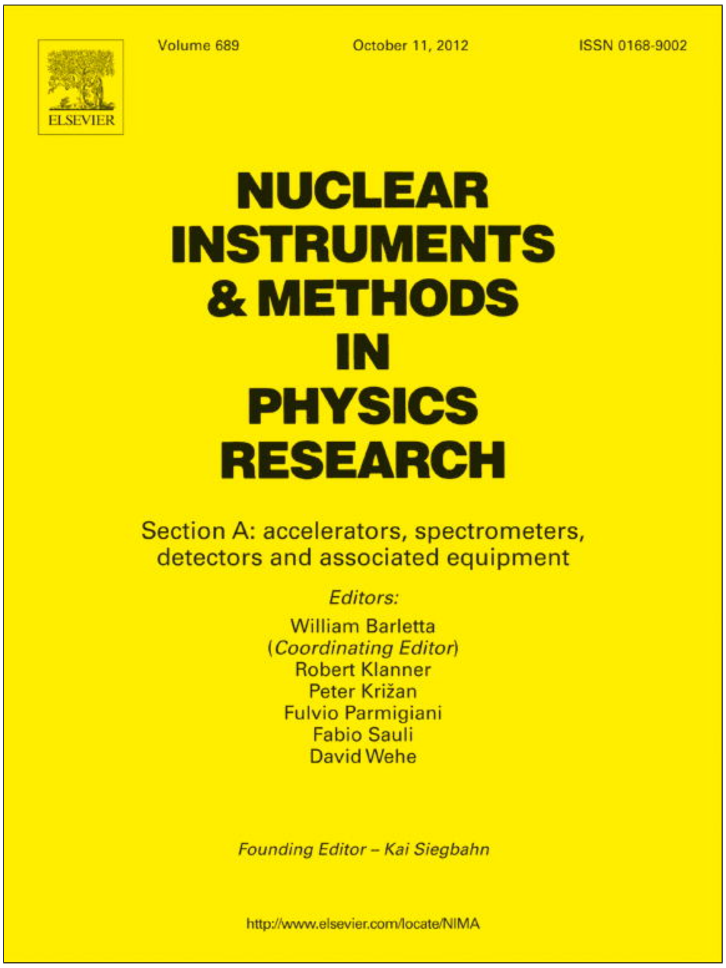


Volume 689 October 11, 2012 ISSN 0168-9002

ELSEVIER

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH

Section A: accelerators, spectrometers, detectors and associated equipment

Editors:

William Barletta
(*Coordinating Editor*)
Robert Klanner
Peter Križan
Fulvio Parmigiani
Fabio Sauli
David Wehe

Founding Editor – Kai Siegbahn

http://www.elsevier.com/locate/NIMA

Nuclear Instruments and Methods in Physics Research A 689 (2012) 52–59

Contents lists available at SciVerse ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

Automated acquisition and analysis of small angle X-ray scattering data

Daniel Franke*, Alexey G. Kikhney[1], Dmitri I. Svergun

European Molecular Biology Laboratory, Hamburg Outstation Notkestrasse 85, D22603 Hamburg, Germany

ARTICLE INFO

Article history:
Received 2 April 2012
Accepted 8 June 2012
Available online 19 June 2012

Keywords:
SAXS
Automation
Workflow
Pipeline

ABSTRACT

Small Angle X-ray Scattering (SAXS) is a powerful tool in the study of biological macromolecules providing information about the shape, conformation, assembly and folding states in solution. Recent advances in robotic fluid handling make it possible to perform automated high throughput experiments including fast screening of solution conditions, measurement of structural responses to ligand binding, changes in temperature or chemical modifications. Here, an approach to full automation of SAXS data acquisition and data analysis is presented, which advances automated experiments to the level of a routine tool suitable for large scale structural studies. The approach links automated sample loading, primary data reduction and further processing, facilitating queuing of multiple samples for subsequent measurement and analysis and providing means of remote experiment control. The system was implemented and comprehensively tested in user operation at the BioSAXS beamlines X33 and P12 of EMBL at the DORIS and PETRA storage rings of DESY, Hamburg, respectively, but is also easily applicable to other SAXS stations due to its modular design.

1. Introduction

Small Angle X-ray Scattering (SAXS) allows the study of the structure and interactions of proteins, nucleic acids and their complexes in solution. Macromolecular folding, unfolding, aggregation, shape, conformation, and assembly processes may be studied under a variety of conditions, from near physiological to highly denaturing. The increasing availability of high-flux third-generation synchrotron radiation sources, recent progress in instrumentation as well as novel analysis methods allow for more efficient application of SAXS in structure analysis, especially in combination with other techniques [1].

A SAXS measurement protocol for biological samples in solution requires repeated exposures of the specimen to the X-ray radiation, usually at different concentrations. For each data set, (1) the specimen must be brought to the measurement position in the sample holder, (2) the beam path cleared, (3) the two-dimensional data collected, (4) the beam parameters monitored for later normalization, (5) the beam path blocked again and (6) the specimen cleaned out from the sample holder. Further, the two-dimensional scattering image collected by the detector usually needs to be radially averaged to obtain a one-dimensional data set. Any mistake in this process is detrimental to the overall result. Manual errors, for example introduced due to fatigue after long hours of continuous measurement, should be avoided. This strongly calls for the automation of the entire measurement protocol.

In high-throughput crystallography [2], a crucial step forward in the large-scale analysis of proteins and macromolecular complexes was the introduction of automatic sample changers and remote operation [3]. The first user-oriented sample changer for biological macromolecules in solution was introduced at the X33 beamline at EMBL Hamburg in 2007 [4]. The second generation sample changer, Mark II, was developed in collaboration between the EMBL and ESRF and installed at X33 in 2009. The final version was deployed at the ID14-3 beamline of ESRF and is in user operation at the P12 BioSAXS beamline at PETRA-III [5].

