electronic reprint

Acta Crystallographica Section D
Biological Crystallography
ISSN 0907-4449
Editors: **E. N. Baker and Z. Dauter**

Integrated software for macromolecular crystallography synchrotron beamlines II: revision, robots and a database

John M. Skinner, Matt Cowan, Rick Buono, William Nolan, Heinz Bosshard, Howard H. Robinson, Annie Héroux, Alexei S. Soares, Dieter K. Schneider and Robert M. Sweet

Acta Crystallographica Section D
Biological Crystallography
ISSN 0907-4449

Integrated software for macromolecular crystallography synchrotron beamlines II: revision, robots and a database

John M. Skinner, Matt Cowan, Rick Buono, William Nolan, Heinz Bosshard, Howard H. Robinson, Annie Héroux, Alexei S. Soares, Dieter K. Schneider and Robert M. Sweet*

Biology Department, Brookhaven National Laboratory, Upton, NY 11973, USA

Correspondence e-mail: sweet@bnl.gov

Received 3 April 2006
Accepted 1 August 2006

This manuscript chronicles the evolution of software used originally to control a diffractometer at a macromolecular crystallography beamline. The system has been augmented and rewritten. A modular and carefully organized suite of programs now handles the whole experimental environment from a single vantage point. It provides automatic logging of the experiment and communication with the user, all the way from an initial proposal to perform the work to the end of data collection. This has included construction of a relational database to organize all details of the experiment and incorporation of a robotic specimen changer to provide automation for high-throughput applications.

1. Background and introduction

For over a decade, workers in the field have recognized that efficient use of a macromolecular crystallography (PX) beamline's experimental station demands software that is easy to use and reliable. Also, in this era where what we used to call 'computing' is now named 'information technology', we can expect our software to keep track of and organize 'information' about our experiments as well as collecting and storing the data. The programmers of this software have dual responsibilities not only to design an intuitive interface to the experiment, but also to provide underlying control code that they can adapt easily to the ever-changing hardware that it controls. In particular, hardware had traditionally been limited to beamline motors, scalers, diffractometers, and X-ray detectors. However, new challenges are arising for software designers as sample-changing robots, the need for remote operation, and associated project-management database systems are required to meet the demands of high-throughput crystallography. It is not sufficient that the scientific users find the software to be easy to use, to work flawlessly, and to organize and annotate data in a useful way; it must also be easy to understand, expand, and modify for the programmers. This document, an update of our previous report (Skinner & Sweet, 1998), presents our progress towards this objective.

We in the NIH- and DOE-funded Macromolecular Crystallography Research Resource (PXRR) at Brookhaven National Laboratory's National Synchrotron Light Source have developed several nicely integrated yet structurally independent components of our software system. The front end of the experimental control is *CBASS* (*Crystallography at Brookhaven Acquisition Software System*), a user-friendly Python-based system that integrates data-collection and

beamline control. The software also communicates with a sample-mounting robot[1] and a project-management relational database system, PXDB, which is accessible to the user through both web-based tools and *CBASS*. An additional component, completely hidden from users and routine operators, is the highly organized and flexible control system for beamline components. A crucial feature of the software is that the software looks and feels the same at every one of our six beamlines, with differences hidden in beamline-specific scripts and variables hidden in startup files. This greatly facilitates the movement of users among the experimental stations.

In our continuing effort to enhance the productivity of visitors to our NSLS beamlines, we have presented numerous innovations to the world. These include the first electronic area-sensitive detector at a beamline in the US and on-site computer resources for real-time data reduction (1991). We developed the first graphical user interface (GUI) for diffractometer and data-reduction control (Skinner *et al.*, 1993), an integrated GUI for beamline control and data collection (Skinner *et al.*, 1996), automated spectrum analysis and energy optimization during MAD data collection (1995), 'one-button' data reduction with graphical analysis tools (Skinner & Sweet, 1998), integrated data-collection strategy prediction (Ravelli *et al.*, 1997), web-based remote observation of the experiment (1998), web-based experimental control (1999), and automatic experiment logging into a web-viewable log file (Sweet *et al.*, 2001). We also have developed a broad and deep mail-in program for collaboration and service wherein specimens are shipped to our scientific staff for measurement[2] (Robinson *et al.*, 2006). This work depended on early work in diffractometer control (Messerschmidt & Pflugrath, 1987), and other work has appeared more recently. In particular, the *Blu-ice* program has been described by the Stanford group (McPhillips *et al.*, 2002) and a system with much of the functioning of that described here has been reported by the structural genomics group at SPring-8 (Ueno *et al.*, 2005).

