

Current Sample: 4

P2 Data Collection
File path: /tmp/visitor/local-user/test/20160929/RA\
Prefix: asdads-
Parameters summary: osc: 1, exp.time: 10000 ms, num.images: 1, resolution: 3, Transmission: 100%

P1 Characterisation

Upcoming Samples

for communication that adapts automatically to the available protocols to ensure the best functionality, going from Websockets to AJAX polling. In the current state of the development, socketio is mainly used for re-emitting the hardware objects internal signals to the client, for example when a data collection has finished its execution.

3.4. Front end

In order to provide an enhanced user experience the front end has been completely redesigned. For that goal we rely the development on React Javascript library, which it is a library that takes advantage of a component-based design.

3.4.1. React React is an open-source Javascript library for building user interfaces using the concept of components [7]. Components make it possible to create independent entities encapsulating functionality, much like widgets in a traditional desktop UI framework. React is mainly concerned with the View part in the classic Model View Controller (MVC) design pattern and for more complicated applications another library needs to take care of the Model and Controller in MVC. It has fast become one of the most popular way to achieve a single page application (SPA).

In React the UI is described as a collection of components, where each component encapsulates code making it easier to reuse and test than in normal web-development. The main objective of a component is to provide a view for rendering data as HTML. The development of React was started by Facebook and is used both in Facebook and Instagram but was open-sourced in 2013 and is now used by many other websites.

Redux Redux is an open-source Javascript library that handles the state of an application, [10]. It is one of the most common libraries to use with React when developing complex applications because it takes care of both the Model and Controller (in the MVC pattern) and is therefore a good fit with React. It does this by keeping the state of the application in one place and only letting actions mutate it. By using this architecture, it is possible to get logging, hot reloading and time travel for debugging purposes with minimal effort.

One of the drawbacks of React is the fact that it does not recommend direct component to component communication, but it does not provide a solution either. Redux solves the problem by storing all your application state in one place, called a *store*. React components then dispatch state changes to the store, not directly to other components. An action can be triggered from the components themselves or from an outside source such as the server. Once the state is changed, it is passed down to the components of React and rendered as HTML. This unidirectional data flow makes the application much more predictable and easier to understand.

4. Conclusions

The existing and successful MXCuBE collaboration has been strengthened with the development of MXCuBE v3. New tools and procedures have been established by different partners with the common goal of providing good software for our users. The current stage of MXCuBE has already prove that it is in good rails with real experiments and helping with BioMax beamline commissioning. A lot of effort has been devoted to proper planning and technology analysis, however, there have been changes compared to what it was planned in 2015, but the MXCuBE team is really committed with the project and quickly adapts to changes. Furthermore, currently both Qt and Web versions of MXCuBE are in development, and this has lead to a very positive feedback where new features and ideas have been shared. The next months are devoted to the addition of the features in order to be ready for welcoming users, as well as to continue improving the current status of the software.

Acknowledgments

The authors are very grateful to the KITS and MX (MAXIV) and BCU and SB (ESRF) teams for their outstanding support during the development.

References

[1] Gabadinho J. et al. MxCuBE: a synchrotron beamline control environment customized for macromolecular crystallography experiments. J Synchrotron Radiation. September 2010, doi: 10.1107/S0909049510020005.
[2] MXCuBE project page, http://mxcube.github.io/mxcube/
[3] Zander et al. MeshAndCollect: an automated multi-crystal data-collection workflow for synchrotron macromolecular crystallography beamlines. Acta Crystallographica Sec. D, Vol. 71, Nov. 2015.
[4] D. de Sanctis et al., Facilitating best practices in collecting anomalous scattering data for de novo structure solution at the ESRF Structural Biology Beamlines. Acta crystallographica. Section D, Structural biology, 2016; 72 (Pt 3) doi:10.1107/S2059798316001042
[5] S. Delageniere et al., ISPyB: an information management system for synchrotron macromolecular crystallography. Bioinformatics. 2011 Nov 15; doi: 10.1093/bioinformatics/btr535.
[6] Python Flask web framework, http://flask.pocoo.org/
[7] React: a javascript library for building user interfaces, https://facebook.github.io/react/
[8] Thunnissen M. et al. The macromolecular crystallography beamlines BioMAX and MicroMAX at the MAX IV laboratory. Acta Crystallographica Section A: Foundations and Advances. 2015.
[9] MXCuBE v3 in github, https://github.com/mxcube/mxcube3
[10] Redux, a predictable state container for JavaScript apps. http://redux.js.org/

DOI: 10.17199/NOBUGS2016.9

Programs and techniques based on ROOT package for acquisition and sorting of the list mode data of the neutron detectors

E I Litvinenko[1,3], A A Bogdzel[1], A V Churakov[1], F V Levchanovsky[1], L Rossa[2], O-P Sauer[2], Th Wilpert[2]

[1] Frank Laboratory of Neutron Physics, Joint Institute for Nuclear Research, Joliot-Curie 6, Dubna Moscow region, 141980, Russia
[2] Helmholtz-Zentrum Berlin fur Materialien und Energie, Hahn-Meitner-Platz 1, Berlin, 14109, Germany

E-mail: litvin@nf.jinr.ru

Abstract. Works on the creation, implementation and adjustment of neutron detectors in the detector groups of the neutron centers require adequate and modern hardware and software tools and techniques. As one of the useful directions we consider measurements in list mode. List-mode data acquisition based on digital electronics offers many advantages over traditional acquisition methods and is rapidly increasing in popularity. Modern digitizers employing list-mode data storage are capable to perform on-line signal processing, greatly reducing data stream sizes and increasing data throughput rates. The report describes software solutions for dealing with large volumes of list mode data from neutron detectors, which are implemented in FLNP (JINR, Dubna) and run on the IBR-2M reactor and also reactor BER-II in HZB, Berlin. Software discussed in the report is designed for position-sensitive detectors with delay line readout and for the recorded data from the gaseous proton recoil telescope for fast neutron spectrometry

1. Introduction

A detector group of a neutron center has to solve daily a variety of non-typical tasks. These tasks may occur during configuration and commissioning of detectors, search for causes of faults, or with the development of new methodologies and the preparation of new types of experiments. That is why the requirements to the software from the detector group are specific and varied. The user oriented software (i.e., software developed in response to requests of external and internal users, who carry out varied measurements using neutron spectrometers) does not cover full range of these demands.

The software requirements as specified by the detector group in Frank Laboratory of Neutron Physics (FLNP JINR, Dubna) include:

- detailed diagnostic tools (in order to receive as much information from the detector as possible),
- autonomy (stand-alone operation, independence from the experiment control systems and local database management systems, etc.),
- fast deployment,

[3] The corresponding author

- flexibility (flexible configurability of parameters, easy modifications, configurable visualization tools),
- compact data formats,
- easy integration into the experiment control software (via communication components or via components which can be used autonomously),
- extensibility with respect to new equipment.

These requirements are the same for different types of detectors. Partially these demands have been realized in previous software like DeLiDAQ-1 [1] developed in collaboration with Helmholtz-Zentrum Berlin fur Materialien und Energie, Berlin (HZB, former HMI).

Recently we have started to follow such requirements in our new developments. Especially important is the first item - to get full information of measured parameters. The most complete information from the detector and about operation of the readout circuit can be extracted by list mode data. The list mode is a special measurement mode in which each event and its precise time of occurrence (timestamp) is recorded sequentially, with selected properties. List mode data can be used as a diagnostics tool to verify the performance of the detection system, but also to find time correlations between signals. This mode of the measurements corresponds to the modern trend of user oriented experimental techniques. Not only in the acquisition techniques for neutron scattering, the same trend is found in related fields [2], and at the present time a new international standard for the data format for list-mode digital data acquisition used in radiation detection and measurement is being developed by the Technical Committee 45 [3] on Nuclear Instrumentation of the International Electrotechnical Commission.

As the most suitable software for work with the event data is well known ROOT package [4]from CERN, due to its concept of the trees, designed for storing large number of same-class objects, with excellent implementation. Another reason to use this package is its compact data format. ROOT provides many other opportunities as an intensively used toolkit with complete suite of C ++ classes for fitting, linear algebra, visualization of large datasets, graphical user interfaces, etc. The report describes the software solutions for the list mode data, some of which are already implemented and used by the detector professionals while others are under development. Software discussed in the report is designed for position-sensitive detectors with delay line readout and for the recorded data from the gaseous proton recoil telescope for fast neutron spectrometry [5].

2. List modes of the DeLiDAQ-1 system

DeLiDAQ-1 hardware [6] was designed for acquisition and storage of data from one- and two-coordinate multi-wire proportional chamber based position-sensitive detectors with delay line readout. A few dozens of such electronic modules with PCI interface are in use in FLNP, HZB, INP (Řež, Czech Republic), and in a number of Russian neutron centers (see overview [7] and references in [8]).

DeLiDAQ-1 detector software [1] developed on the basis of ROOT libraries [4] saves all measurement results and parameters into the file in ROOT format, at that each single measurement is saved into separate folder inside this file. Although DeLiDAQ-1 software was designed for the detector groups, this program has the remote communication modules and thus can work under control of the external experiment control program. The software offers, in addition to standard histogramming modes, interface for two types of measurements in list mode: the conventional and the packed list mode. These list modes have not been properly described in the earlier publications.

In the conventional list mode the measurement lasts during predefined time but can be interrupted by the user. The list mode can be started in the raw regime (without filtering of incoming data flow by FPGA), and in the regime with filtering (in which FPGA inspects the data stream in order to avoid incomplete events and events that do not meet the parameters of the delay line), and in both cases the mode can be with or without time-of-flight (TOF). The data flow is recorded directly to disc without any transformations. The resulting binary data file can be converted at any time by the same program into the sets of ROOT histograms and ROOT trees, which are stored in a special folder inside the measurement data file.

In the packed list mode the measurement in the TOF regime with filtering lasts infinitely until it is stopped by the user. The program converts the filtered events into a more compact binary format (32 bits per event) in real-time. When the size of the binary data file exceeds the limit of 2 GB, it is closed and a new file is started. During the measurement the user can see the evolution of the ordered histogram on the screen (typically, it is 1D TOF histogram), which is stored in the measurement folder inside ROOT data file

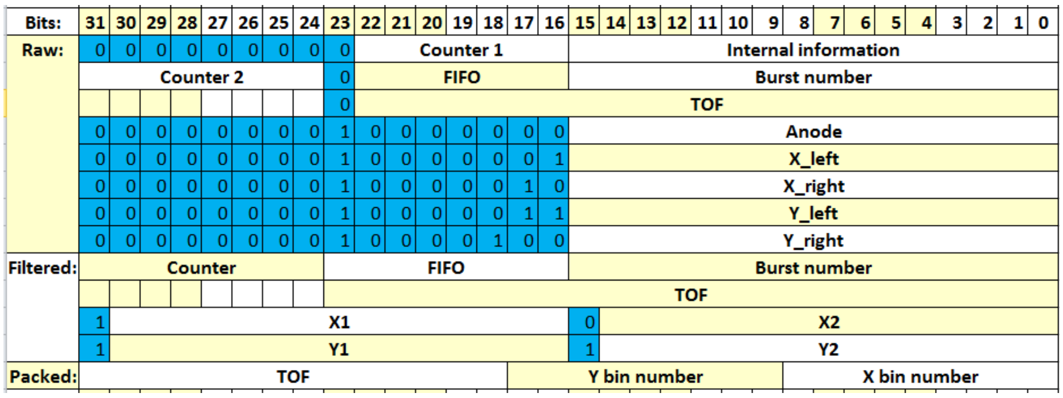
2.1. Internal structure of the encoded event data of DeLiDAQ-1

The internal structure of the encoded event data for these list modes (the conventional raw list mode, the conventional filtered list mode, and the packed list mode) is shown in figure 1.