Besides reliable sample loading, automation of a SAXS experiment requires the coordination of a multitude of devices, e.g. valves, motors, digital I/O signals and, of course, the detector and the sample changer itself. Obviously, any integrated communication protocol between these devices depends on the device control system of any given beamline. Recently, Classen et al. [6] described their efforts to adapt the Blu-Ice/DCS control system, initially developed by McPhillips et al. [3] for crystallography beamlines, to SAXS. Here, we present a basically control system independent, flexible and easily extensible approach to SAXS experiments—not only to provide automation, but to allow for remote, unattended and eventually autonomous data acquisition.

* Corresponding author. Tel.: +49 40 89902244.
E-mail addresses: d.franke@embl-hamburg.de, franke@embl-hamburg.de (D. Franke), a.kikhney@embl-hamburg.de (A.G. Kikhney), d.svergun@embl-hamburg.de (D.I. Svergun).

[1] Contributed equally to the work.

0168-9002/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.nima.2012.06.008

2. Automation of data acquisition

2.1. Device communication

The most important part of any automation system is a reliable communication between the associated devices. To this means, we employed the three-fold integrated networking environment, TINE [7]. TINE was developed as control system for the synchrotron operation at DESY, Hamburg, Germany. Here, TINE was initially used to control the HERA experiment, currently it operates the DORIS and PETRA storage rings and will also be used to run the biological crystallography and SAXS beamlines at PETRA-III which are being commissioned by the EMBL.

2.2. TINE control system

The TINE control system provides the basic programming interface to implement device servers. Device servers allow any hardware device, from digital I/O panels and motors to a detector, to be addressed, queried and controlled over the network utilizing their exported properties. A property, most often named after a command supported by the device, may either be read, written or both. It is to note that a device server may also itself be a client of other servers.

The device servers implemented are split into a control-system dependent frontends and device specific backends (Fig. 1). The frontend receives the client's queries, verifies the validity of the request and its corresponding arguments and passes the valid ones on to the backend which then handles the request. Besides the frontend and the backend which are active only during a query, there is also the background thread which is present in every device server. The background thread handles ongoing communications and any kind of events generated by the device and acts accordingly.

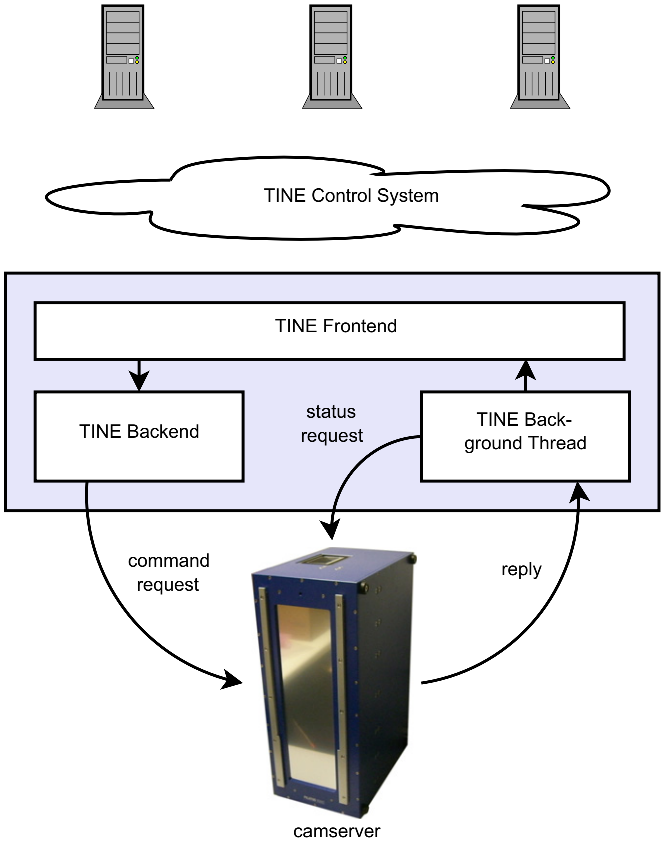


Fig. 1. Schematics of a modularized TINE device server. One or more clients may access a device through the TINE control system to get or set device properties, here of the Pilatus detector. The TINE-specific frontend deals with incoming requests which are dispatched to the Pilatus-specific backend if appropriate. The backend sends commands to the camserver software of the Pilatus detector via a socket connection, a background thread regularly requests status information and retrieves any system replies.

Further, there is also a possibility to combine similar appliances that are used mutually exclusive, e.g. detectors or sample-changer robots, into a generic device. The generic device abstracts away the differences between the specific implementations and allows a client to address a common property, e.g. *load* of a generic samplechanger server to load a sample, independent of the hardware currently in use.

Here it might be prudent to emphasize the point that, due to the frontend–backend separation, our development is decoupled from TINE and can thus be easily ported to other control systems by providing a respective frontend implementation of the server.

2.3. Detector

Users of X33 may currently choose between a mar345 image plate (http://www.marresearch.com) and a Pilatus-1M pixel detector [8]. Further, users may synchronously record wide angle-scattering X-ray scattering (WAXS) with a Pilatus-300k WAXS detector. At the P12 BioSAXS beamline a Pilatus-2M device is available.

The various detectors are all driven by the same generic TINE frontend which provides access to individual devices and the combined SAXS/WAXS setup alike.

2.4. Sample environment

The sample environment of an automated solution SAXS experiment consists of a sample loader device, where the sample is stored until being loaded into the measurement cell.

Users may currently choose from a range of sample changing robots: at X33 are available the samplechanger Mark I that was developed together with the Fraunhofer Institute, Stuttgart [4], and the samplechanger Mark II; at P12, the final setup of the EMBL/ESRF samplechanger is deployed.

All devices are accessible via the same generic TINE frontend which provides unified access to any of the devices.