With careful attention to modular design, the software systems described in our earlier references were easily adapted to run detectors manufactured by Enraf–Nonius, MAR, Brandeis, and ADSC. The program also controlled diffractometers produced by Enraf–Nonius, MAR and Crystal Logic. With the development of a suite of convenience tools to facilitate data-collection strategy, data processing, and remote control and observation, the package served the NSLS PX community well throughout the 1990s. Given the early roots of the programming efforts, it is not surprising that the programs comprised well over 100 000 lines of C and Fortran code. To meet the needs of a continually changing landscape of hardware and software, we needed to perform a major restructuring of our entire software effort. We relate the results of this work below.

[1] We have adapted the LBL-ALS automount robot in collaboration with Thomas Earnest and his colleagues (Snell *et al.*, 2004).
[2] Also known as a 'FedEx' operation.

2. Evolution of the software

By 2001, our motor-control hardware needed replacing. We chose to accomplish motor and scaler control through *EPICS* (*Experimental Physics and Industrial Control System*; http://www.aps.anl.gov/epics/), some aspects of which we will describe later, implemented on a VME-based system. Prototyping of the higher level software control began in Python and we soon recognized that Python would be an excellent language for the production system. Use of Python enabled us rapidly to develop and deploy a stable system to provide command-line control and scripting facilities that would sit between *EPICS* and our existing Motif-based beamline-control GUI. The code was small, readable, and adaptable.

Given the success of the Python/*EPICS* project, Python was used in 2002 to develop a program to replace the massive Fortran code in *MADNES* (Messerschmidt & Pflugrath, 1987), the underlying data-collection control software. The approach was straightforward, using *SWIG* (*Simplified Wrapper and Interface Generator*; Beazley, 1996) to generate Python interfaces to low-level diffractometer- and detector-control libraries supplied by the vendors, and then providing command-line control logic and scripting capabilities with Python. The result was that over 70 000 lines of Fortran code were replaced by less than 6000 of straightforward Python code with loss of only some little-used functionality. At this point, all of our underlying data-collection and beamline-control code was implemented in Python, while the Motif/C-based GUIs remained.

With the new sample-changing robot and project-management database system on the horizon in 2003, it was clear that significant and frequent changes would need to be made to the user-interface software to accommodate robot control, new data-collection protocols, and interaction with the database system. Although the existing GUIs were robust and stable, we recognized that extensive modifications and extensions to the Motif/C code would be extremely labor-intensive. Starting from scratch in a different programming language seemed like a daunting task, but if performed properly the result would be a flexible system that could be adapted easily to service the changing aspects of the user's interface to the experiment. The result is a package accessible from a single GUI that has nearly all of the functions of the original system described in the original publication (Skinner & Sweet, 1998). It now also includes intimate control of the sample changer, the database, and the components of the beamline that were accessible before through a separate GUI.

Despite our having previous GUI-programming experience in Java and Tcl/Tk, we chose Python/Tkinter to rewrite and extend our graphical interface. Tkinter is the Python interface to Tk, the GUI toolkit for Tcl/Tk, so it was easy to learn to construct GUIs with Python. Development was facilitated further by the rich set of interface components, or widgets, provided by the Python megawidgets (Pmw) package. The Tkinter/Pmw combination provided a powerful, easy to use and well documented software platform for GUI development. WxPython, another toolkit for Python-based GUI

development that has become popular, was not a consideration for us in 2003, owing in part to sparse documentation and to a wider acceptance of Tkinter as the standard GUI toolkit for Python at that time.

Also, consistent with our modular approach, we constructed the new system with 'client/server' architecture. As we will describe below, multiple clients (copies of the *CBASS* GUI or other clients such as a java-based web client) communicate with the single *CBASS* server at each beamline. This server is invisible to users and depends on additional control modules to communicate with each piece of hardware. By mid-2004, the six PXRR beamlines were under control of the rewritten and extended Python-based *CBASS*.