Raw data. In the raw data stream, taken from the hardware, the header of three 32-bit words indicates the start of a new trigger window. Zero, one or more hits of the detector can happen within one trigger window interval. Since the system reads the information from the detector with a delay line, one detector hit includes a set of times of arrival of signals - from the anode, and from both ends of the two (or one) cathodes relative to the trigger window. Ideally single event from 2-dimensional detector contains 8 32-bit words (the ideal raw event structure is shown in figure 1).

Filtered data. In the filtered data stream, taken from the hardware, the header contains two 32-bit words, and other two words (or one word in one-dimensional case) contain 4 times of arrival of signals from both ends of cathodes relative to the anode signal. A single filtered event from 2-dimensional detector contains 4 32-bit words.

Packed list mode data. In the packed list mode case each binary data file has a textual header. The first four named sections of the header are normally filled by the information transferred by external experiment control software (like CARESS in HZB) to DeLiDAQ-1 and contain experiment specific information. The last section of the header is filled by the information about the file (path, start time of the measurement, number of file, list mode request parameters). The next four 32-bit words of the binary data file contain two timestamps: the time of saving the buffer and the time of reading from the board. These two timestamps accompany each subsequent portion of the 1024 events. After them the event list is placed, where each 32-bit word contains 9 bits for x bin number of the hit, 9 bits for y bin number, and 14 bits for TOF channel numbers (i.e. maximal resolution of the data is 512x512x16384). The ranges of X and Y channels correspond to the ranges of the main 2D histogram requested by the user.



Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Raw:	0	0	0	0	0	0	0	0	0	Counter 1							Internal information															
	Counter 2								0	FIFO							Burst number															
									0	TOF																						
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	Anode															
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	X_left															
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	X_right															
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	Y_left															
	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	Y_right															
Filtered:	Counter								FIFO								Burst number															
									TOF																							
	1	X1															0	X2														
	1	Y1															1	Y2														
Packed:	TOF														Y bin number									X bin number								

Figure 1. DeLiDAQ1 event data structure for three list mode options. Bits marked as blue are used to detect the header and coordinates in the data stream.

2.2. Conversion of event data

It has already been mentioned that the program has the event data conversion tools. Besides converting to text, from the raw or filtered data the program can create a set of requested histograms, and optionally the ROOT tree with the event parameters for further analysis. The ROOT tree contains all event-by-event information available from the system. For example, in the raw TOF case, the tree of good events (named 'good' tree) will have 11 components (leafs): index, TOF, x1, x2, y1, y2, anode, internal index, FIFO buffer counter, the reactor burst number, taken from hardware, the burst number defined according to TOF changes. And another tree of bad events ('bad' tree) contain 18 leafs: index, TOF, the last good event index, anode, x1 (or 0), x2, y1, y2, first copy of internal index, second anode (or 0), second x1 (or 0), second x2, second y1, second y2, second copy of internal index, the burst number from hardware, the burst number according to TOF changes, and FIFO buffer counter. Careful check of different combinations of these parameters can reveal different problems, starting from search of bugs in the firmware and finishing with the tasks on the optimization and the noise analysis. The ROOT Tree Browser is an excellent tool for such studies for the beginners, while the ROOT command interpreter permits a user to get picture with very sophisticated selections and cuts with only one line of code. With conversion of the filtered data we got more compact ROOT tree, because number of components (10 leafs for 'good' tree and no 'bad' tree) is less than in raw case. The table 1 illustrates the level of compactness of the discussed data formats on the real world examples taken during adjustment works with the detectors on the reactor IBR-2 in FLNP. One can see that the ROOT tree files, fully ready for plotting and express analysis, have the size not more than 2/3 from the original binary data file. The sizes of the ROOT tree files can be further reduced by increasing the compression level parameter (the program used the smallest compression ratio, that is the default value) and by the optimization of the data types of the tree leafs.

Table 1. The file sizes in bytes divided to the number of good events (i.e. number of bytes per good event in the file). Real world examples of the DeLiDAQ-1 list mode data from the detector.

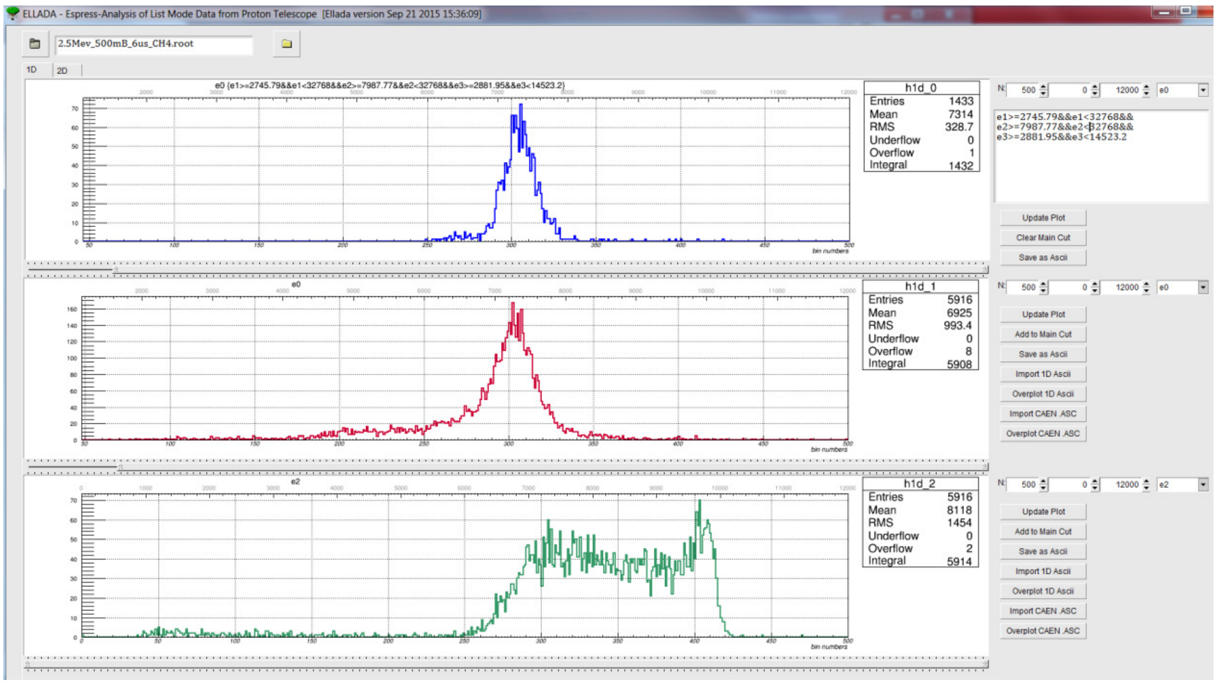
Mode (all TOF):	Binary file:	ROOT tree file:
Raw (2D)	*(Ideal value 32)* 36; 55	21; 30
Filtered (2D)	16	11
Packed (2D)	4	-
Raw (1D)	*(Ideal value 24)* 64	34
Filtered (1D)	12	8

3. ELLADA program

The program ELLADA was developed for the analysis of the list mode data from the neutron spectrometer based on a proton telescope with electronic collimation of recoil protons, developed and created in the FLNP [5]. A proton telescope measures the energy of recoil protons in a gaseous medium with the use of spectra reconstruction procedure. In the FLNP telescope, the collimation occurs by the coincidence of pulses from recoil protons in several tubes of the device and, subsequently, in the sorting of experimental data using software in the chosen ranges of amplitude distributions of the signals picked off the cathodes. Although the very first prototype of the detector was equipped with electronics in the standard CAMAC, as well as the locally developed software "Lada 2010" [5] for data acquisition and processing, further steps were taken to move to the N6724 digitizer [9], produced by Caen.

For measurement of the telescope data it was decided to use the program MC2Analyzer provided by the Caen and thus store the list mode data in its binary format.The measured data (of four telescope channels - anode and three cathodes) are recorded as four files, each of which contains a timestamp value and the corresponding energy (signal amplitude) for a single measurement channel. The assignment of the program ELLADA is to assemble these independent chains of signals into a united set of events and to provide convenient interface for viewing and interactive cutting of these data using different selection criteria.

To accomplish this, the program converts the raw data into a set of four independent trees in the ROOT format, and then starts sorting intensively taking advantage of ROOT package. When sorting, ELLADA collects into one event all hits at the cathodes, which occurred close in time to a hit at the anode. The end result is also stored as a tree in the ROOT format. After sorting the program shows the various components of sorted events in form of 1D and 2D plots. The succeeding search of selection criterion is usually done by the choice of energy ranges for the various detector channels and the timing relationships between them (figure 2). Found selection criteria can be saved into the text files for further use.



```
ELLADA - Espress-Analysis of List Mode Data from Proton Telescope  [Ellada version Sep 21 2015 15:36:09]
2.5Mev_500mB_6us_CH4.root

e0 {e1>=2745.79&&e1<32768&&e2>=7987.77&&e2<32768&&e3>=2881.95&&e3<14523.2}

e1>=2745.79&&e1<32768&&
e2>=7987.77&&e2<32768&&
e3>=2881.95&&e3<14523.2
```

h1d_0	
Entries	1433
Mean	7314
RMS	328.7
Underflow	0
Overflow	1
Integral	1432

h1d_1	
Entries	5916
Mean	6925
RMS	993.4
Underflow	0
Overflow	8
Integral	5908

h1d_2	
Entries	5916
Mean	8118
RMS	1454
Underflow	0
Overflow	2
Integral	5914

N: 500 0 12000 e0 (panel 1); N: 500 0 12000 e0 (panel 2); N: 500 0 12000 e2 (panel 3)

Figure 2. The main window of the user interface of the program ELLADA.

4. CaDeLin program

The program CaDeLin is being developed in the FLNP as part of the pilot project on the application of the digitizer N6730 [10] for the neutron position-sensitive detector with delay line readout. In this project we use five channels of the digitizer running DPP-PHA firmware to collect the positional signals from the detector (one anode and four cathodes). For each channel we collect the timestamp, the amplitude and the waveform (using so called mixed mode of the digitizer with the configuring of the desirable content of the stored event). The waveforms are required to us in order to increase the accuracy of calculating the position. To get the tool able to perform necessary to us measurements in mixed mode, we started to develop our own data acquisition program based on the CAENDigitizer library. After the measurements the program should build events from the chains of hits. The criterion of the hits selection is that all four hits of cathodes must happen close in time to but after the hit of anode, as well as the condition for the delay line length must be satisfied.

