Acta Crystallographica Section D
Biological Crystallography
ISSN 0907-4449

JBluIce–EPICS control system for macromolecular crystallography

Sergey Stepanov,[a]* Oleg Makarov,[a] Mark Hilgart,[a] Sudhir Babu Pothineni,[a] Alex Urakhchin,[a]‡ Satish Devarapalli,[a]§ Derek Yoder,[a] Michael Becker,[a] Craig Ogata,[a] Ruslan Sanishvili,[a] Nagarajan Venugopalan,[a] Janet L. Smith[a,b] and Robert F. Fischetti[a]

[a]GM/CA-CAT at the APS, Biosciences Division, Argonne National Laboratory, Argonne, Illinois 60439, USA, and [b]Life Sciences Institute, Department of Biological Chemistry, University of Michigan, Ann Arbor, Michigan 48109, USA

‡ Present affiliation: Citadel Investment Group, Chicago, IL 60603, USA.
§ Present affiliation: WebMD Inc., New York, NY 10011, USA.

Correspondence e-mail: sstepanov@anl.gov

Received 29 October 2010
Accepted 22 December 2010

The trio of macromolecular crystallography beamlines constructed by the General Medicine and Cancer Institutes Collaborative Access Team (GM/CA-CAT) in Sector 23 of the Advanced Photon Source (APS) have been in growing demand owing to their outstanding beam quality and capacity to measure data from crystals of only a few micrometres in size. To take full advantage of the state-of-the-art mechanical and optical design of these beamlines, a significant effort has been devoted to designing fast, convenient, intuitive and robust beamline controls that could easily accommodate new beamline developments. The GM/CA-CAT beamline controls are based on the power of *EPICS* for distributed hardware control, the rich Java graphical user interface of Eclipse RCP and the task-oriented philosophy as well as the look and feel of the successful SSRL *BluIce* graphical user interface for crystallography. These beamline controls feature a minimum number of software layers, the wide use of plug-ins that can be written in any language and unified motion controls that allow on-the-fly scanning and optimization of any beamline component. This paper describes the ways in which *BluIce* was combined with *EPICS* and converted into the Java-based *JBluIce*, discusses the solutions aimed at streamlining and speeding up operations and gives an overview of the tools that are provided by this new open-source control system for facilitating crystallographic experiments, especially in the field of microcrystallography.

1. Introduction

In recent years a number of factors have driven a substantial improvement in control systems for macromolecular crystallography (MX) beamlines (Skinner & Sweet, 1998; Abola *et al.*, 2000; McPhillips *et al.*, 2002; Ueno *et al.*, 2005; Rees *et al.*, 2005; Nurizzo *et al.*, 2006; Skinner *et al.*, 2006; Okazaki *et al.*, 2008; Yamada *et al.*, 2008; Gabadinho *et al.*, 2010). Firstly, many MX beamlines are oversubscribed and continue to experience growing demand. Sophisticated controls allow more efficient use of these beamlines by reducing the required beamtime per experiment. Further increases in efficiency and cost reduction come with remote access and mail-in MX experiments that are tightly dependent on robust automation. Also, users frequently conduct experiments at different facilities, fostering a demand for standardized and friendly interfaces that can be learned quickly. Finally, new capabilities of high-performance MX beamlines, especially in microcrystallography, require sophisticated control functions. A robust, user-friendly and adaptable interface is thus an essential factor contributing to the success of an MX facility.

Recognizing these challenges early on, the General Medicine and Cancer Institutes Collaborative Access Team (GM/CA-CAT) sought to build a control system that would combine a frontend that is easily understood by users and a backend framework that is suitable for the design of fast, easily extensible, distributed beamline controls (Stepanov *et al.*, 2004, 2006). Other requirements of the backend were that it should contain a comprehensive choice of drivers for typical beamline-control electronics, involve a wide collaboration of developers, thus guaranteeing a long lifetime and support, and provide interfaces to a variety of computing languages in order to involve more beamline staff in development. No existing system satisfied all the requirements. However, the SSRL *BluIce* (*Beam Line Universal Integrated Configuration Environment*; Abola *et al.*, 2000; McPhillips *et al.*, 2002) was found to be the best candidate for the frontend owing to its comprehensive and powerful tabbed interface, which had become a de-facto standard in the US MX community and beyond. *EPICS* (*Experimental Physics and Industrial Control System*) was concluded to be the best-suited backend framework because of its excellent record at synchrotron facilities and ongoing wide collaborative development community at APS, DESY, Diamond, ALS, NSLS and other large multi-user facilities (Dalesio *et al.*, 1991; Mooney *et al.*, 1996, 2000). Our task was to combine *BluIce* and *EPICS* in the most efficient way. At the start of the project, *BluIce* and *EPICS* were not interfaced to each other. In this paper, we report the open-source beamline-control system resulted from 'marrying' *BluIce* and *EPICS* and discuss some of the technical solutions that helped to make it fast, efficient for crystallographic data collection and easily extendable.