3. *CBASS*, the experiment-control software

The *CBASS* window is a 'notebook' with up to six 'pages' that can be selected with tabs named Collect, Setup, Robot, Pucks, Canes and Beamline. Which pages are to be displayed in a particular environment are controlled by environment variables set in the *CBASS* configuration file. Fig. 1 shows the main window with the six page tabs available at the time of writing. The entering of text in windows or selection of buttons on the GUI triggers transmission of a command to the *CBASS* server. These same commands can be entered as text directly into the Command window, used typically by operations staff to obtain status details. Notice the connection to our experiment-tracking database, PXDB: group and project identifiers (PxID) can be selected from pop-up selection boxes with appropriate buttons next to the command line. Counter readouts and monochromator wavelength are also displayed in this horizontal strip, which is always visible. The white scrolled window displays output from the server.

3.1. The Collect page

This page, shown in Fig. 1, is the origin of all operations of the diffractometer and detector. It shows the current status of both in the middle panels on the left and right. There are also several options for immediate control in these panels and the row of control buttons below, such as setting a diffractometer axis, opening or closing the

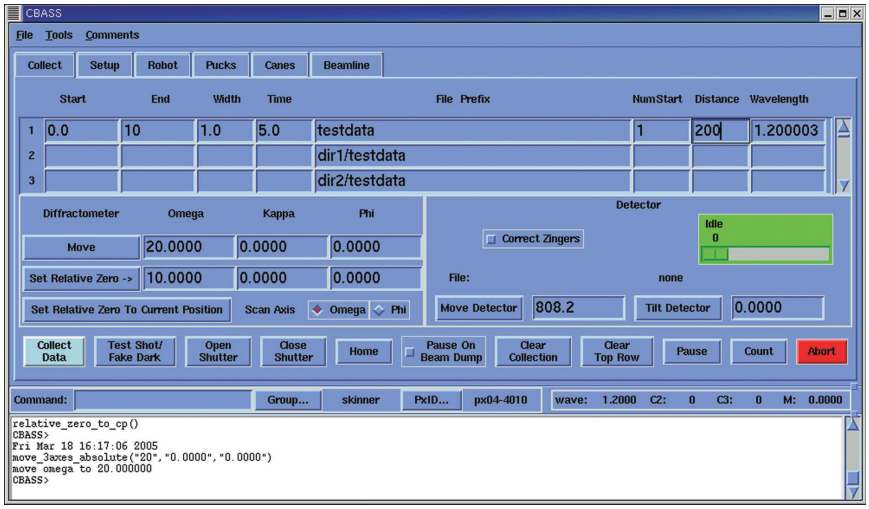

Figure 1
Overview of *CBASS* display showing the Collect screen.

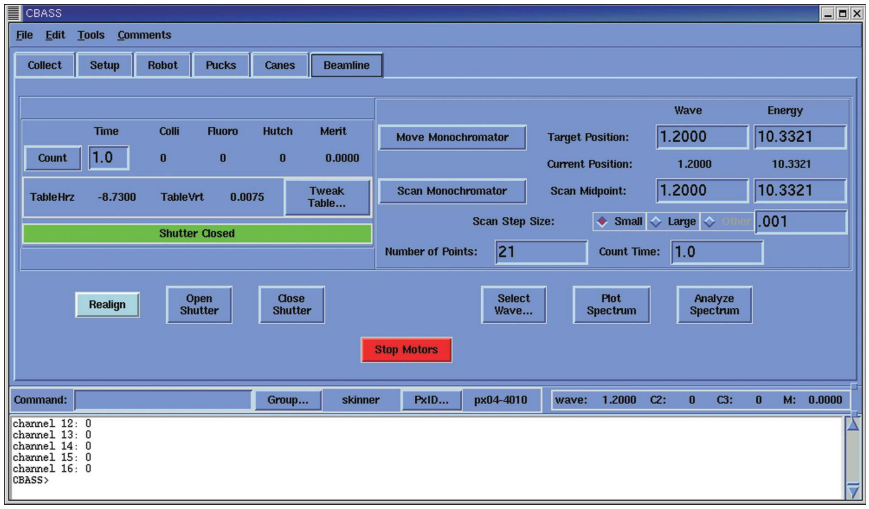

Figure 2
The *CBASS* beamline-control page.