We decided to store the raw data directly to the ROOT tree format. The data are saved during the measurement in form of five independent trees inside a single file, while the complete events are being built out of them later. With our parameters using a fairly simple data readout circuit, we collect information without losses at input frequencies up to 130 kHz trigger rate (and up to 300 kHz when for testing purposes we have configured the measurement without reading and saving the waveform). The tests of very long measurements (a night with 70 kHz trigger rate) were successful up to the full overflow of the hard disk for data files, each time after reaching 100 GB the current file testing was saved, and a new file was produced automatically by ROOT (our file sizes are less than 60% of the simple binary variant). We have set the parameters of the readout configuration and have measured the response of the real detector (figure 3). The works on the optimization of the stored information and the sorting algorithm are continued.

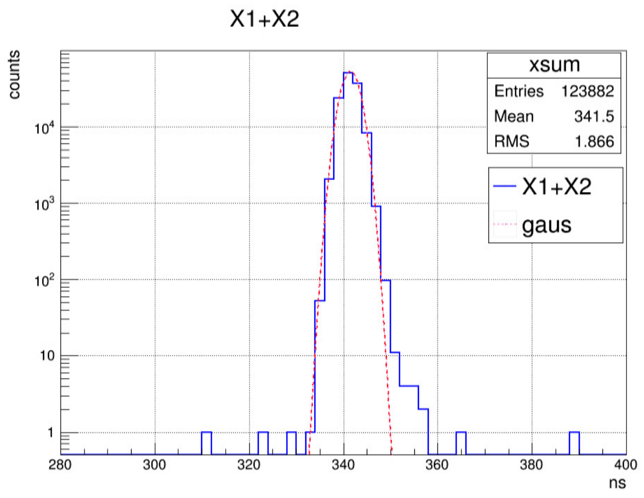


Figure 3. One of the first measurements with CaDeLin program – X1+X2 histogram of the neutron detector

5. Conclusions

We would like to draw attention to the techniques that are currently becoming more and more relevant for neutron instrumentation developments, and to offer approaches that seem to us effective, but still not often used in this field. It can be argued that the use ROOT for sorting and building of events greatly simplifies these tasks, while for further analysis of list-mode data this software is seems to be the best choice.

Acknowledgments

The authors wish to acknowledge many colleagues in Dubna and Berlin for preceding joint works related to the creation and use of the system DeLiDAQ-1 and the proton telescope, and especially Burckhard Gebauer for his encouragement of the idea of use ROOT for list mode data in the very beginning of the DeLiDAQ-1 project.

References

[1] Levchanovsky F V, Litvinenko E I, Nikiforov A S, Gebauer B, Schulz C and Wilpert T 2006 *Nucl. Instruments Methods Phys. Res. Sect. A* **569(3)** 900-4, See also 2007 *Nucl. Instruments Methods Phys. Res. Sect. A* **572** 1004
[2] Peräjärvi K, Keightley J, Paepen J, Tengblad O, Toivonen H 2014 *Publications Office of the European Union* EUR **26715**; doi: 10.2788/88299; http://publications.jrc.ec.europa.eu/repository/handle/JRC90741
[3] http://www.iec.ch/dyn/www/f?p=103:23:0::::FSP_ORG_ID,FSP_LANG_ID:1244,25
[4] Brun R, Rademakers F 1997 *Nucl. Instrum. Meth. A* **389(1)** 81-6, See also http://root.cern.ch
[5] Milkov V M, Panteleev T T, Bogdzel A, Shvetsov V N, Kutuzov S, Borzakov S B and Sedyshev P V 2012 *Physics of Particles and Nuclei Letters* **9(6)** 508-16
[6] Levchanovski F V, Gebauer B, Litvinenko E I, Nikiforov A S, Prikhodko V I, Schulz C and Wilpert T 2004 *Nucl. Instruments Methods Phys. Res. Sect. A* **529(1–3)** 413-6
[7] Kulikov S A, Prikhodko V I 2016 *Physics of Particles and Nuclei* **47(4)** 702-10

[8] Litvinenko E I, , Ryukhtin V, Bogdzel A A, Churakov A V, Farkas G, Hervoches Ch, Lukas P,

Pilch J, Saroun J, Strunz P, Zhuravlev V V 2017 Upgrade of detectors of neutron instruments at Neutron Physics Laboratory in Řež *Nucl. Instruments Methods Phys. Res. A* **841** 5–11
http://www.sciencedirect.com/science/article/pii/S0168900216310531
(free access to the article until 4-Dec-2016)
[9] http://www.caen.it/csite/CaenProd.jsp?parent=12&idmod=626
[10] http://www.caen.it/csite/CaenProd.jsp?parent=12&idmod=773

DOI: 10.17199/NOBUGS2016.16

"Manyo-Lib" Object-Oriented Data Analysis Framework for Neutron Scattering

Jiro Suzuki[1], Yasuhiro Inamura[2], Takayoshi Ito[3], Takeshi Nakatani[2] and Toshiya Otomo[1]

[1] High Energy Accelerator Research Organization (KEK), 1 Oho, Tsukuba-city, Ibaraki, Japan.

[2]Japan Atomic Energy Agency, 2-4 Shirakata, Tokai-mura, Naka-gun, Ibaraki, Japan.

[3]Comprehensive Research Organization for Science and Society, 162-1 Shirakata, Tokai-mura, Naka-gun, Ibaraki, Japan.

jiro.suzuki@kek.jp

Abstract. "Manyo Library" (ML) is a standard framework for developing data analysis software for neutron scattering experiments. The data analysis software based on ML has been installed on the 16 instruments in Materials and Life Science Experimental Facility of J-PARC. ML is a C++ class library which was designed so as to handle large scale data in high-performance and has common and generic analysis functionalities for neutron scattering experiments. Users can perform data reduction and analysis on the C++ library through the Python interface. Data containers for one, two and three dimensional histograms are prepared in ML. NeXus format files can be created on ML from the data containers with NeXus-C-API. In 2016 a new interface between the data containers and NeXus file was developed with HDF5 library on ML, and data input/output (I/O) rate between the present and new interfaces is compared. The I/O rate of the new interface is two or four times faster than that of the present interface.

1. Introduction

Japan Proton Accelerator Research Complex (J-PARC) is a proton accelerator complex, and Materials and Life Science Experimental Facility (MLF) is a user facility providing pulsed neutron and muon sources for experiments in J-PARC. Twenty-one instruments for neutron scattering experiments have been installed in MLF, but requirements of data reduction and analysis software for each instrument are different among many instruments and scientists. Thus the MLF data-analysis environment group decided to provide a software framework for developing data-reduction and analysis software which can be applied to each aim of instrument and scientist.

Our basic concept of analysis environment is to provide a framework which has common and generic analysis functionalities for neutron scattering experiments [1, 2]. The framework, "Manyo Library" (ML), is a standard framework for developing data analysis software. ML is working on the 16 beam lines which are shown in Figure 1. Because ML should be maintained and improved by many scientists, it is developed on object-oriented methodology. Origin of the name ML is "Manyo-shu" which is a Japanese classical and one of the most famous and oldest anthology edited in 7th and 8th centuries in Japan. The poems were composed by all sorts of people: emperors, warriors, fisherman, etc. We think that ML can contribute to developing data analysis software utilized by all sorts of scientists using neutron scattering instruments.

ML is a C++ framework and class library which was designed so as to handle large scale data in high-performance. The class library is wrapped by a Python user interface created by Simplified Wrapper and Interface Generator (SWIG). Users can perform data reduction and analysis on the C++ library through the Python interface. Figure 2 shows a screen shot of the graphical user interface working on the chopper spectrometer, BL01, and the data-analysis software based on ML is called from the interface. ML is a software component in the software framework in MLF/J-PARC[3]. The framework is available on various operating systems: Linux, MacOSX and Windows, and the binary install packages have been provided.

Design of data containers is one of key issues of the framework; efficient and simple data containers are required, because it determines system performance including development efficiency of application software on the framework. The design concept was discussed and introduced in detail in the previous papers [1, 2].

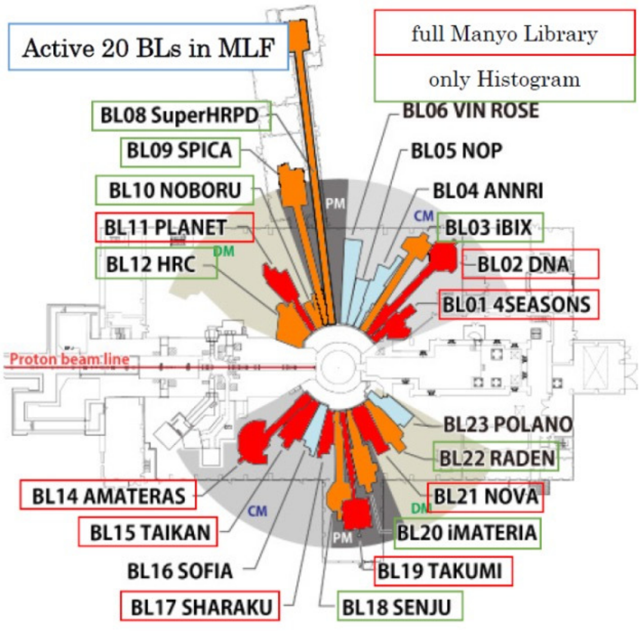
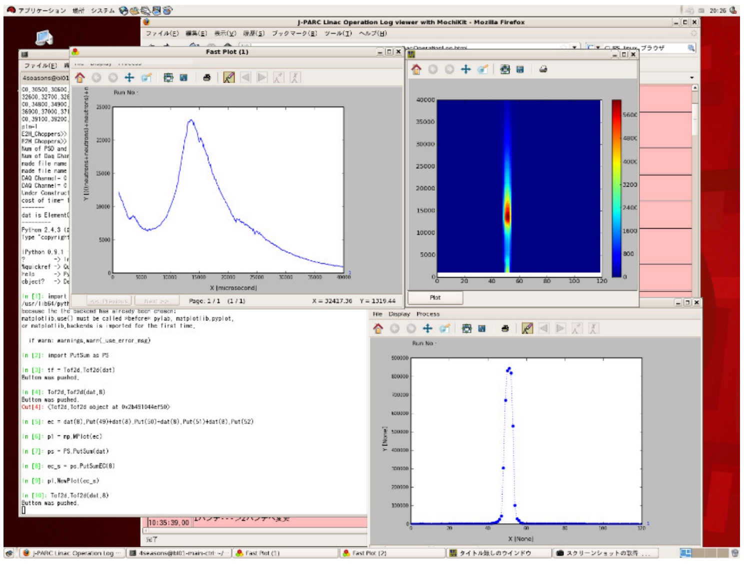


Figure 1: Floor layout of MLF and ML are working on the 16 beam lines indicated by orange and green boxes.

"ElementContainer" (EC) is a simple and fundamental data container in ML, and a one-dimensional histogram with its error values can be stored in an EC, where any number of vector<double> objects can be handled with their names. Basic mathematical arithmetic-operators and statistical operators provided in ML can operate ECs with error-propagation. Data containers for two- and three-dimensional histograms, "ElementContainerArray" (ECA) and "ElementContaienrMatrix" (ECM), are prepared, and raw data and analysis data are stored in the EC family on ML.