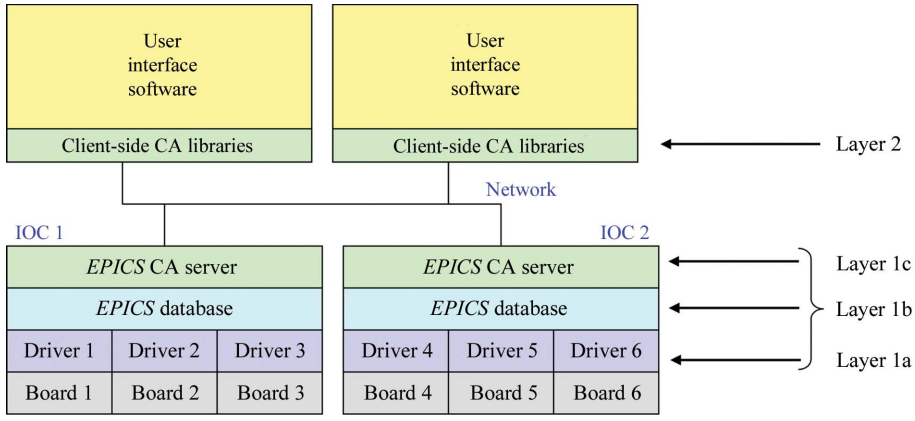

Figure 1
EPICS architecture.

2. Merging the two systems

§§2.1 and 2.2 briefly review the structure and functionality of *EPICS* and SSRL *BluIce*.

2.1. *EPICS* architecture

EPICS is a toolkit for building distributed controls. It provides an easy way for beamline application programs to communicate with distributed electronics on local networks. Fig. 1 illustrates the structure of *EPICS*.

EPICS is based on electronics boards in networked computers called input–output controllers (IOCs). An *EPICS* IOC can be hosted in a VME crate by a Motorola CPU under the VxWorks or RTEMS real-time operating systems, by computers running Windows, Mac OS X or UNIX or even by standalone device controllers with embedded Linux. *EPICS* consists of two major software levels. In the lower level are device drivers for the electronics boards, a database (DB) of multiple records known as process variables (PVs) and a channel-access (CA) server. The top level consists of client-side CA libraries (Fig. 1). The device drivers map the status of devices into the *EPICS* DB, while the CA server allows client software to access the DB remotely *via* the network. An *EPICS* DB represents a hardware abstraction of different electronics devices that imposes a uniform format for communication with any of them. For example, the position of a motor and its state (moving/stopped) are read by client software *via* CA-get `caget` requests to the respective DB fields, while driving the motor to a requested position is initiated *via* a CA-put `caput` command into another DB field. *EPICS* CA also has a callback subscription mechanism to notify clients automatically of a DB field change, eliminating the need for resource-consuming DB polling. In the *EPICS* architecture the client-side software does not need to know either the IP addresses of the IOCs or which IOC controls which hardware because the CA requests are broadcast on local subnets until some CA server acknowledges that the respective PV is under its management. Finally, *EPICS* provides a number of tools to facilitate DB maintenance and application debugging (*e.g.* MEDM, Striptool and caMonitor) and a C-like State Notation Language (SNL) specifically designed to program server applications that run inside IOCs. An additional convenience of *EPICS* for beamline controls is the *synApps* subproject to collect, develop and support *EPICS* drivers for the electronics frequently used at synchrotron-radiation beamlines (Mooney *et al.*, 1996, 2000).

2.2. *BluIce* architecture

SSRL *BluIce* is a beamline-control system developed by the Structural Molecular Biology group at the Stanford Synchrotron Radiation Laboratory (McPhillips *et al.*, 2002). It consists of three software layers as illustrated in Fig. 2: distributed hardware servers (DHS), a central coordination layer called the distributed control system server (DCSS) and a user-interface layer, which is a GUI client of the DCSS that can be launched in multiple instances. Only one GUI instance at a

time can be a master, *i.e.* have the rights to control hardware, while the others run in monitoring mode. The three *BluIce* layers communicate *via* network sockets, although the layers may physically reside on the same computer.