2.5. Beamline meta server

To coordinate the TINE services described in the previous section, we also designed and implemented a service that mediates user input to the individual components. As it was devised as a service to communicate with other beamline servers, it was termed the Beamline Meta Server (BMS).

Like a hardware server, this service is also separated into a TINE-specific frontend and a service-specific backend. Here, the backend controls the subsequent execution of user-submitted commands for all other beamline servers.

For a convenient setup and usage of the BMS the Python scripting language is employed (e.g. [9]). Each controlled device and each executable command is encapsulated into individual Python classes each with a defined interface, both grouped into Python packages. This organization allows to quickly incorporate new devices and commands as prototyping can be done at the Python command-line prompt while deployment is done by copying the file into the package directory and restarting the BMS application. It is to emphasize that this procedure allows to include any user device that can be controlled by a Python (extension-)module and have it controlled by the BMS during an experiment.

Further, the BMS accepts user input in the form of Python scripts. When such a script is received, it is immediately run by the Python interpreter embedded in the BMS. For each device specific command, a command object is queued in the server's backend for later execution (Fig. 3). Example device setup and command configuration files and exemplary user scripts are available in the supplemental material of this publication.

2.6. Commands

Automation of a SAXS experiment needs to take the full beamline environment provided by appropriate monitor readings into account. Thus each command needs to contain the information required to decide whether the command can safely be executed or not. For example, the samplechanger device itself must be *ready* and not *busy* doing anything else, the storage ring must be *operational* and no refill be scheduled within the next few minutes, the ring and optics shutters must be *open*, and the sample holder temperature needs to be stable at the preset value. If and only if all these conditions are met, a loading command may be executed. Regardless of anything else, the loading command is considered to be *running* if the robot changed its state from *ready* to *busy*, and the process is completed when the device's state changes to *measuring*, i.e. the sample was successfully loaded.

By design, every command starts in the *blocked* state and may not be executed as long as the configured conditions for the *runnable*-state are not met. As soon as they are, the state of the command is changed to runnable. If not executed immediately, external conditions may change and the command may become *blocked* again. Otherwise the command may be submitted and its state shall be changed according to whether the command *failed*, or whether the command was successfully *submitted*. A *submitted* command is checked whether the running conditions are met and if yes, its state is changed to *running*. A *running* command may either *fail* due to hardware problems, be *aborted* by the user or successfully *finish* as expected. All state transitions of the commands are also detailed in Fig. 2.

Importantly, commands are not limited to driving hardware but may also run software when certain conditions, e.g. the availability of a set of input files, are met. Currently implemented are software commands for radially averaging two-dimensional image files to one-dimensional scattering curves and a regularly run summary creation tool.

2.7. Queuing

Utilizing the above command states, automation takes place when a list of commands is executed, not only in sequence, but taking the interdependencies between commands into account, also in parallel. For example: an image plate may be erased while the samplechanger robot flushes the sample holder while the waterbath adjusts the cell temperature while the radially averaging software transforms the most recent two-dimensional image to a one-dimensional scattering curve.

Command scripts, also known as workflows, are sent to the BMS, interpreted and the commands objects are queued there prior to execution (see supplemental material). Fig. 3 visualizes the flow of commands, from initial queuing within the *Transaction Queue*, via the *To-Do* and *Run Queues* to the command *History Queue*. The *Transaction Queue* ensures that only command objects from valid scripts are queued for later execution as any commands in the transaction queue are discarded on syntax or other errors. If the *To-Do Queue* contains commands for execution, only the very first command is polled for a state-change and is started as soon as it becomes *runnable*. However the command remains in the *To-Do Queue* until verified that it is either running or already finished—this is a safe-guard to avoid commands being submitted too quickly if there is some lag between submitting the first command and feedback from the device. For example: let the two commands *detector_scan* and *detector_clear* be queued for the MAR345 image plate. If scanning is submitted and immediately removed from *To-Do Queue*, the clear command may appear to be *runnable* in the next time slice because the detector may not have begun to scan yet. When verified to be running, the topmost command is transferred to the *Run Queue* and the next command is checked whether it can be run. The *Run Queue* keeps all currently running commands and is periodically cleaned, i.e. the finished commands are moved to the *History Queue* for reference.

2.8. Unattended operation

To allow for automated and unattended operation, error handling and error recovery are very important. As currently implemented, the BMS can, for example, cope with the beam loss due to the inputs provided. It is not (yet) possible to repeat previous experiments in the case of beam loss during the measurement; if this happens, the system will wait until the beam reappears and continues with the next command in the queue.

With this restriction in mind, unattended operation of the system is feasible, which is especially useful for screening experiments where the possible loss of one specimen may not be a severe problem.

2.9. Graphical user interface

The Graphical User Interface (GUI) used at the BioSAXS beamlines X33 and P12 and shown in Fig. 4 is laid out in multiple

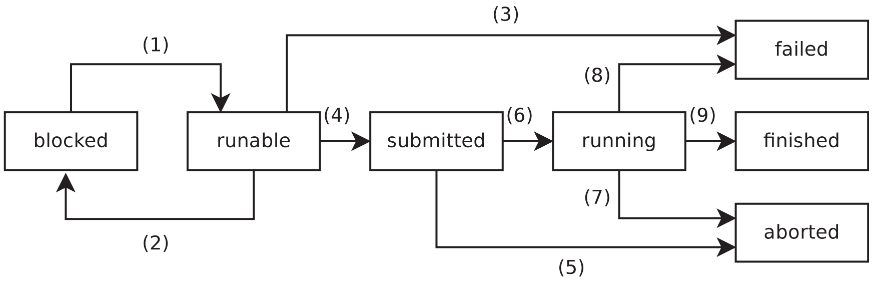


Fig. 2. States and state transitions of commands. A *blocked* command cannot be executed as the conditions for this command are not met. Once they are, the state transition (1) occurs and the command is *runnable*. If a *runnable* command is not immediately executed, external conditions may change and the command may become *blocked* again (2). A *runnable* command that cannot be executed *fails* on attempt to run (3). Successfully started commands change state to *submitted* (4), an intermediate state when the command was successfully sent, but the device itself did not yet change its internal state to meet the expected conditions for the *running* state. When submitted, the command may be *aborted* by the users (5), or become *running* (6). The *running* command may be again be *aborted* by the user (7) or *fail* on its own accord (8). If the final state is reached and thus the command successfully completes, the state changes to *finished* (9).