Figure 3 shows the structure of ECM. The number of vector objects in an EC is Nv. Any number of EC and ECA objects can be stored in an ECA and ECM, and the number of EC and ECA in a container is N_{EC} and N_{ECA}, respectively. The procedures of memory allocations and their garbage collections have been prepared in the containers. Users can build data analysis software in Python environment by using the operators prepared in the C++ library; ML provides a rapid prototyping environment for data analysis software without coding in C++ language.



```
In [1]: import
/usr/lib64/pyth
because the the backend has already been chosen;
matplotlib.use() must be called *before* pylab, matplotlib.pyplot,
or matplotlib.backends is imported for the first time.

  if warn: warnings.warn(_use_error_msg)

In [2]: import PutSum as PS

In [3]: tf = Tof2d.Tof2d(dat)
Button was pushed.

In [4]: Tof2d.Tof2d(dat,8)
Button was pushed.
Out[4]: <Tof2d.Tof2d object at 0x2b491044ef50>

In [5]: ec = dat(8).Put(49)+dat(8).Put(50)+dat(8).Put(51)+dat(8).Put(52)

In [6]: p1 = mp.MPlot(ec)

In [7]: ps = PS.PutSum(dat)

In [8]: ec_s = ps.PutSumEC(8)

In [9]: p1.NewPlot(ec_s)

In [10]: Tof2d.Tof2d(dat,8)
Button was pushed.
```

Figure 2: Screen shot of graphical user interface of the chopper- spectrometer in MLF.

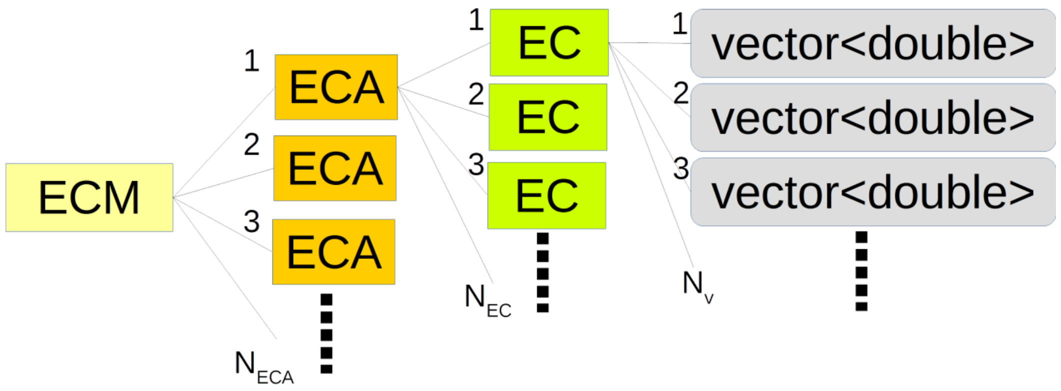


Figure 3: Structure of the EC family defined in ML. The number of vectors, EC and ECA in an EC, ECA and ECM are N_v, N_{EC} and N_{ECA}, respectively.

2. Data File Format

The size of data files handled on ML is increasing with increasing the intensity of proton beam producing pulsed neutron beam in MLF. At now we should concern how to increase the data input/output (I/O) rate on the framework. Histogram data handled on ML can be stored in two data format, (1) a binary format produced by the boost C++ serialization library (serialized format) and (2) the NeXus format [4] by the NeXus-C-API. The points in the issue are: (1) Data files in the serialized format cannot read on the other operating systems. For example, histogram data are stored in MLF with a Linux system, they cannot be read in their home laboratory with a Windows system. Because the data I/O rate is very efficient, temporary files during data analysis on ML are usually written in the serialized format. (2) Data files written in NeXus format can be read by the other operating systems, but the I/O rate for NeXus files on ML is smaller than that for the serialized files. NeXus file format is based on HDF file format [5], and defined and supported by the NeXus international advisory committee. Data files in NeXus format are produced in many neutron, muon and X-ray facilities. If data are stored in NeXus format, they can be read and shared by scientists in the world. We decided that standard data file format in MLF is NeXus, and we should resolve the issue of file I/O rate.

A new and simple data file structure was defined with keeping NeXus format and a new interface between the EC family and NeXus file was developed by using the HDF5 C-API library [4] on ML. The data structure is improved and simplified from the present structure, where the number of groups and the depth of the hierarchical structure in the file structure are decreased.

3. Result and Discussion

In the first half of 2016 we finished prototyping of the new interface on ML. The rates of reading and writing ECMs from/to NeXus files with the present and new interfaces were obtained and shown in Table 1. The file size on the disk are 4.0 and 1.3G bytes at (N_v,N_{EC},N_{ECA})=(3,100,300) and (3,10000,1), respectively. The structures and dimensions of the former and latter data sets are the same as those of three-dimensional histogram data produced by the chopper- spectrometer and the powder diffractometer in MLF. Each EC holds a one dimensional histogram data obtained by a pixel of two dimensional detector. One dimensional histogram can be described by a set of three vector objects: histogram bin, intensity and their error values. In this case the size of each vector stored in an EC is 4000 and the dimension of two dimensional detector are described by N_{EC} and N_{ECA}. The measurements were performed on an Ubuntu Linux (64bit) system with an Intel Core i5-4460 3.2GHz processor and 16 GB memory. The version of HDF5 library is 1.8.16. The I/O rates with the new interface are about 1.6 and 4 times larger than those with the present interface.

N_v	N_{EC}	N_{ECA}	Tw	Tr
3	100	300	24.7(37.9)	8.33(33.2)
3	10000	1	10.39(16.9)	2.76(11.46)

Table 1: Comparison between new and present interfaces in the efficiency of reading and writing NeXus files from/to ECMs. Tw and Tr are the writing and reading times in seconds with the interface developed in this study, and the numbers in the parentheses are those with the present interface. Nv, N_{EC} and N_{ECA} indicate the dimensions of ECM.

The I/O rate on ML can be increased by using the new interface, and we will develop and integrate it into ML. We also decided that the present interface should be kept in ML, because the data files in the file archiver in MLF should be read. Because the new file format developed in this study is in NeXus format, the files produced with the new interface can be shared by the other facilities in the world by using NeXus-API or HDF5 library.

References

[1] Suzuki J, Murakami K, Manabe A, Kawabata S, Otomo T, Furusaka M 2004 Nucl. Instr. and Meth. A 534 175-179

[2] Suzuki J, Nakatani T, Ohhara T, Inamura Y, Yonemura M, Morishima, Aoyagi T, Manabe A and Otomo T 2009 Nucl. Instr. and Meth. A 600 123-125

[3] Nakatani T, Inamura Y, Ito T, Harjo S, Kajimoto R, Arai M, Ohhara T, Nakagawa H, Aoyagi T, Otomo T, Suzuki J, Morishima T, Muto S, Kadono R, Torii S, Yasu Y, Hosoya T, Proceedings of ICALEPCS2009, Kobe, Japan https://accelconf.web.cern.ch/accelconf/icalepcs2009/papers/thc005.pdf

[4] http://www.nexusformat.org/

[5] https://www.hdfgroup.org/

DOI: 10.17199/NOBUGS2016.70

IROHA2: Standard instrument control software framework in MLF, J-PARC

Takeshi Nakatani[1], Yasuhiro Inamura[1], Takayoshi Ito[2] and Kentaro Moriyama[2]

[1] Materials and Life Science Division, J-PARC Center, Japan Atomic Energy Agency (JAEA), 2-4 Shirakata, Tokai-mura, Naka-gun, Ibaraki, Japan
[2] Neutron Science and Technology Center, Comprehensive Research Organization for Science and Society (CROSS), 162-1 Shirakata, Tokai-mura, Naka-gun, Ibaraki, Japan

E-mail: takeshi.nakatani@j-parc.jp

Abstract. We have developed the MLF standard instrument control software framework which is called "IROHA2" by upgrading IROHA. IROHA2 consists of the device control server for operating, monitoring and logging devices, the instrument management server for measurement management, configuration of an instrument and user authentication, the sequence management server for an automatic measurement and the integrate control server for administration of an instrument. Because each software component of IROHA2 has a web interface, we are able to use its all functions under multi-platform environment via a web browser. For the remote access environment in MLF, IROHA2 is also connected to and cooperated with several external systems which are the measurement monitoring system, the authentication system and the information linkage database.

1. Introduction

Neutron and muon experimental instruments in Materials and Life Science Experimental Facility (MLF), Japan Proton Accelerator Research Complex (J-PARC) carry out many kinds of measurements and produce an enormous number of experimental data, such as raw data and metadata representing measurement conditions, by a lot of external users from not only academic but also industry. It is important to be user-friendly and automated control software to perform efficient measurements. In the beginning of MLF, we have developed the MLF standard instrument control software framework for the integration of data acquisition (DAQ) and device control [1, 2]. The name of the software framework is "IROHA". Origin of the name IROHA is the old order of the Japanese alphabet like "ABC" in English. After the usage of IROHA for several years, we have upgraded IROHA to the next generation control software framework which is called "IROHA2" [3]. IROHA2 consists of four core software components, i.e. the device control server to control and monitor each device, the instrument management server to authenticate a user, manage a measurement and configure an instrument setup, the sequence management server to do an automatic measurement and the integrate control server to unify instrument control and monitoring. We can use all the functions of these software components with its web interface. We realized the multi-platform control environment via a web browser [4]. Currently, IROHA2 has supported many kinds of devices (over 50 devices including subtle difference), for example, a beam narrower, a chopper, a goniometer and a temperature controller and has been installed and operated in several neutron and muon experiment instruments in MLF.

2. Software components

The common architecture and implementation of IROHA2 were written in the previous paper [3]. IROHA2 can be run on a relatively low performance computer shown in table 1.

Table 1. Required specifications for IROHA2.

Item	Details
CPU	Xeon 1.86GHz
Memory	8GByte
Hard disk drive	2TB (with log file)
Operating system	Linux (64bit)
Runtime environment	Python 2.6

There are four core software components in IROHA2 shown in figure 1. The functions of the device control server and the instrument management server were mentioned in the previous paper [4]. The functions of the sequence management server and the integrate control server are as follows. The data linkage between the instrument management server and the MLF external systems will be mention in the next section.