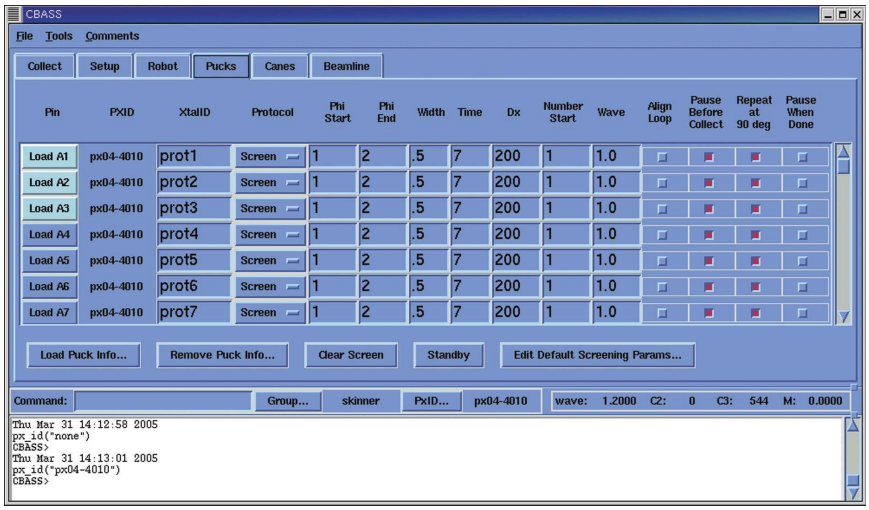

Figure 3
The Pucks page from *CBASS*. This page displays information about individual specimens in cassettes (pucks) for the cryogenic automount robot. The information is typically loaded from the PXDB database.

X-ray shutter *etc*. At the top is a scrollable table, each line of which represents a sweep of data collection. The Collect Data button starts up automatic data collection from the instructions present in the sweep table, starting at the top and working its way down. Colors change during operation so the operator can monitor progress at a glance. The Test Shot button takes only the first image from the sweep indicated in the top row of this table. It also operates quickly by taking a quick and only approximate 'dark' image from the detector for image correction. The Wavelength box is live: the current wavelength is shown when the page is opened and then the monochromator will move to the wavelength indicated, just before the sweep is started, if it has changed. If the character I or P follows the number, the excitation spectrum will be scanned and the true rising inflection point or peak will be chosen, respectively, for the sweep of data. This allows automatic MAD or SAD data collection with quite precise selection of the desired wavelength. This table can also be loaded from the Pucks, Canes or Setup page, as we will describe later.

3.2. Beamline page

Basic beamline control, such as monochromator scans and frequently used beamline alignments, are controlled from the Beamline page (Fig. 2). The intent of this page is to provide an interface to the beamline components and functions used during typical experiments. More complex beamline operations, such as mirror adjustments, which are normally performed by beamline staff, are handled with our separate *GrEpx* software or *EPICS* interfaces, providing full access to beamline motors.

Selection of the Count button will display the X-ray intensities as counter readings at the collimator, fluorescence detector, and an additional upstream counter. Monochromator settings and scan parameters are controlled in the right-hand panel. The Select Wave button displays a periodic table with selectable elemental absorption edges. Other buttons at the bottom of the page allow for beamline-specific realignments, shutter control, and monochromator scan analysis. Although the Realign button nearly always works to optimize the X-ray beam through the diffractometer apertures, the diffractometer table can be adjusted with a Tweak Box (Skinner & Sweet, 1998), which employs left and right mouse clicks to adjust motorized components.

3.3. Pucks, Canes, Setup and Robot pages

The Pucks page (Fig. 3) is designed to be used with the PXRR sample-mounting robot. The Pin buttons A1–D16 map to the robotic automounter dewar, which contains four crystal cassettes (pucks) A–D, each of which contains 16 crystal-mounting pins. The investigator will load information describing the contents of each puck and the desired actions on each crystal into PXDB with a web-based interface that looks essentially like this page. Pushing the Load Puck Info button in *CBASS* will pop up a selection box that allows the operator to select pucks from PXDB entries. Selecting a puck

Figure 4
The HTML Log in action.

will fill in the chosen puck position (A–D) with information and parameters found in PXDB.