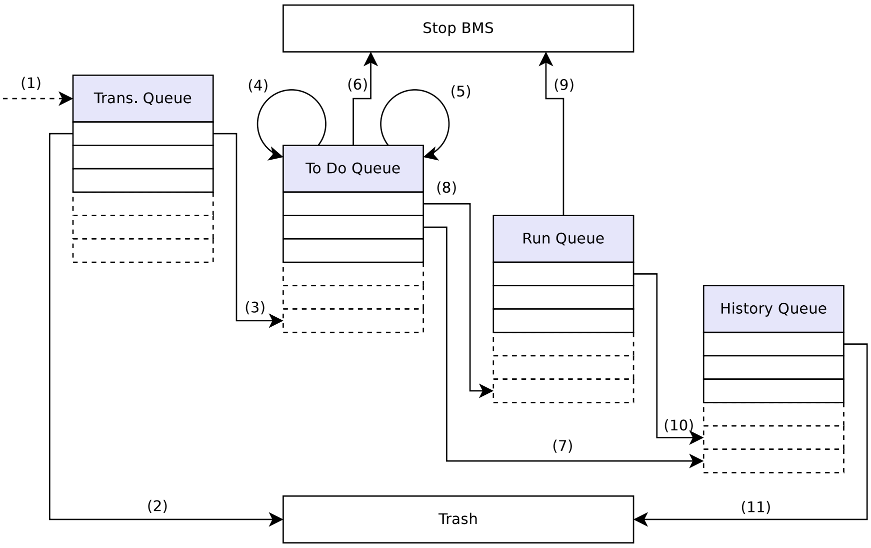
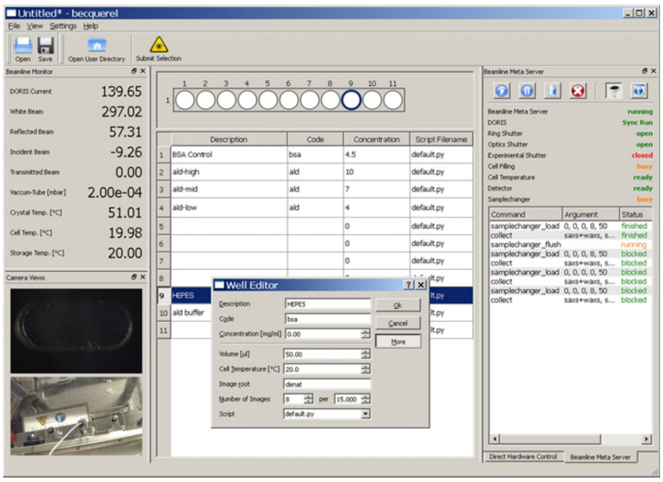


Fig. 3. States and state transitions of the Beamline Meta Server. At any time, scripts which are submitted to the BMS are immediately interpreted and all queueable commands found are added to the *Transaction Queue* (1). If a script error occurs, the *Transaction Queue* is cleared (2), otherwise all commands are transferred to the *To-Do Queue* (3). The *To-Do Queue* is periodically queried to determine the state of the topmost entry; is it blocked or submitted, do nothing (4), start it if it is runnable (5), stop the BMS if it was aborted in submitted state (6), further transfer aborted, failed or finished commands to the *History Queue* (7) and running commands to the *Run Queue* (8). All commands in the *Run Queue* are periodically checked to determine their respective status; if any is aborted, stop the BMS (9), transfer all failed, aborted or finished commands to the *History Queue* (10). The *History Queue* is periodically checked for size. If more than *N* entries are found, all but the most recent *N* are removed (11).



DORIS Current	139.65
White Beam	297.02
Reflected Beam	57.31
Incident Beam	-9.26
Transmitted Beam	0.00
Vaccum-Tube [mbar]	2.00e-04
Crystal Temp. [°C]	51.01
Cell Temp. [°C]	19.98
Storage Temp. [°C]	20.00

	Description	Code	Concentration	Script Filename
1	BSA Control	bsa	4.5	default.py
2	ald-high	ald	10	default.py
3	ald-mid	ald	7	default.py
4	ald-low	ald	4	default.py
5			0	default.py
6			0	default.py
7			0	default.py
8				lt.py
9	HEPES			lt.py
10	ald buffer			lt.py
11				lt.py

Description	HEPES
Code	bsa
Concentration [mg/ml]	0.00
Volume [µl]	50.00
Cell Temperature [°C]	20.0
Image root	denat
Number of Images	8 per 15.000
Script	default.py

Beamline Meta Server	running
DORIS	Sync Run
Ring Shutter	open
Optics Shutter	open
Experimental Shutter	closed
Cell Filling	busy
Cell Temperature	ready
Detector	ready
Samplechanger	busy

Command	Argument	Status
samplechanger_load	0, 0, 0, 8, 50	finished
collect	saxs+waxs, s...	finished
samplechanger_flush		running
samplechanger_load	0, 0, 0, 8, 50	blocked
collect	saxs+waxs, s...	blocked
samplechanger_load	0, 0, 0, 0, 50	blocked
collect	saxs+waxs, s...	blocked
samplechanger_load	0, 0, 0, 8, 50	blocked
collect	saxs+waxs, s...	blocked

Fig. 4. Graphical user interface as used at X33. Beamline Monitor and camera panels are shown to the left, Direct Hardware control and BMS panels to the right. The center widget shows the wellplate layout of the samplechanger robot Mark II, the table with sample descriptions and the extended well edit dialog on top.

sections, a center widget representing the current sample loader's well arrangement plus a table listing the currently available well information and a set of dockable panels. Below we shall shortly describe the available panels and their functionality.