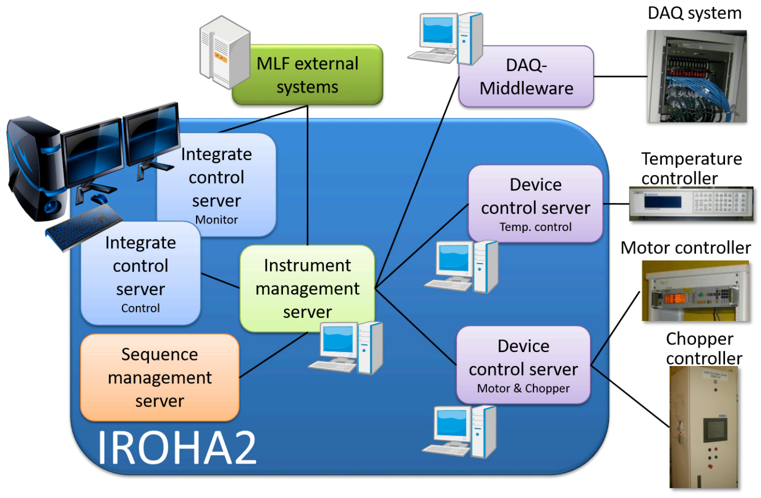

MLF external systems
DAQ-Middleware
DAQ system
Integrate control server Monitor
Integrate control server Control
Instrument management server
Sequence management server
Device control server Temp. control
Device control server Motor & Chopper
Temperature controller
Motor controller
Chopper controller
IROHA2

Figure 1. IROHA2 software components (Device control server, Instrument management server, Sequence management server and Integrate control server)

2.1. Sequence management server

The sequence management server has many kinds of functions which is necessary in an automatic measurement. We can edit a sequence for an automatic measurement with a drag and drop operation, graphically confirm a sequence by visualization of changing parameters, run control (Begin / End / Pause / Resume) a sequence and monitor progress of a sequence on a web user interface. In addition, if a new device is attached to an instrument, we can generate operating commands of the device by obtaining the parameter list from its device control server via a web browser. The sequence is available for nested loop structure by numerical increasing parameters as well as list data in Python. Because each command of the sequence can return values of the result, we can do feedback measurement. Because the sequence is running in background process by the Python Debugger, we can change the parameters of the sequence dynamically.

2.2. Integrate control server

The role of the integrate control server is integration of an instrument. We can run control not only a DAQ system but also the sequence management server, operate the devices and monitor all of the information which is necessary in a measurement, for example, the DAQ status, the device status of the present values and the trend graphs, the proton beam power and the facility status. Because the web user interface of the server is able to be customized flexibly, the instrument scientists can perform access control (visible / invisible and operational / not operational) by each parameter of the devices in their instrument.

We can select the operation mode when the integrate control server is started. The modes are two types, which are “Control mode” and “Monitor mode”. The “Control mode” is that the server is run as mentioned above. The “Monitor mode” is that the server collects the status and the parameters from the other servers and produces the static web page files at stated periods. The files are transferred to the measurement monitoring system which is accessible from Internet. The users can monitor the present measurement status with a web browser on their own Internet connection device after the authentication.

3. Data linkage

One of the main functions of the instrument management server is producing run information which is recorded each measurement result. The run information is included the location of the raw data, the user, the theme, the sample, the devices with their parameters and the measurement time, which are typically written in an experimental logbook. In MLF, the location of the raw data is configured by the configuration file of the DAQ middleware, the user is authenticated by the login process to IROHA2, the information of the theme and the sample is obtained from the users’ manual inputs or the user proposal and support system in J-PARC Users’ Office, the information of the devices is collected from the device control servers and the time is recognized from the system clock adjusted by NTP. The data flow between the server and the external systems is shown in figure 2. When the user logins to IROHA2, the instrument management server connects the authentication system by LDAPS, load the user data and certificates the user. To obtain the information of the theme and the sample, the server connects the information linkage database in cooperation with the user proposal and support system and load the data of the theme and the sample by the Oracle Instant Client [5]. After the run information recorded by the server is archived and cataloged in the MLF experimental database [6], the users can access their run information via a web browser from inside of the facility as well as their home laboratory.

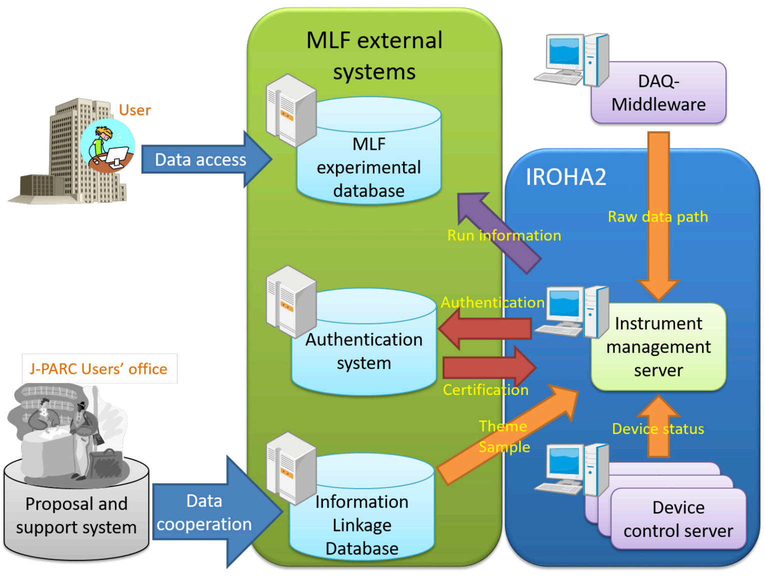


Figure 2. Data flow diagram between IROHA2 instrument management server and the MLF external systems.

4. Conclusion

We have upgrade and operated the next generation MLF standard instrument control software framework called "IROHA2". Because IROHA2 can automatically produce the information written in an experimental logbook, we will realize that the users in MLF can do "logbook-less" experiments.

Acknowledgments

We acknowledge the valuable discussions with and suggestions of the computing environment group in MLF.

References

[1] Nakatani T, Inamura Y, Ito T, Harjo S, Kajimoto R, Arai M, Ohhara T, Nakagawa H, Aoyagi T, Otomo T, Suzuki J, Morishima T, Muto S, Kadono R, Torii S, Yasu Y, Hosoya T and Yonemura M 2009 *Proc. of ICALEPCS2009* 673-675.

[2] Nakatani T, Inamura Y, Ito T, Moriyama K and Otomo T 2014 *JPS Conf. Proc*. 1, 014010.

[3] Nakatani t, Inamura Y, Ito T and Otomo T 2015 *Proc. of ICANS-XXI (JAEA-Conf 2015-002)* 493-496.

[4] Nakatani T, Inamura Y, Ito T and Otomo T 2015 *JPS Conf. Proc.* 8, 036013.

[5] http://www.oracle.com/technetwork/database/features/instant-client/index.html

[6] Moriyama K and Nakatani T 2015 *Proc. of ICALEPCS2015* 834-837

DOI: 10.17199/NOBUGS2016.23

Recent Progress in the Development of MLF EXP-DB in J-PARC

Kentaro Moriyama[1] and Takeshi Nakatani[2]
[1] Neutron Science and Technology Center, Comprehensive Research Organization for Science and Society (CROSS), 162-1 Shirakata, Tokai-mura, Naka-gun, Ibaraki, Japan
[2] Materials and Life Science Division, J-PARC Center, Japan Atomic Energy Agency (JAEA), 2-4 Shirakata, Tokai-mura, Naka-gun, Ibaraki, Japan

E-mail: k_moriyama@cross.or.jp

Abstract. The MLF Experimental Database (MLF EXP-DB) is a core system aimed at delivering advanced services for data management and data access in J-PARC/MLF. Its main role is to safely and efficiently manage a huge amount of data created in neutron instruments and provide effective data access to facility users. It is a web-based integrated database system based on a three-layer architecture model. By collecting experimental data and associated information, e.g., proposal, principal investigator, and samples from instruments and other business database systems, it creates an experimental data catalog and enables facility staff and users to access this catalog by providing them with web portals. Recently, we redesigned the MLF EXP-DB to enhance its availability and scalability toward achieving a full-scale operation in the future. In addition, we developed web portals to improve its usability. This paper presents the details of the recent developments and the current operating status of MLF EXP-DB.

1. Introduction

A huge amount of experimental data can be produced in neutron scattering experiments using current high-intensity beams; therefore, data management is a crucial factor in the research workflow. Data should be managed efficiently for a long duration and delivered quickly to users in order to promote the acquisition of scientific results. Metadata is useful for the utilization and management of such data. Thus, we are developing the MLF Experimental Database (EXP-DB) as a core system in the data management infrastructure of J-PARC/MLF [1], which creates experimental metadata by collecting experimental data from instruments and provides data management services and access to facility staff and users via web portals. This system is currently under limited operation for data management on a trial basis with several instruments in J-PARC/MLF.

One of the features of MLF EXP-DB is an XML database system. In MLF, the measurement condition is recorded as a measurement log in XML format by IROHA, which is a standard measurement control software framework in MLF [2], and this log can have different XML structures relative to the given measurement conditions. As the XML database has flexibility and extensibility for the data structure, this architecture is suitable for handling our measurement logs. However, the XML data size can easily become large owing to structural information such as tags and attributes, and its analytical process is generally heavy. Therefore, implementing features to effectively and stably store, search, and manage a large amount of XML data is challenging. In addition, facility users must have quick and effective data access. Thus, the issues that need to be resolved to achieve a full-scale operation of MLF EXP-DB in the future are as follows:

- High reliability is required for a core system; however, the conventional MLF EXP-DB runs on a single physical server. By removing this single point of failure, service outages due to system failures and maintenance should be avoided as much as possible.
- Scalability for data collection is required. The load for data collection depends on the data production rate, which is increasing continuously with improved instrument performance and neutron beam intensity. The beam intensity is scheduled to increase according to the development plan for accelerators in the facility.
- A web portal that enables effective and quick data access should be provided to facility users. A data search function is especially important to find required data for analysis from a large amount of experimental data.

To address these issues, we redesigned MLF EXP-DB and developed the web portal as follows:

- **High availability:** We enhanced the reliability of MLF EXP-DB by improving it as a redundant distributed system.
- **Scalability:** We redesigned the MLF EXP-DB to a scaled-out configuration. It is possible to scale the system performance based on the data rate.
- **Flexible data search:** We improved the web portal for data access by implementing a flexible data search function.

2. MLF EXP-DB

MLF EXP-DB is a web-integrated database system that employs Java-based commercial middleware with an XML database management system named “R&D Chain Management (RCM) System Software” [3], which is based on a three-layer architecture model comprising three primary components: RCM-Web, RCM-Controller, and RCM-DB. RCM-Web provides the system’s web interface, and the RCM-Controller processes requests from RCM-Web and queries for RCM-DB. RCM-Controller has a workflow engine that executes the workflow described in the XML format. All MLF EXP-DB functions are implemented with the workflow. RCM-DB is a hybrid XML database system, i.e., the XML database is implemented as a relational database and the XML tree is registered in relational tables. After collecting experimental data from instruments, RCM-Controller creates metadata and registers them in RCM-DB as an experimental data catalog. Facility users and staff can access this catalog via the RCM-Web web portal. To reduce communication overhead with a large volume of XML data among these components, all components run on a single physical server.

3. Recent developments

3.1. High availability

To improve the reliability of MLF EXP-DB, we redesigned the system to remove the single point of failure. The system, which previously ran on a single physical server, was upgraded to a redundant distributed system comprising two physical servers with the same specifications in a switch-over relationship. This system configuration realizes high availability, especially for data collection from instruments. In addition, for MLF EXP-DB, we deployed a redundant front-end web server, which is used for data management by facility staff. Figure 1 shows a schematic of the current system, and Table 1 shows the MLF EXP-DB server specifications.