The data-collection parameters of all specimens listed in the Pucks page can be edited globally or individually, as desired. Pushing a Pin button (*e.g.* 'Load A') will load information into the first unused row in the sweep table of the Collect page (Fig. 1). Selection of the Collect Data button from there, after items have been loaded in this fashion, will cause the robot to mount samples from the appropriate puck/pin positions and proceed to obtain the data. Directories of the form '`./[PXID]/[XtalID]/sweep_number/`' will be created automatically, into which the resulting data images will be stored. 'Sweep number' is determined from the record of previous data sweeps performed on the XtalID with the current PXID as stored in the PXDB. If the Protocol option menu was set to 'Screen' in PXDB, then data collection will employ default parameters established at the beamline, which can be changed by selection of Edit Default Screening Params . . . .

The scientists in our mail-in data-collection program (Robinson *et al.*, 2006) receive crystal-filled dewars for on-site data collection on behalf of remote investigators. The crystals, in vials, are typically held in tubes or 'canes'. These investigators transfer information about the specimens on a Canes page from the PXDB that can be loaded into *CBASS* in the same way that the Pucks screen is populated. Vial positions in the canes are numbered 1–6 starting at the bottom of the cane. Pushing the Load button for any sample will load the Collect screen using the same file and directory-creation protocol as the Pucks page. Some experimenters liked the automatic directory-creating protocol found in the Pucks and Canes pages and requested that a page be constructed where one could take advantage of those protocols for individual specimens. The Setup page provides this possibility.

The Robot page was prepared to provide simple mounting and dismounting of individual samples, where the only information provided is one crystal position in a puck, *e.g.* C-7. However, the Pucks page provides everything necessary to drive the robot during automated data collection for any number of crystals.

3.4. The Tools menu

A variety of features can be run from the Tools menu on the main menu bar.

3.4.1. AutoAdxv. AutoAdxv executes the *ADXV* image-viewing program from Area Detector Systems Corp. with the 'autoload' option. In this mode, *ADXV* displays each image as it is collected.

3.4.2. AutoMax. The *CBASS AutoMax* facility can be used on certain beamlines to perform automatic beamline alignments during data collection. This tool prompts for the frequency in minutes for which the user would like optimizations to be run. When it is time for *AutoMax* to perform an optimization, it pauses data collection, runs the beamline-specific realignment and then resumes collection.

3.4.3. CrystalView. The CrystalView tool displays a live magnified video image of the crystal. On beamlines with a motorized goniometer head this image can be used to center the crystal in response to the user clicking the crystal center on the image.

3.4.4. Edit Preferences. Several options can be toggled from the Edit Preferences menu. Most of these are placed in the `$CBASS_SITE_FILE` by the beamline administrator to establish default behavior for the beamline. HTML Logging turns on and off the HTML logging of the experiment (see below). Max Before Sweep determines whether the beamline-specific beam-intensity optimization is performed before each data sweep. I_0 Logging turns on/off collection of a continuous record of the intensity of the X-ray beam striking the crystal: the I_0 logging. The record contains a count that is performed during the recording of each image. A plot of the results is placed in the data html directory, a link to that plot is in the log for each run, and a 'master' I_0 file is generated in the directory from which *CBASS* was started.

3.4.5. The HTML log. *CBASS* generates a record, formatted in HTML (hypertext markup language), that is easily viewed in any browser and is structured as follows. A 'root' HTML document contains monochromator scan plots and analysis, as well as other parameters and links to the HTML data logs of each sweep. The HTML data logs (Fig. 4) pointed to by the root HTML document contain JPEG 'thumbnail' images of each data image, periodic pictures of the crystal, and an I_0 plot at the end of the run if I_0 logging is activated.

4. A project-management and experiment-tracking system, PXDB

As our mail-in data-collection project (Robinson *et al.*, 2006) developed, it became clear that we required a reliable mechanism both to communicate with our clients and to keep records automatically of the work that was performed. To meet these needs and simplify work for all of our users and staff, we built a project-management system that is accessible through the web and integrated it with our data-collection system, *CBASS*. Ultimately, this project aims to automate, as much as possible, generation of the diffraction-experiment components of a Protein Data Bank submission (Berman *et al.*, 2000). We call this system PXDB and the access point is http://www.px.nsls.bnl.gov/pxdb/.