BluIce is a complete (full-scale) control system primarily targeted at macromolecular crystallography experiments. It organizes the many and varied tasks that are part of diffraction experiments, including sample mounting, sample alignment, sample evaluation, fluorescence spectroscopy, data collection, beamline controls and beam optimization. The tabbed organization of the *BluIce* graphical interface (Hutch, Collect, Scan, Screen *etc.*) is easily understood by users. The all-encompassing GUI makes remote operation of beamlines feasible. *BluIce* tabs are written in the Tcl/Tk scripting language, which allows them to be executed under a number of operating systems, including Linux, IRIX, Solaris and Microsoft Windows. The backend layers (the DCSS and DHS) are implemented in C/C++ and the *BluIce* team maintains an effort to keep them portable in terms of operating system. However, the pool of hardware drivers (*i.e.* the DHS) available with *BluIce* is limited. For example, *BluIce* includes drivers for very few of the electronics controlling the GM/CA-CAT beamlines, while most are supported by *EPICS*. Since adding the missing DHS would require considerable and cumbersome low-level programming, we chose to implement *BluIce* communication with *EPICS* IOCs, thus allowing *BluIce* to use all of the *EPICS*-supported hardware.

2.3. *BluIce–EPICS* interface and *JBluIce*

Many considerations have an impact on the transfer of a beamline-control system from one environment to another. We sought to preserve the task-oriented philosophy of the SSRL *BluIce* GUI while connecting it to modern hardware under *EPICS* control. The simplest approach to merging the systems may be based on the similarity of the *EPICS* IOC and the *BluIce* DHS, both of which control devices but differ in protocol: *BluIce* DHS uses socket connections, whereas the *EPICS* IOC is interfaced *via* CA. In our initial merger of *BluIce* and *EPICS*, several '*EPICS* DHS' wrappers interconverted *BluIce* and *EPICS* hardware calls. However, the wrapper approach added an extra networking layer to the existing three layers of *BluIce*, with associated problems of network latencies and of synchronizing servers that operate asynchronously. In addition, the DHS and IOC provide different levels of access to the electronic controllers: the IOC maps every state of the electronic controllers into the client-accessible *EPICS* DB, whereas the DHS suggests a higher level of generic macro commands such as 'move motor from *A* to *B*'. The macro level is convenient for simple operations, but precludes advanced electronics features such as on-the-fly scanning and data acquisition with PMAC controllers, as discussed below. We also encountered problems with conflicting aspects of the *BluIce* and *EPICS* architectures. The *BluIce* DCSS is designed as a single control-system master with sole authority to issue control commands, whereas the liberal *EPICS* architecture allows multiple clients to issue hardware-control commands. Maintaining the DCSS in the *EPICS* environment was problematic because the DCSS needed to be extended with monitors and additional handshaking with the DHS in order to know that other players may change the system state between its own control commands. These considerations argued for a different merger of *BluIce* and EPICS, one that improved speed and versatility and simplified management.

We created a streamlined *BluIce–EPICS* by attaching an *EPICS* API directly to the *BluIce* GUI, eliminating the middle DCSS layer (Fig. 3). In *BluIce–EPICS* the GUI sends and receives messages to the API as in SSRL *BluIce*, but they are converted from socket communications to internal function calls handled by the API within a single application. The API was implemented as a set of C++ classes. The initial implementation of *BluIce–EPICS* included about 50 000 lines of new code and 96 000 lines of code from SSRL *BluIce*. The streamlined architecture replaced the two levels of communications in SSRL *BluIce* (the GUI with the DCSS and the DCSS with the DHS) with one level of *EPICS* channel-access communication. *BluIce–EPICS* thus became a classic *EPICS* client-side program. The *BluIce* exchange protocol was preserved at the GUI level, maintaining compatibility with SSRL *BluIce* for new user-interface developments and allowing code sharing between the two systems.