Both redundant systems adopt an active-standby deployment model. In MLF EXP-DB, the active server collects experimental data and provides a web portal for data access through a front-end web server. Real-time replication among the databases of the active and standby servers is also performed. When a system failure or maintenance occurs, executing a switching script migrates the operation from the active server to the standby server. It is possible to complete operational migration within 30 min. Conversely, automatic failover of the front-end web system is realized using a cluster heartbeat. The active and standby servers share a virtual IP address; thus, facility staff can access the web portal easily.

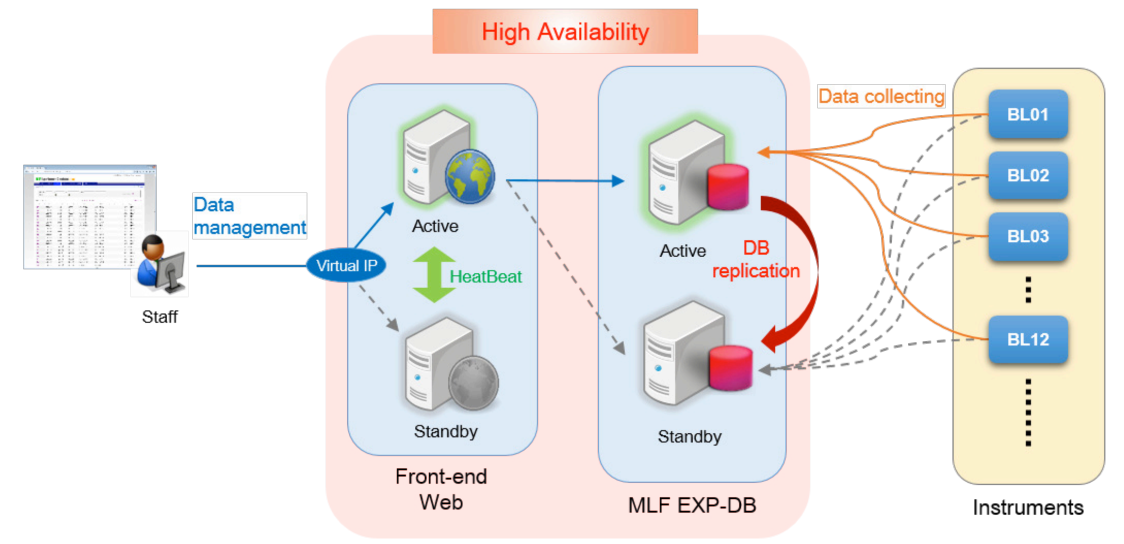


Figure 1. Operation of the redundant distributed system

Table 1. Server specifications

Item	Specification
OS	Red Hat Enterprise Linux 6.3 64 bit
Processor	Intel Xeon E5-2407 v2, 2.4 GHz, 4 core × 2
Memory	128 GB
HDD	SAS 7,200 rpm, 1 TB × 3 RAID 5
Network	1 Gb Ethernet × 4 ports
Power Supply	Dual, hot-plug redundant power supply (350 W)

3.2. Scaling-out

Although XML has high flexibility and extensibility in terms of data structure, the amount of data is generally becoming large, and the loads for creation and analysis of XML data are becoming heavy; this results in significant performance degradation. To improve the performance of MLF EXP-DB, we scaled out the system; this enables data collection load balancing and partitioning of increasing amounts of XML data. The improved system comprises two nodes, and each node is a redundant system (described in the previous subsection). This architecture allows us to improve performance by adding a node in response to the data rate. Data access transparency in the distributed system is provided by Node 1 acting as a master server that receives all requests from a user and performs cross-searching over nodes.

Figure 2 shows a schematic of the system. In MLF, 20 instruments are currently under operation, and these instruments are deployed in two experimental halls in the facility. Two nodes in the scaled-out system, i.e., Node 1 and Node 2, are allocated to instruments in the first and second experimental halls, respectively. Users can access data via the web portal using the front-end web server.

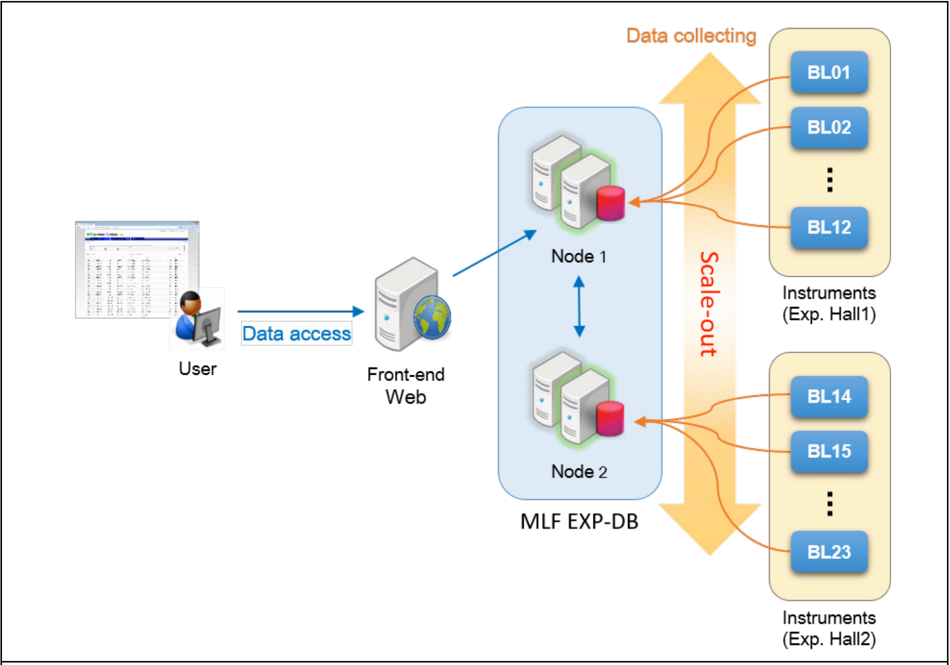


Figure 2. Scaled-out system for data collection and unified web interface for the distributed system

We made a preliminary evaluation of the data collection performance for the scaled-out system. Figure 3 shows the data collection throughput, which includes the process of creation and registration of metadata after collection of experimental data. In this case, the number of simultaneously collected datasets is 200, and each dataset includes raw measurement data and log files. The size of the raw data is approximately 500 MB, and the size of log files is less than 1 MB. The number of tags in the XML format log file is 754. These are typical data sizes in MLF. However, the number of tags changes in response to the number and type of devices used. The throughput gradually decreases with the number of processes; however, it is possible to improve performance by adding a node.

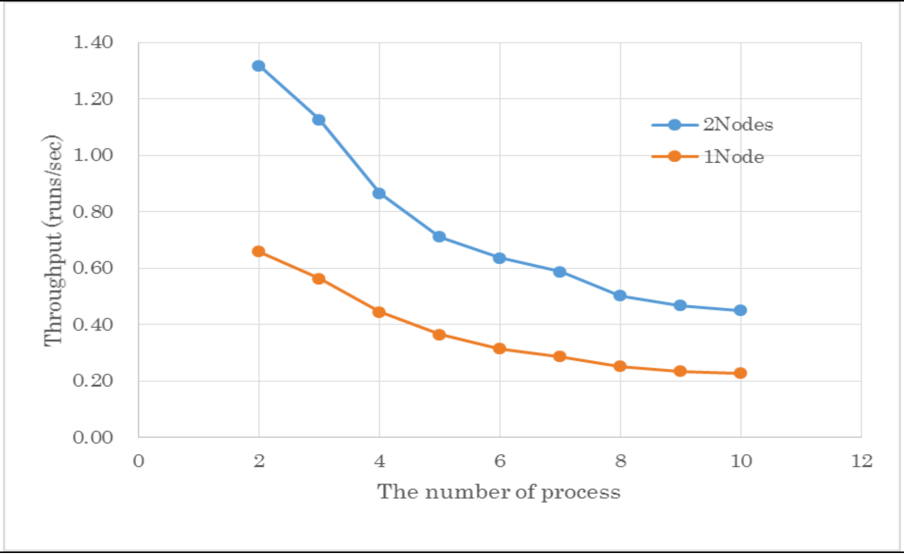


Figure 3. Data collection throughput in the scaled-out system

3.3. Flexible data search in web portal

A quick and easy search for the data required for analysis is important. Therefore, we implemented a flexible data search function in the web portal that enables search of various experimental metadata.

- **Experimental Proposal:** Proposal ID, Title, Primary investigator, Period
- **Measurement:** Experimental ID, Run No., Comment
- **Sample:** Sample ID, Substance name, Chemical formula
- **Device condition:** Device type, Setting parameter, Physical value

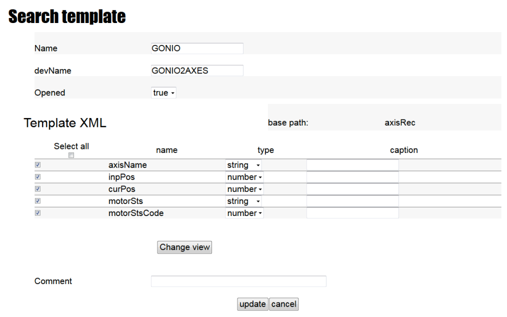
A search with the device condition is useful for data generated under a sequence of measurements by changing sample environmental conditions such as temperature and magnetic field. However, a search for metadata with a flexible XML structure in the XML database can significantly degrade performance and provide insufficient results. Furthermore, the user must be able to search for data without considering a complex XML structure.

Thus, we developed an effective search procedure using a search template, which is an XML subset with a different structure relative to the device type. Figure 4 shows an example measurement log, search template, and function. The search template includes the device name, axis name, and type of physical value, such as a string or a numerical value. When a new device is introduced, the instrument staff must create a template. The user can execute a search by selecting devices whose templates are registered for each instrument.

a)

```
<logs>
  <params>
    <comment/>
    <intervalStanby>10</intervalStanby>
    <intervalMeas>1</intervalMeas>
    <axisInfo>
      <axisRec>
        <axisName>phi</axisName>
        <inpPos>50.000</inpPos>
        <curPos>0.053</curPos>
        <motorSts>Stop</motorSts>
        <motorStsCode>1</motorStsCode>
      </axisRec>
      <axisRec>
        <axisName>ome</axisName>
        <inpPos>50.000</inpPos>
        <curPos>0.249</curPos>
        <motorSts>Stop</motorSts>
        <motorStsCode>1</motorStsCode>
      </axisRec>
    </axisInfo>
  </params>
  <devName>GONIO2AXES</devName>
</logs>
```

b)



Search template

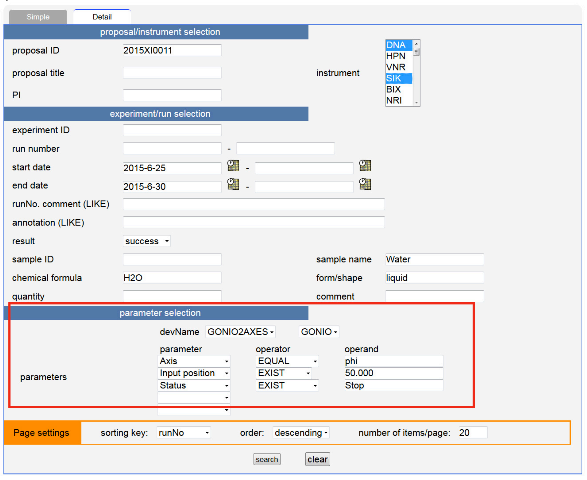
Name	GONIO
devName	GONIO2AXES
Opened	true