The system is designed to integrate seamlessly with the NSLS's Proposal, Allocation, Safety and Scheduling (PASS) system (https://pass.nsls.bnl.gov/), both to provide a mechanism for potential users to request beam time for experiments (PASS) and then to keep track of those experiments (PXDB), especially if they were being performed remotely or as mail-in clients. A major objective is that the investigator should never have to type in the same thing twice. To this end, each individual associated with an experiment will have personal contact information stored in the Brookhaven Laboratory Guest Information System, accessible both to PASS and PXDB, and individual projects are defined once and then are

stored so that the investigator can find important information easily.

We index every user and project to the group leader or principal investigator on the project and there is password protection for each group's information within the database. A simplified view of the organization may be seen schematically in Fig. 5. The database is organized by research groups or programs, with a principal investigator at the head. There may be several projects associated with each group or program. Individual experimenters will be associated with one or more projects and may be part of several groups. There may be many crystals used for each project. Each data sweep is recorded for each crystal and these need not be performed during the same experimental session. Although this section of the database is not represented here, the PXDB entries for the dewar-shipping canes or automounter cassettes (pucks) may cut across project boundaries.

After logging in to the PXDB, a user has access to all of the group's password-protected information. From there a user has access to all the group's information. Individuals can be added to or deleted from the group and new projects may be defined in a way parallel to the project definition in PASS. These project definitions, or PX Forms, carry a unique PX-ID number that indexes through the database. They include a complete title and abbreviation, an abstract of the project that is used for reporting to granting agencies, and typical crystallographic details. Then, for rapid-access visits or mail-in submissions, the user and beamline staff have status logs and an archived 'chat' system for secure two-way communication.

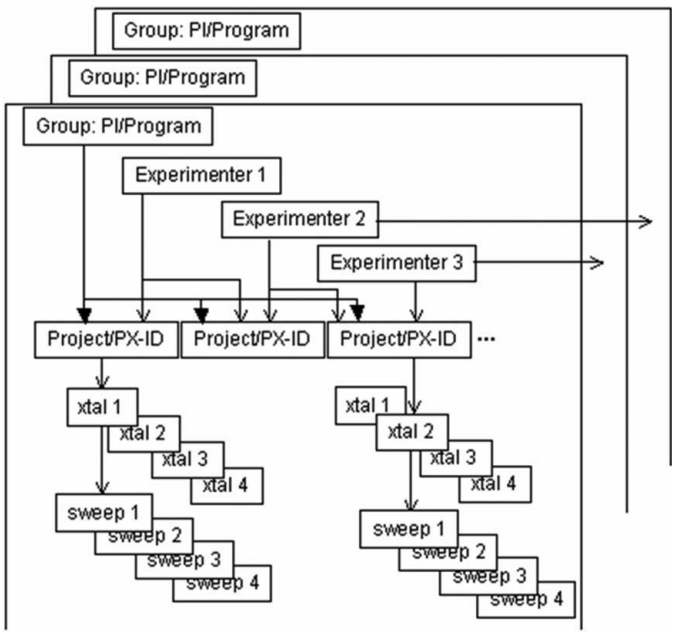

Figure 5
The organization of PXDB. The database is organized by research groups, defined by broad programs, with a principal investigator at the head. There may be several projects associated with each group or program. Individual experimenters will be associated with one or more projects and may be part of several programs. There may be many crystals used for each project. Each data sweep is recorded for each crystal and these need not be performed during the same experimental session.

Of course, the Puck and Cane forms implied in the description of *CBASS* are available in PXDB. The system is used heavily by the beamline staff, especially the mail-in program scientists, to monitor the state of various experiments and there is a range of administrative functions available, hidden to the users, for this work.

Work being performed at all of the PXRR beamlines can be monitored in real time through a Sweep Query tool. Here one finds, for every sweep of data measured, the beamline and other useful information, including a link to the relevant HTML log (Fig. 4), as well as discussion and other links for the project. The discussion links lead to threads in a bulletin-board system for project correspondence and notes, primarily intended to facilitate and record communications between mail-in scientists and their clients. As implied by the name, the Sweep Query is a search interface, providing a variety of criteria for tuning the list of sweeps one wishes to see.

In general, the PXDB is central to the organization of our mail-in program in the way that it organizes work under the authority of a single principal investigator and records the action on every crystal. Its usefulness appears to be growing among outside users.