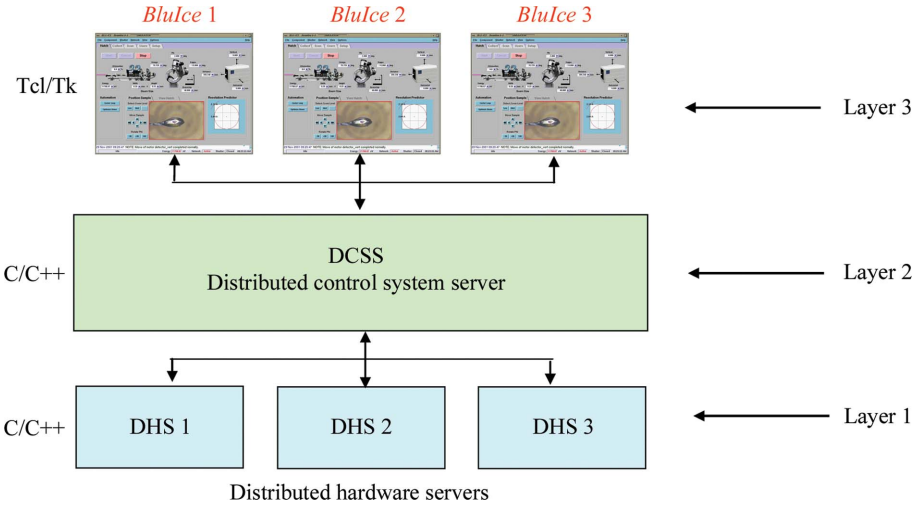

Figure 2
SSRL *BluIce* architecture.

2.3.1. Multiple instances of *BluIce–EPICS*. Two complementary mechanisms were implemented in the new architecture for communication between multiple instances of *BluIce* clients, which are handled by the DCSS layer of SSRL *BluIce*. Firstly, we added communication PVs to the *EPICS* database. Using the *EPICS* callback subscription mechanism, each instance

of *BluIce* can post a message to the DB *via* the CA protocol and the other instances are notified immediately. In addition, a shared MySQL database was created in which all *BluIce* clients initialize their startup configuration and exchange information at run time. As MySQL does not provide a callback mechanism, *EPICS* messages are used to notify clients of any need to re-read MySQL data. This seemingly cumbersome mechanism is a consequence of restrictions on the *EPICS* side: the number and type of *EPICS* PVs are fixed during run time and cannot be changed without rebooting the IOC, while MySQL records can be generated on the fly. A callback-capable DB might be advantageous over MySQL in this case, but the simplicity of MySQL outweighed this consideration.

2.3.2. 'Helper' plug-ins. In order to simplify *BluIce–EPICS* clients, we implemented 'helper' plug-ins (Fig. 4). The overall system was made lighter and more manageable by the use of external applications in lieu of placing all beamline-control functionality in a single program that also handles the GUI. This feature also preserves the key SSRL *BluIce* philosophy of an adaptable architecture in which new functions can easily be developed, tested and implemented. The helpers accomplish complex *BluIce* tasks such as 'mount sample *N* on the goniometer with automounter' or 'perform on-the-fly fluorescence scan'. Simple tasks for which speed is important, such as 'move a motor from *A* to *B*' or 'change beam attenuation', are managed directly by the main application.

The helpers interact with *BluIce–EPICS via EPICS* PVs and some also share MySQL initialization. This is the same mechanism of interaction as between different instances of the *BluIce–EPICS* application. *BluIce–EPICS* includes two types of helpers: permanently running *EPICS* servers and scripts or executables that run temporarily to accomplish a specific task. Each helper is a relatively independent program that can be written in any programming language with an *EPICS* CA library without knowledge of the internals of the main *BluIce* application. This feature considerably widens the base of developers. In our case, the server-type helpers are written in *EPICS* SNL (to run inside an IOC) and also in Perl. The script-type helpers are written in C, Java, Perl and Tcl, representing the expertise of local staff. The helpers make *BluIce–EPICS* a $2\frac{1}{2}$-layer control system that deploys two layers when it commands hardware directly and three layers when it works through its helpers.