Template XML — base path: axisRec

Select all	name	type	caption
✓	axisName	string	
✓	inpPos	number	
✓	curPos	number	
✓	motorSts	string	
✓	motorStsCode	number	

Comment:

c)



proposal/instrument selection

proposal ID	2015XI0011
proposal title	
PI	
instrument	[✓ DNA], HPN, VNR, [✓ SIK], BIX, NRI

experiment/run selection

experiment ID			
run number	-		
start date	2015-6-25 -		
end date	2015-6-30 -		
runNo. comment (LIKE)			
annotation (LIKE)			
result	success		
sample ID		sample name	Water
chemical formula	H2O	form/shape	liquid
quantity		comment	

parameter selection

parameters — devName: GONIO2AXES, GONIO

parameter	operator	operand
Axis	EQUAL	phi
Input position	EXIST	50.000
Status	EXIST	Stop

Page settings — sorting key: runNo; order: descending; number of items/page: 20

Figure 4. Flexible data search with device condition: a) device condition in the measurement log, b) screenshot of search template management, and c) screenshot of flexible data search

4. Current status and future plan

Currently, data management using MLF EXP-DB is being performed on a trial basis with some instruments, i.e., BL02, BL11, BL17, and BL18 in the MLF. In this trial operation, the data produced in previous experiments are primarily collected to accumulate metadata and verify the required information in the experimental data catalog. We plan to start a full-scale operation of MLF EXP-DB soon, which will enable real-time data collection in synchronization with experiments to provide quick data access and extend the management scope to other instruments in the MLF.

For data access, the basic development of the web portal has been completed. We will begin a trial of the web portal and remote access from outside the facility with a limited number of users. However, some issues regarding security need to be addressed before a full-scale operation can begin. By addressing these issues, we will start a full-scale operation for data access later this year.

5. Conclusion

We successfully improved MLF EXP-DB and enhanced its availability and scalability toward achieving a full-scale operation of the system and facility in the future. The current system is a scaled-out, redundant distributed configuration. In addition, we developed a flexible data search function in a web portal. A user can easily and quickly search for data required for analysis from a large amount of data using this function.

References

[1] Moriyama K and Nakatani T 2015 *Proc. of ICALEPCS2015* 834-837.
[2] Nakatani T et al., 2009 *Proc. of ICALEPCS2009* 676-678.
[3] http://www.i4s.co.jp/rcm/rcmabs.html (in Japanese).

DOI: 10.17199/NOBUGS2016.14

Safety systems at the SAFARI-1 neutron diffraction facility

J.C. Raaths, P.R. van Heerden, D. Marais, A.M. Venter

Research and Development Division, Necsa SOC Limited, PO Box 582, Pretoria, 0001, South Africa

E-mail: deon.marais@necsa.co.za

Abstract. Key safety features of the neutron diffraction facility located at the SAFARI-1 research reactor of the South African Nuclear Energy Corporation (Necsa) SOC Limited are described. This includes the interlock software systems integrated with the primary and secondary beam shutters, the verification of inadvertent entry to radiation risk areas, as well as controls of instrument movements.

1. Introduction

Personnel safety at beam line facilities at nuclear research installations are of primary importance, in conjunction with the establishment of a safe working environment and minimizing of the risk to exposure to ionizing radiation [1]. This can is achieved through robust engineering design of independent safety systems and work-safe procedures.

2. The SAFARI-1 neutron diffraction facility

The SAFARI-1 neutron diffraction facility (NDIFF) presented in Figure 1, accommodates two instruments on one reactor beam line. These instruments are the materials probe for internal strain investigations (MPISI) [2] and the powder instrument for transition in structure investigations (PITSI) [3]. All features along the primary radiation line are shared which include the in-pile collimator with its beam shaper, filtering system and the primary beam shutter (PS). From the monochromator stages on each instrument functions independently with its own instrument control system and secondary beam shutter (SS). The latter comprises two linked sections comprising respectively an *internal* part located within the monochromator chamber, and an *external* part attached to the chamber wall.

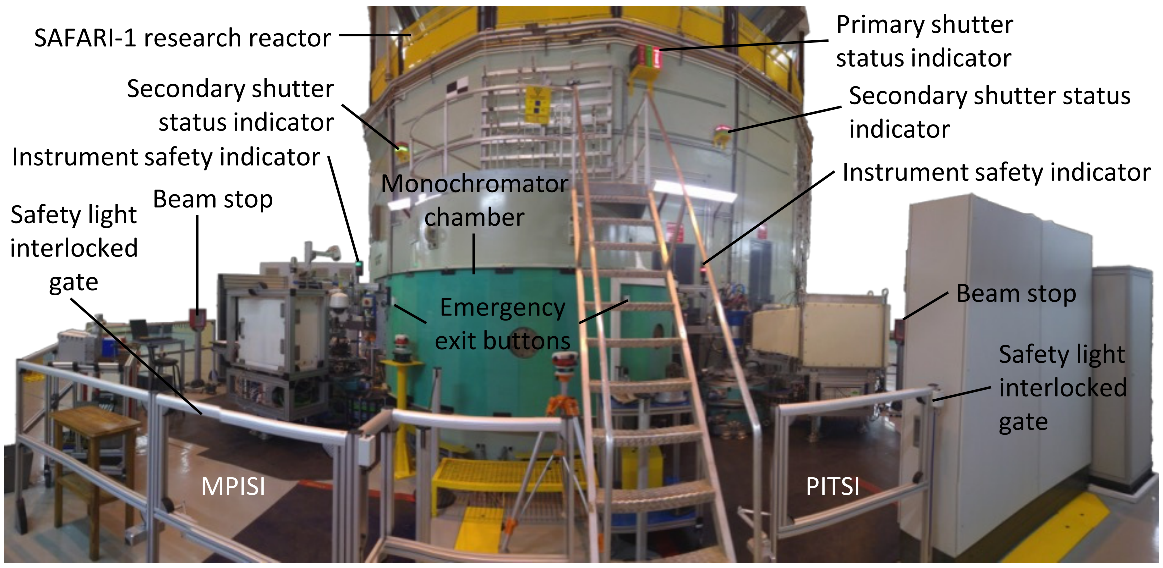

SAFARI-1 research reactor
Secondary shutter status indicator
Instrument safety indicator
Safety light interlocked gate
Beam stop
Monochromator chamber
Emergency exit buttons
Primary shutter status indicator
Secondary shutter status indicator
Instrument safety indicator
Beam stop
Safety light interlocked gate
MPISI
PITSI

Figure 1. Photograph of the NDIFF with a number of safety systems and indicators identified.

3. Beam shutters

3.1. Shutter description

A functional safety shutter system is the primary protection against accidental exposure to ionizing radiation at beam line facilities. The material configuration of the NDIFF shutters were optimized to ensure that virtually all radiation is attenuated when the shutters are in the closed positions. The PS and internal part of the SS is a layered structure of borated polyethylene, borated aluminium and low carbon mild steel. The external part of the SS contains a 1 mm layer of cadmium and a 50 mm layer of lead.

The PS is located in the core box cavity where the biological reactor wall interfaces with the in-pile collimator. It comprises a large rotatable drum equipped with designated *closed*, *high-intensity* (100 x 60 mm^2 beam) or *high-resolution* (60 x 60 mm^2 beam) positions. It is driven by a chain linked electric motor in conjunction with a resolver that serves as status verification. The drum and chain are visible in Figure 2. The *internal* (Figure 2) and *external* (Figure 3) parts of the SS are operated through a linked pneumatic system. In the event of a power failure, air pressure to the pneumatic pistons is released and the shutters move into their closed positions under gravity, rendering this a passive safety system.

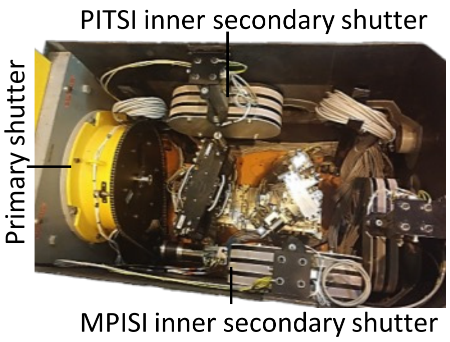


Figure 2. Photograph of the NDIFF monochromator chamber showing the *primary* and *internal* secondary shutters.

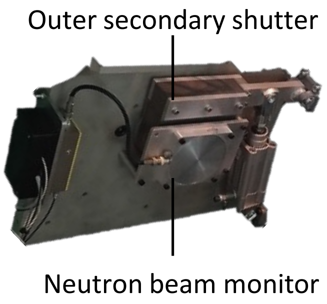


Figure 3. Photograph of an *external* secondary shutter and neutron beam monitor.

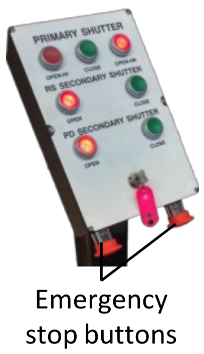


Figure 4. Photograph of the shutter control console.

3.2. Shutter control

All shutters are operated from a shared console (shown in Figure 4) located on a pedestal in the NDIFF computer room. The top row of buttons pertains to the PS, the middle to the SS of MPISI and the bottom to the SS of PITSI. Shutters can only be opened if the lockout key is in place. This minimizes the risk that the PS is accidentally opened during maintenance activities inside the chamber.

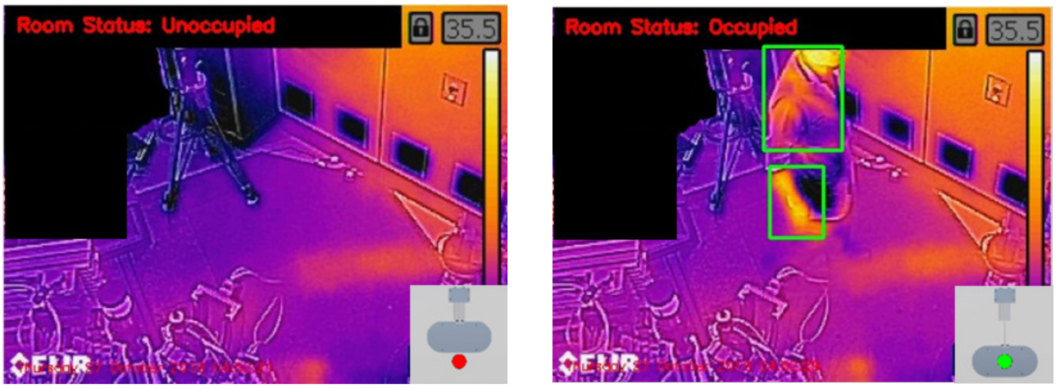
The safety system logic trees and control routines were implemented on a GALIL DMC 2280 motor controller using the GALIL command language [4]. In addition to using the shutter control console, the SS can be closed by executing a command from the instrument control software, enabling the automated closure of the shutter after completion of investigations.

An emergency button located on the motion control cabinet door is within reach from the instrument area. All motor motions are stopped and the SS is closed when this button is depressed.