2.10. Panels and views

Panels are subwindows of the main application that may be freely moved, rearranged or hidden if not needed. Here, each panel encapsulates a particular functionality of the beamline. Fig. 4 shows the default GUI layout of X33 after startup.

The *Beamline Monitor* panel in the upper left part of Fig. 4 displays the monitor readings along the beam path together with other parameters of interest, e.g. the pressure within the detector tube and temperature readings of the monochromator, sample storage and the sample cell are provided. All monitor readings are freely configurable, any value available in the TINE control system can be displayed.

Below the beamline monitors, camera views are shown: firstly, the cell camera, monitoring the cell loading status and secondly a camera observing the samplechanger robot's movements. The former view is particularly useful if the system recognizes a bubble and flags a loading error: the user may then employ the direct pump controls to remove the bubble manually. Further, a snapshot image is taken from this camera before and after the exposure to document the loading status of the cell for the respective scattering pattern. Also, the camera views are user configurable, any image source from a web-camera or a TINE servers may be included.

On the right, the *Beamline Meta Server* panel provides information on the status of the BMS. This includes a summary of the device states, the progress of the current operation where applicable and the list of currently queued commands, including their arguments and current status. In this panel, the BMS may be started or stopped, running commands may be aborted if necessary and the user may get an information why the next command is not started yet (blocker-info).

Also on the right, hidden by the panel for the *Beamline Meta Server*, the *Direct Hardware Control* panel provides direct access to hardware devices like the currently active samplechanger robot, the detector, beam shutter, temperature control, etc. The interface provided by this panel is mostly meant for testing and troubleshooting and is usually not used during normal operation.

2.11. Conducting an experiment

Prior to an experiment, the users need to associate the sample and buffer information with the respective wells to provide identification of samples and buffers during analysis. This can be done either by entering the information directly into the table row associated with a particular well, or by double-clicking a well and entering the basic information into the well edit dialog (Fig. 4). If a specific sample requires non-default conditions, e.g. cell temperature, exposure time or a different experiment setup, it may be specified in this dialog together with the basic information. The values defined per well override the general default values which are configurable application wide.

The direct association process is convenient and easy to use for a small number of samples, however, setting up a 96-wellplate in this manner is tedious, the table is much easier to use for copy-pasting information. Beside the on-the-fly set up at the beamline, it is also possible to import a file generated by a stripped-down version of the application that users may download prior to their beamtime and prepare the wellplate files in advance.

With the required information available, an experiment is scheduled simply by selecting, i.e. single-clicking, the wells of interest and pressing the *Submit Selection* button above the well plate. The commands required to conduct the experiment are sent to the BMS, queued and subsequently executed. Progress may be observed in the BMS panel, additional experiments may be queued at any time.

After radial averaging, the data files are picked up for automated data analysis.

3. Automation of data analysis

Processing the data obtained from an automated SAXS experiment is the next essential step both for unattended measurements and for user-controlled experiments. For each specimen measured, one generally needs to (1) radially average the two-dimensional detector images to a one-dimensional scattering curve, (2) subtract the background scattering, (3) determine the overall parameters such as radius of gyration (R_g), molecular mass, excluded volume, maximum dimension (D_{max}) and, optionally, (4) compute an ab initio three-dimensional model. Additional modeling steps may be performed later if information from complementary methods is available.

In this section, we shall first summarize the features of the initial implementation of an automated data analysis pipeline deployed at X33, AUTOSUB, as described by Petoukhov et al. [10] and its adaptation to the BMS framework. Then, we shall detail the rationale and the design of the next generation modular pipeline which are deployed at the BioSAXS beamline of PETRA-III and describe the XML storage format employed for data saving and logging.

3.1. AUTOSUB

The AUTOSUB application may be either run in an online mode accompanying the measurements for real-time analysis or in an offline mode to (re-)process previously acquired data. The program is initialized by a configuration file read during start-up.

While active, AUTOSUB periodically checks a configurable file system location for incoming radially averaged scattering profiles. Radial averaging of two-dimensional images to one-dimensional curves was previously done by the application AUTOMAR [10]. This application was subsequently adapted to work with image files from either detector and is now driven by the BMS via a software command. Whenever a collection of data frames is complete, this averaging application is automatically supplied with all information to apply the radial average transformation to the frame or frames just collected. To monitor for possible radiation damage, successive radially averaged frames from the same specimen are compared. The frames showing radiation damage are excluded from radial averaging and an appropriate warning is given. It is to note that SAXS and WAXS patterns may be collected at the same time and also may be processed in parallel. To simplify file handling for AUTOSUB, the results are stored in separate directory tree, one for SAXS, one for WAXS.

Using the header information of the data files, AUTOSUB then determines which sample and background measurements are to be processed together and applies the appropriate background subtraction accordingly. For each possible subtraction a goodness parameter is evaluated. This parameter takes into account the quality of the Guinier fit at low angles as determined by the program AUTORG [10] and the proximity of sample and background scattering at high angles where the useful signal is expected to be relatively small but positive. The subtracted curve with the highest goodness parameter is progressed to the further analysis.

Based on the radius of gyration as evaluated by the program AUTORG the application AUTOGNOM [10] looks for the optimal maximum particle size D_{max} in the range from $2R_g$ to $4R_g$ and computes the $p(r)$ function. Subsequently, the latter is then passed to the application DAMMIF [11] which computes an ab initio model and estimates the excluded hydrated volume. The molecular mass of the particle is determined by comparing its forward scattering with that of a standard with known molecular mass. The scattering patterns are screened against the scattering profiles deposited in the DARA database [12]. Any structural information obtained is made available in a summary file in XML format as detailed below.