5. The use of *EPICS* in our control system

To provide a link between *CBASS* and many of the devices in the beamline, mostly motors and scaler/timers, we have adopted *EPICS* (*Experimental Physics and Industrial Control System*). This allows us to take advantage of active developments by a world-wide collaboration (http://www.aps.anl.gov/epics/) of a thoroughly modern device-control architecture distributed as an open source by the University of Chicago. A tremendous advantage to this approach is that the open-source *EPICS* collaboration has provided almost all the driver and device support we have ever required.

EPICS operates as a network-based client/server system. Each 'server' itself drives one or more device and we operate two sorts of server at each beamline. One is a VME-based processor and crate operating under *RTEMS*, driving stepping motors and scalar/timers; the other is a computer running linux that controls analogue and digital IO, serial communication channels and many additional functions. In our case, the only 'client' available to casual users is *CBASS*. We have created special-purpose clients that allow more specialized and simultaneous operation of the devices by the beamline staff to monitor selected devices, to display statuses, to record parameter histories and to debug problems. Operations are both convenient and secure because the servers allow connection for control through a well defined and tightly constrained protocol.

An illustration of the versatility and effectiveness of *EPICS* in the hands of an experienced programmer is the recent rapid development of a gap-control method for a new undulator installed recently at beamline X25 (Tanabe *et al.*, 2006). Our layered *EPICS* construct allowed us to create an energy-changing protocol that would translate wavelength change requests by the *CBASS* user into coordinated mono-

chromator-angle and undulator-gap adjustments. This leaves the NSLS ring operators with priority access to the undulator gap for ring filling and other adjustments.

Although the effort to learn to use *EPICS* was substantial, the rewards we have reaped have proven to be worth it. The ability to incorporate new hardware and devices rapidly, without the need to modify *CBASS*, has sped our controls developments. Additionally, providing a clearly separate domain of control from *CBASS* has allowed beamline controls to be improved and deployed without interfering with user operation or demanding effort to create changes in *CBASS*.

6. System architecture and configuration of *CBASS*

The *CBASS* software is run under Linux at the NSLS. Since Python code is highly portable, it should be possible also to run it under other operating systems. *CBASS* is set up as a client/server system in which the GUI communicates with the *CBASS* control server over TCP/IP sockets (Fig. 6). Multiple clients can communicate with the server. For example, an alternate client was created for NSLS beamline X25, where a Java applet running on a Windows-based laptop runs as a *CBASS* client to facilitate in-hutch goniometer control, while the main *CBASS* GUI runs at the user end-station.

The *CBASS* server is a multi-threaded program with individual threads responsible for processing commands from clients, sending status information to clients, reading the status of beamline motors that are under *EPICS* control and listening for new client connections. One of the strengths of *CBASS* is that it is a relatively small and readable program consisting of roughly 6000 lines of Python code. The server code is modular, with separate Python modules written for diffractometer, detector, beamline, database and robot control. This approach simplifies adapting *CBASS* to different hardware since only small individual modules need to be coded.

At the PXRR beamlines, lower level control is implemented with servers. This was performed to isolate hardware differences within modules such as `goniometer_lib` and `detector_lib`, a strategy that helped us to negotiate gradual upgrading of individual components as we moved away from older hardware that occasionally required the use of outdated operating systems on obsolete computers. By writing servers with identical communication interfaces, we avoid having to change the upper level code if we move or replace hardware.

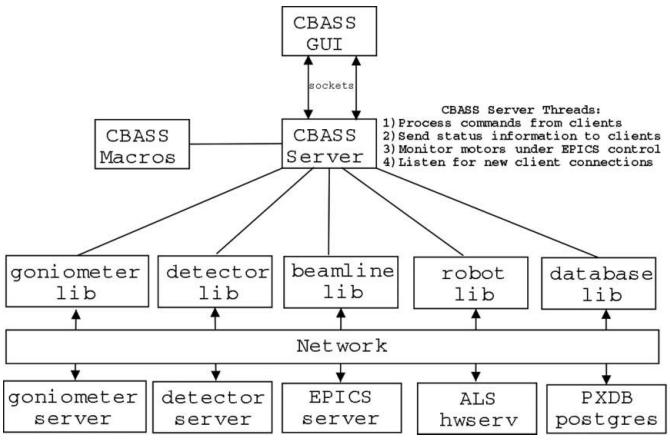

Figure 6
Architecture of *CBASS*.