3.3. Continuous monitoring system prototype

Amongst others, beam line facilities enforce *search procedure* that necessitates that strategically placed *search buttons* to verify that no person accidentally remains within the instrument area when the beam is turned on. At NDIFF, a continuous monitoring approach is employed that utilises a video camera to detect undue movement, or the use of thermal imaging, of personnel within the instrument area. In the event that a person is detected, the SS cannot be opened. In addition, upon unlawful entry into the experimental area, the SS will be closed automatically. This eliminates the need for a search procedure.

The implementation has been performed using a USB camera and a low-cost Raspberry Pi 3 microcomputer with the motion detection aspects done with Python scripts [5] utilizing the OpenCV library [6]. Functionality is also provided to mask out areas on the video frame which should be excluded the monitoring process. To reduce false positives, a thermal USB camera which is insensitive to lightning level changes can be used. Figure 5 shows selected video frames where a person entering the MPISI instrument area is detected. A flow chart of the implementation of the continuous monitoring system is given in Figure 6.



(a) (b)

Figure 5. Video frames retrieved from the continuous monitoring system showing: (a) the unoccupied instrument area; (b) the detection of a person in the area. The secondary shutter status is indicated on the bottom right section of each photograph.

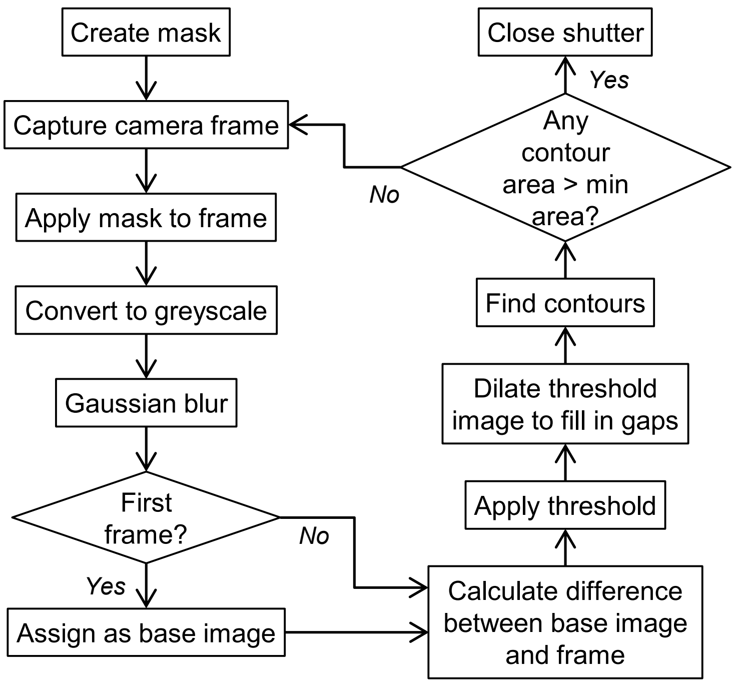


Figure 6. Flow chart of the implementation of the continuous monitoring system at the NDIFF instruments.

4. Visual displays

4.1. Facility and instrument shutter indicators

On all indicator lights, green signifies a safe condition and red an at-risk condition. The PS status indicator (Figure 7) is prominently mounted to the reactor wall and is visible from the SAFARI-1 and NDIFF control rooms. SS indicators (Figure 8) are mounted against the reactor wall above each instrument and are visible from the NDIFF computer room and the respective instrument beam stop locations. These indicators rely solely on the shutter positions and are switched through the GALIL controller.

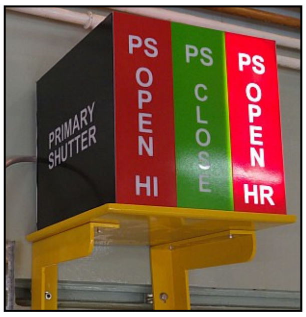


Figure 7. Photograph of the primary shutter status indicator.

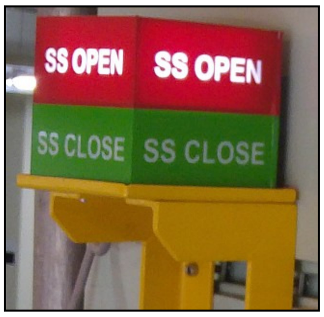


Figure 8. Photograph of a secondary shutter status indicator.

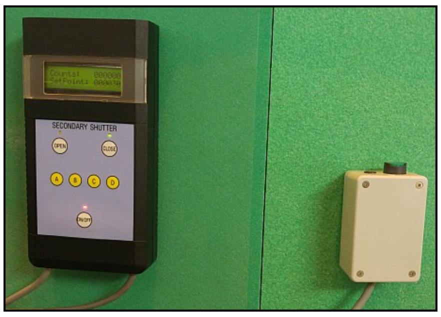


Figure 9. Interlock box and emergency exit button.

4.2. Instrument safety indicators

Each instrument also has a beam status indicator mounted at eye level at the respective beam exit ports as shown in Figure 10. The indicator status is controlled by an interlock box (Figure 9) containing a C programmed dsPIC30F6012 microcontroller using the neutron beam monitor counts directly and will only switch the safety indicator to a safe condition if the neutron count rate is below a pre-determined threshold and the SS is in the closed position.

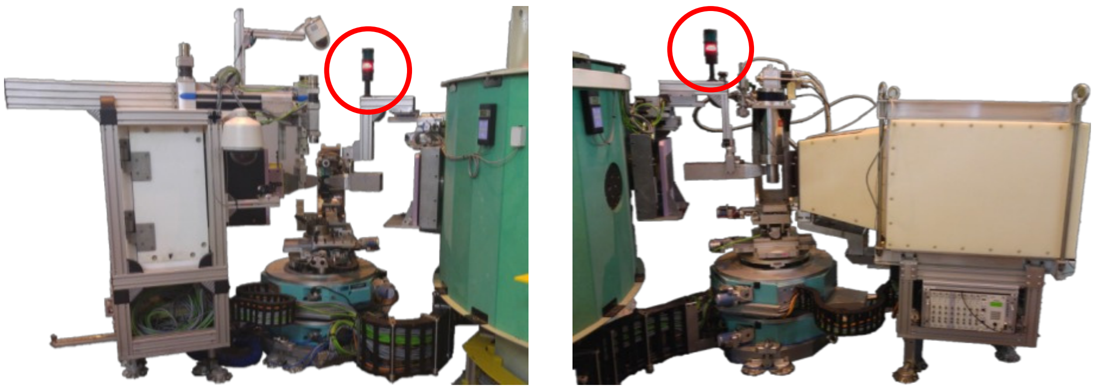


Figure 10. Photographs of (a) MPISI and (b) PITSI, highlighting the safety indicators.

5. Gate interlock system

To restrict unauthorized access to the instrument areas during operation, each instrument is enclosed with a fence and interlocked gate. The gate is controlled by the interlock boxes and magnetically locked when the SS are opened and the neutron count subsequently exceeds a threshold value.

In the inadvertent situation that a person enters the monitor area, the gate can be unlocked using an emergency exit button located on the chamber wall. This unlocks the gate for 5 seconds after which the lock is reactivated. Maintenance entry is possible by depressing a button located on the gate for 25 seconds that unlocks the gate for 5 seconds where after the lock is reactivated.

6. Motion interruption

6.1. Motor limit switches

Numerous motorized motion stages exist on both instruments to enable the positioning of the detectors at a specific diffraction angle, as well as the translation of beam apertures and the sample on the goniometer table. All motions pose a risk of damaging the instrument during collisions, or cause injury to inattentive users. Limit switches, which stops all motion once activated, constrain movements within the envisaged work envelope. In addition, beam apertures are mounted on knuckles equipped with contact breakers which stop all motions in the event that the aperture is accidentally crashed into during sample translations.

6.2. Emergency stop buttons

Each instrument is equipped with an emergency motion stop button that is attached to the sample table. This intervention proved to be convenient for sample setup since it enables suspension of motion when a desired sample position has been reached. Emergency stop buttons are also present on the shutter control console inside the NDIFF computer room as well as on the instrument control software graphical user interface, Gumtree [7].

7. Beam stops

Due to the open workspace in the reactor hall, it is of utmost importance to ensure that the neutron beam is contained to the instrument area. Since the primary beam is well collimated and the beam size restricted, a relatively small beam stop shown in Figure 11 is adequate to ensure capture of the transmitted beam. The beam stop is constructed from a 25 mm thick borated polyethylene (5wt% boron) sheet sandwiched between two 5 mm thick sheets of 95% ^{10}B enriched borated aluminium 1100 Alloy (4.5wt% boron) on the beam side and a 16 mm mild steel sheet on the back side.

The effectiveness of the beam stops are summarized in Table 1.

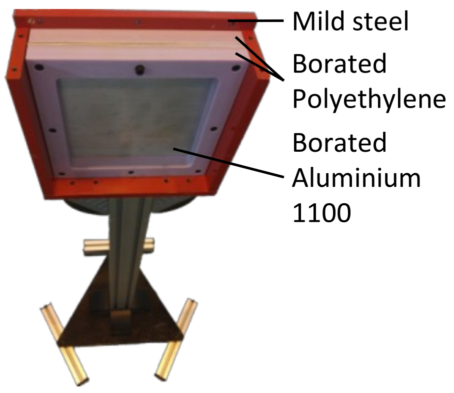


Figure 11. Photograph of an NDIFF instrument beam stop

Table 1. Effective neutron and gamma dose rates in the beam path measured when scattering from the Si (331) monochromator plane

Position	Dose rate [μSv/h]	
	Neutrons	Gammas
Beam port	*470*	*318*
Front of beam stop	*333*	*277*
Behind beam stop	*2.3*	*10.9*

8. Conclusion

Every effort has been made to ensure that the neutron diffraction facility adheres to best practices to ensure the safety of personnel. Reliable software at neutron facilities is not only important for data acquisition, control, analysis and visualisation, but also for safety systems.

References

[1] South Africa 2006 National nuclear regulator act (Act no 47 of 1999): on safety standards and regulatory practices (Government notice no. R388) *Government gazette* 28755, 28 April

[2] Necsa SOC Limited, MPISI - Materials Probe for Internal Strain Investigations, http://www.necsa.co.za/Products-and-Services/Research/Facility-and-Instrument-characteristics/MPISI, Date of access: 27 September 2016

[3] Necsa SOC Limited, PITSI - Powder Instrument for Transition in Structure Investigations, http://www.necsa.co.za/Products-and-Services/Research/Facility-and-Instrument-characteristics/PITSI, Date of access: 27 September 2016

[4] Galil Motion Control Inc. 2011 DMC-2x00 User Manual Rev 2.1 California

[5] Rosebrock A 2015 Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox, http://www.pyimagesearch.com/2015/06/01/home-surveillance-and-motion-detection-with-the-raspberry-pi-python-and-opencv/, Date of access: 27 September 2016

[6] Itseez Open Source Computer Vision Library, https://github.com/opencv/opencv, Date of access: 27 September 2016

[7] Hugh Rayner H, Hathaway P, Hauser N, Fei Y, Franceschini F and Lam T 2006 *Physica* B **385-386** 1333-1335