The AUTOSUB system has been in user operation at X33 from 2007 to 2011 and at ID14-3 at the ESRF in Grenoble from 2009 to 2011, serving in total over 400 user groups.

3.2. Next generation analysis pipeline

The data analysis pipeline should work in real time to provide immediate feedback to the user during the measurements. The approach implemented in AUTOSUB is based on sequential execution of data analysis steps. Therefore, the next data processing step cannot be performed until the previous step is finished which may lead to an accumulation of delays. Furthermore, AUTOSUB is hardware dependent making it difficult to adapt to new environments. To overcome these limitations we have developed a set of hardware-independent tools for automated SAXS data manipulation and analysis which allow for rapid sample characterization and provide the overall parameters. With these tools an automated, robust, flexible SAXS data analysis pipeline was implemented. This pipeline now covers major processing and interpretation steps including background subtraction, normalization, extrapolation to infinite dilution, estimation of the overall parameters, calculation of the distance distribution function and ab initio low-resolution model building.

In the available SAXS data processing packages [13,14] single operations are directly linked to the master application, which provides limited or no scripting abilities. This makes it difficult to adapt the sequence of steps in data processing schemes. We have therefore introduced modularized data processing tools, each designed as a small and simple application to perform a specific task. Thus, each data processing step is performed by one tool or a set of tools. No tool may call another tool; if certain processed input data is needed, the caller should provide this data, presumably obtained from another tool in advance. For example, to subtract the background from the sample one needs to decide if the background scattering pattern measured before the sample, after the sample or an average of the two background measurements should be used. Here, three applications are needed, one for comparing the background scattering patterns, another to perform the subtraction and yet another to estimate the quality of the subtracted data.

The developed tools for automated data processing include:

- DATCMP calculates the discrepancy between two or more data sets; used e.g. for checking of radiation damage and for comparing background scattering patterns.

- DATAVER averages two or more data sets; used e.g. for averaging backgrounds or multiple sample exposures.
- DATOP performs arithmetic operations; used e.g. for subtracting background scattering from the sample, for scaling against monitor values or sample concentration.
- ALMERGE merges data collected from two different concentrations and extrapolates it to infinite dilution assuming moderate particle interactions.
- AUTORG automatically computes R_g and $I(0)$ using the Guinier approximation, estimates data quality, finds the beginning of the useful data range.
- DATCROP crops the range of experimental data points.
- DATGNOM estimates D_{max}, computes the distance distribution function $p(r)$ and the regularized scattering curve.
- DATPOROD computes Porod volume from the regularized scattering curve.
- DAMMIF creates an ab initio dummy atoms model, estimates volume.
- DAMAVER compares multiple ab initio models, finds the most probable one.

All modularized tools use the open source *libsaxsdocument* library to read the experimental data files. This library was designed to provide a format-independent interface for reading and writing data files in various formats. The design of the library allows one to incorporate upcoming or existing data formats such as SasCIF [15], NEXUS [16] or XML-based formats without changing each particular tool every time a new format is added.

The data processing pipeline itself is implemented as a separate application that supervises serial or parallel execution of the above modules. Each data processing step is represented by an individual component that employs one or more modularized tools to perform operations (Fig. 5). These components are therefore decision-making blocks that do not perform any actual data processing. The components communicate with each other by passing messages. A message is sent when a particular event occurs, e.g. when a new file becomes available for processing or when a tool finishes. A message may be received by several components. This way of connecting different components enables one to modify the behavior of the pipeline to meet different requirements, e.g. by including or excluding certain steps if needed. This architecture allows one to introduce new components that may perform additional data processing operations sequentially or in parallel.

Operation of the pipeline begins after a detector image is radially averaged to a one-dimensional data set which is stored on a file system where it becomes available for the data processing pipeline. The *File System Notifier* component monitoring the storage file system then detects the presence of the new data set and notifies the subsequent components.

3.3. Background subtraction

The measurement of the solution is often surrounded by two solvent measurements which are compared using DATCMP to characterize the stability of the solvent measurements. The scattering pattern of the macromolecular solute is then obtained by subtracting the scattering of the solvent.

If the compared solvent data sets are statistically indistinguishable from each other, appropriate averaging operations are done with DATAVER. The averaged background is subtracted and the data is scaled against sample concentration by DATOP. If there is a noticeable difference between the compared solvent data sets, then the individual solvent data sets are subtracted along with the averaged background thus creating three alternative subtracted data sets. For every data set, the quality is estimated by AUTORG based on the linearity of the Guinier plot [17], the accuracy of the R_g value and the distance of the Guinier interval from the origin. The data with the best quality estimate is kept for further analysis.

3.4. Evaluation of the overall parameters

It is imperative for reliable data analysis to get an accurate initial estimate of the particle size. This is done in the *Guinier* component that employs AUTORG to estimate the radius of gyration from the low angles using the Guinier approximation. The molecular mass is then estimated from the forward scattering $I(0)$. The R_g value can be estimated not only from the low angles of the curve, but also from the $p(r)$ function. Clearly the two R_g values obtained from these two methods should correlate, i.e. estimating the R_g value from the Guinier approximation allows us to automate the calculation of the distance distribution function. This is done by the *P(r)* component. First, the quality of the data as reported by AUTORG shall be verified to be in an acceptable range. If data quality is deemed too low, no further processing is performed. Otherwise the data is processed by DATGNOM, a derivative of the AUTOGNOM program [10]. DATGNOM performs multiple GNOM [18] runs varying D_{max} between $2R_g$ and $3.5R_g$, optimizing the $p(r)$ function to match its Fourier transformation to the experimental data and its R_g to the one estimated by the AUTORG tool. The excluded volume of the particle may then be computed directly from the scattering pattern using the Porod