We can mention a few details about organization. The beamline motors and scalers are served through the *EPICS* channel access server. Database access is accomplished by Python's interface to the PostgreSQL database server. We control the sample-mounting robot by communicating with a robot hardware server written at the Lawrence Berkeley Laboratory (Snell *et al.*, 2004). Beamline-specific routines, such as those used for automatic beam-dump recovery and beam alignments, are found in the `cbass_macros.py` file in the *CBASS* configuration directory local to each beamline. Beamline staff or even users, depending on file permissions, can add or modify routines and load them into the running system by typing 'reload macros' in the *CBASS* command line. The program can be tailored to different configurations, including use of the robot and project-tracking database, through the editing of environment variables set in `cbass_env.txt` in the *CBASS* configuration directory. Most *CBASS* defaults can be set in the `cbass.site` file local to each beamline. This command file sets common flags and parameters, such as whether or not images should be binned or counter intensities should be monitored during data collection.

7. Summary

Only a few years ago, almost all protein crystallography experiments performed at synchrotron beamlines involved a user group coming to the beamline to mount manually and to collect data on several crystals per day that were usually associated with a single project. Higher beam intensities, as well as advances in techniques related to expression, purification, crystallization, and other beneficial fallout from the Human Genome Project, have changed the way many beamlines are used. It is now not unusual for a high-brightness beamline to collect data from dozens of crystals in a day that belong to several different projects from multiple user groups. Automation, such as robotic sample changers and project-management databases, has been developed to cope with the increase in crystals and intensity. The core of this efficiency is good software. The PXRR *CBASS* and PXDB systems have been developed to respond to these demands. We find that their modularity, combined with the relative ease of use of the Python programming language, has resulted in a robust and easily extensible software package for our PX experimental stations.

Financial support comes principally from the Offices of Biological and Environmental Research and of Basic Energy Sciences of the US Department of Energy, and from the National Center for Research Resources of the National Institutes of Health.

References

Beazley, D. (1996). *Proceedings of the Fourth USENIX Tcl/Tk Workshop*, pp. 129–139. Berkeley, CA, USA: USENIX.

Berman, H. M., Westbrook, J., Feng, A., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res.* **28**, 235–242.

McPhillips, T. M., McPhillips, S. E., Chiu, H.-J., Cohen, A. E., Deacon, A. M., Ellis, P. J., Garman, E., Gonzalez, A., Sauter, N. K, Phizackerley, R. P., Soltis, S. M. & Kuhn, P. (2002). *J. Synchrotron Rad.* **9**, 401–406.

Messerschmidt, A. & Pflugrath, J. W. (1987). *J. Appl. Cryst.* **20**, 306–315.

Ravelli, R. B. G., Sweet, R. M., Skinner, J., Duisenberg, A. J. M. & Kroon, J. (1997). *J. Appl. Cryst.* **30**, 551–554.

Robinson, H. H., Soares, S., Becker, M., Sweet, R. & Héroux, A. (2006). *Acta Cryst.* D**62**, 1336–1339.

Skinner, J. M., LaBarca, R. S. & Sweet, R. M. (1996). *Nucl. Instrum. Methods Phys. Res. A*, **383**, 627–630.

Skinner, J. M., Pflugrath, J. W. & Sweet, R. M. (1993). *SHARE 80 Proceedings*, p. 210.

Skinner, J. M. & Sweet, R. M. (1998). *Acta Cryst.* D**54**, 718–725.

Snell, G., Cork, C., Nordmeyer, R., Cornell, E., Meigs, G., Yegian, D., Jaklevic, J., Jin, J., Stevens, R. C. & Earnest, T. (2004). *Structure*, **12**, 537–545.

Sweet, R. M., Skinner, J. M. & Cowan, M. (2001). *Synchrotron Radiat. News*, **14**, 5–11.

Tanabe, T., Ablett, J., Berman, L., Harder, D. A., Hulbert, S., Lehecka, M., Rakowsky, G., Skaritka, J., Deyhim, A., Johnson, E., Kulesza, J. & Waterman, D. (2006). *Proceedings of the Ninth International Conference on Synchrotron Radiation Instrumentation.* In the press.

Ueno, G., Kanda, H., Kumasaka, T. & Yamamoto, M. (2005). *J. Synchrotron Rad.* **12**, 380–384.