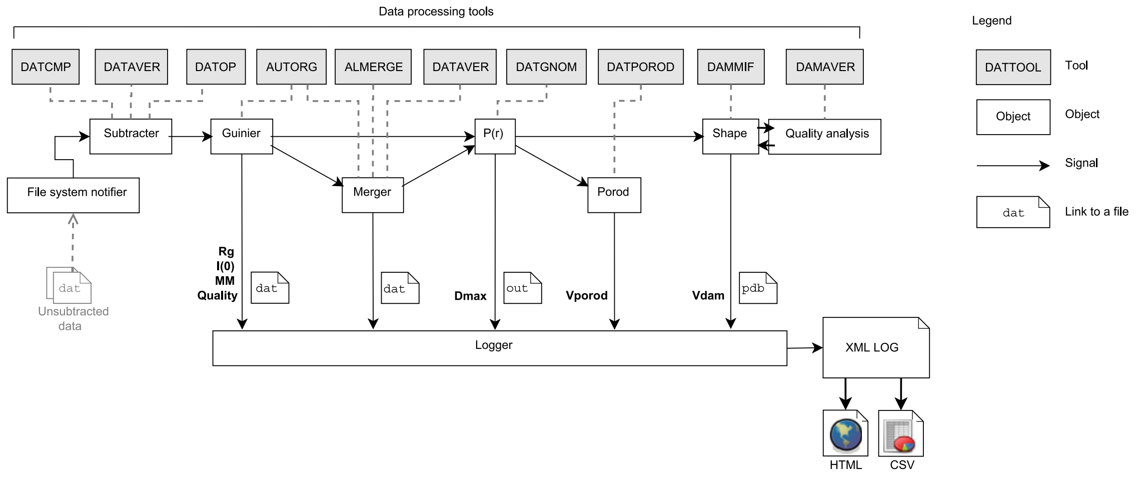


Fig. 5. The data analysis pipeline with modularized tools.

equation [19]; however when applying this formula directly to the experimental data one might be confronted with difficulties due to higher noise level at higher angles. The DATPOROD tool, encapsulated by the *Porod* component, estimates the volume by applying the Porod equation not to the experimental data but to the regularized curve received from the $P(r)$ component. Prior to this an appropriate constant is subtracted from each intensity point to force the s^{-4} decay following Porod's law for homogeneous particles.

3.5. Processing multiple concentrations

Inter-particle interference may affect the initial part of the scattering curve. To eliminate this influence one can in many cases extrapolate to infinite dilution, i.e. having measured a sample with different concentrations one reduces the data to zero concentration assuming a linear dependence of interference effects on concentration. This is done by the *Merger* component which extrapolates several data sets to one "infinite dilution" data set using the ALMERGE tool. Prior to this, the R_g-concentration dependence is checked; if it is non-linear (e.g. because of aggregation) or the R_g changes insignificantly with concentration (no inter-particle interference), the extrapolation step is skipped. If no concentration effects are observed, i.e. R_g does not change, the highest concentration data set is marked as final and passed further down the pipeline. If only two samples of satisfactory quality and different concentrations are available, they are merged by ALMERGE without extrapolating to infinite dilution. The overall parameters of every composite curve are estimated as described above.

3.6. Shape determination and quality control

To reconstruct the low resolution shape of the particle the *Shape* component executes the ab initio modeling program DAMMIF. A single DAMMIF run in fast mode produces a rough shape along with a volume estimate within a few minutes; however, it is important to do this without blocking the pipeline while other calculations are in progress. Therefore, the time-consuming tools are executed in the background, in parallel to the other tasks. One of the challenges of automated data processing lies in determining an appropriate scoring to characterize the reliability of the results. Apart from the data quality estimation implemented in AUTORG, values obtained from different tools may be used to independently confirm the results. The *Quality analysis* component compares the molecular mass estimated from volume and $I(0)$ of infinite dilution data. Data simulated from 53 protein models with molecular mass ranging from 14 kDa to 500 kDa show that for globular proteins the Porod volume in nm^3 is about 1.6 times the molecular mass in kDa; the volume of a DAMMIF model in nm^3 is typically about twice the molecular mass in kDa. If variation of the three derived molecular mass values is in an acceptable range, then a good consistency check for the ab initio model can be performed using DAMAVER [20]. To assess the uniqueness of the low resolution shape, DAMMIF is executed by the *Shape* component several times (6–12 by default, depending on available computational resources). The normalized spatial discrepancy (NSD) value, which is computed by DAMAVER, characterizes the stability of the obtained low resolution models. In order to save computational resources, multiple DAMMIF runs are performed only on data from the composite curves.

3.7. Storage of the results

It is vital for automated procedures dealing with large amounts of data that both the obtained information and history of the data analysis (including the experimental data and the computed parameters and models) are easily retrievable in human- and machine-friendly forms. For a convenient hierarchical data storage and report generation, an XML-based file format was employed. After each data processing step the *Logger* component receives the corresponding message and a summary of results such as R_g, molecular mass, quality estimation, D_{max}, volume etc. is immediately written to the XML-summary file, which remains fully consistent and readable during the experiment. The format of the file allows for data processing using the standard Extensible Stylesheet Language (XSL). Stylesheets to transform to a human-friendly HTML representation and to comma-separated value tables are available, other conversions are possible. The described system is used at the X33 beamline at EMBL Hamburg since October 2008 and at the ID14-3 beamline at the ESRF in Grenoble since June 2009.

4. Remote access

With the software setup presented here it became possible to employ NX of NoMachine (www.nomachine.comas a remote desktop solution to grant the external users remote access to the beamlines [21]. The users need to install the free NX client before connecting to x33.embl-hamburg.de with their assigned username and password to access the beamline controls. Remote users have full control over the measurement process via the graphical user interface described above while the task of the local contact is limited to making available the samples in the samplechanger robot.

To our knowledge the world's first remote SAXS experiment was conducted in this way on May 26, 2009, during a course of biological small angle scattering at Nanyang Technological University in Singapore. The second live presentation was held during the Small Angle Scattering conference 2009 in Oxford. Since then, multiple external user groups have successfully used the remote access to X33 for their SAXS data collection. Remote access in this manner will also be available for the P12 BioSAXS beamline at PETRA-III.

5. Future plans

To further improve the system, we plan to develop a software for full autonomous operation of a beamline, e.g. for unattended screening of a 96-wellplate. When provided with a wellplate information file created by one of the user interfaces, this software will then execute the full measurement sequence, taking the general environment, in particular status of the synchrotron, into account.

Future plans for the automated analysis pipeline include a scripting interface to simplify the addition of tools and decision making blocks to further increase its usability and adaptability to various environments.

6. Conclusions

In conclusion, a modular and flexible automated SAXS data acquisition and analysis system was developed. A beamline meta server (BMS) was implemented as a key component of the automated data collection. Each hardware device of the SAXS experiment setup has a device server that facilitates reliable communication between associated hardware and software entities. Full control of the hardware devices by BMS coupled with an automated data analysis pipeline permits fully automated and remote controlled SAXS studies.

Feedback of over 400 external user groups that performed their experiments at the BioSAXS beamlines of EMBL Hamburg during beam periods from 2009 to 2011 shows that the presented setup allows for fast, convenient and reliable experiments, leading to improved fidelity of the structural results.

Due to its modularity and basically control-system independence, this flexible system setup can readily be adapted to various control systems at other synchrotron radiation sources.

Acknowledgments

This work was supported by the WeNMR project (European FP7 e-Infrastructure grant, contract no. 261572, www.wenmr.eu and the BMBF research grant SYNC-LIFE (contract no. 05K10YEA). The authors would also like to thank Clement Blanchet, Manfred Roessle and Petr Konarev for discussions and in-development testing of the system. Also we would like to thank Philip Duval for his constant TINE support.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org.10.1016/j.nima.2012.06.008.

References

[1] H. Mertens, D.I. Svergun, Journal of Structural Biology 172 (2010) 128.

[2] M.D. Winn, A.W. Ashton, P.J. Briggs, C.C. Ballard, P. Patel, Acta Crystallographica Section D 58 (2002) 1926.

[3] T.M. McPhillips, S.E. McPhillips, H.-J. Chiu, A.E. Cohen, A.M. Deacon, P.J. Ellis, E. Garman, A. Gonzalez, N.K. Sauter, R.P. Phizackerley, S.M. Soltis, P. Kuhn, Journal of Synchrotron Radiation 9 (2002) 401.

[4] A.R. Round, D. Franke, S. Moritz, R. Huchler, M. Fritsche, D. Malthan, R. Klaering, D.I. Svergun, M. Roessle, Journal of Applied Crystallography 41 (2008) 913.

[5] C.E. Blanchet, A.V. Zozulya, A.G. Kikhney, D. Franke, P.V. Konarev, W. Shang, et al., Journal of Applied Crystallography 45 (2012) 489–495, http://dx.doi.org/10.1107/S0021889812013490.

[6] S. Classen, I. Rodic, J. Holton, G.L. Hura, M. Hammel, J.A. Tainer, Journal of Synchrotron Radiation 17 (2010) 774.

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

[8] C. Broennimann, E.F. Eikenberry, B. Henrich, R. Horisberger, G. Huelsen, E. Pohl, B. Schmitt, C. Schulze-Briese, M. Suzuki, T. Tomizaki, H. Toyokawa, A. Wagner, Journal of Synchrotron Radiation 13 (2006) 120.

[9] D. Beazley, Python Essential Reference, 4th ed., Addison-Wesley Longman, Amsterdam, 2009.

[10] M.V. Petoukhov, P.V. Konarev, A.G. Kikhney, D.I. Svergun, Journal of Applied Crystallography 40 (2007) 223.

[11] D. Franke, D.I. Svergun, Journal of Applied Crystallography 42 (2009) 342.

[12] A.V. Sokolova, V.V. Volkov, D.I. Svergun, Journal of Applied Crystallography 36 (2003) 865.

[13] J. Ilavsky, P.R. Jemian, Journal of Applied Crystallography 42 (2009) 347.

[14] M.V. Petoukhov, D. Franke, A.V. Shkumatov, G. Tria, A.G. Kikhney, M. Gajda, C. Gorba, H.D.T. Mertens, P.V. Konarev, D.I. Svergun, Journal of Applied Crystallography 45 (2012) 342.

[15] M. Malfois, D. Svergun, Journal of Applied Crystallography 33 (2000) 812.

[16] D. Maddison, D. Swofford, W. Maddison, Systems Biology 46 (1997) 590.

[17] A. Guinier, Annals of Physics 12 (1939) 161.

[18] D.I. Svergun, Journal of Applied Crystallography 25 (1992) 495.

[19] O. Glatter, O. Kratky, Small Angle X-ray Scattering, Academic Press, London, 1982.

[20] V.V. Volkov, D.I. Svergun, Journal of Applied Crystallography 36 (2003) 860.

[21] S.M. Soltis, A.E. Cohen, A. Deacon, T. Eriksson, A. González, S. McPhillips, H. Chui, P. Dunten, M. Hollenbeck, I. Mathews, M. Miller, P. Moorhead, R.P. Phizackerley, C. Smith, J. Song, H. van dem Bedem, P. Ellis, P. Kuhn, T. McPhillips, N. Sauter, K. Sharp, I. Tsyba, G. Wolf, Acta Crystallographica Section D 64 (2008) 1210.