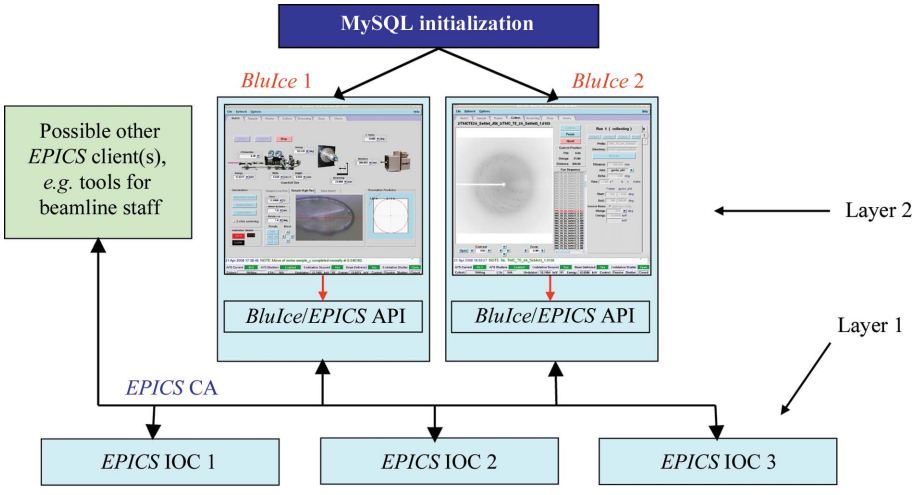

Figure 3
BluIce–EPICS architecture. The streamlined *BluIce–EPICS* applications are standard *EPICS* clients and are equal-rights players on the *EPICS* channel-access network. They can coexist with other *EPICS* clients as shown on the left.

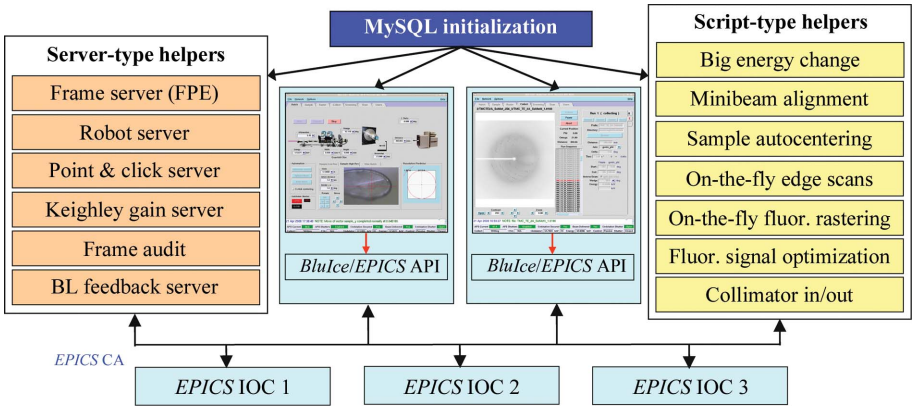

Figure 4
Interaction of *BluIce* clients (two instances shown in the center) with external helpers.

2.3.3. *JBluIce–EPICS*. In parallel with the architectural changes, the SSRL *BluIce* GUI, written in Tcl/Tk and C++, was adapted to GM/CA controls and hardware. This version of *BluIce–EPICS* was used on the GM/CA-CAT beamlines for several years from the start of operations in 2004, where it proved to be both robust and user-friendly. However, adapting the SSRL *BluIce* GUI and the addition of new features revealed a shortage of convenient widgets in Tcl/Tk, which is a relatively small language (Welton, 2010). Therefore, we transitioned the *BluIce–EPICS* GUI from Tcl/Tk and C++ to Java. Three considerations motivated the change. Firstly, the existence of both the C++ and Tcl/Tk languages in the same program complicated debugging owing to the inability of debuggers to cross the language barrier. Secondly, Java is a more productive language owing to its reliable memory management and wide array of high-quality tools such as the Eclipse Integrated Developer Environment (IDE). Thirdly, Java is a widely used language, thus securing continuity of development. The Java transition provided the opportunity to remove Tcl and C++ code that was unused and unneeded owing to the architecture

and hardware differences between the SSRL and GM/CA versions of *BluIce*. Some proprietary network communications used in SSRL *BluIce* were replaced by HTTP provided by the Java standard libraries. The total code reduction was more than threefold, with a new total of 40 000 lines of code. The Java transition also introduced multiple threads to the GUI. Actions that take more than a few milliseconds, including motor moves and script-type helpers, spawn a separate GUI thread, which simplifies tracking the operation. Both Tcl/Tk and Java screens (tabs within the *BluIce* GUI) coexisted in the deployed *BluIce–EPICS* during the gradual transition, which took place over a three-year period without interruption to the GM/CA-CAT user operations. The new control system received the name *JBluIce–EPICS* because of its Java implementation.

2.3.4. Comparison with other control systems. Other modern beamline-control systems, such as *OpenGDA* (Diamond Light Source) and the *BLISS* Framework (ESRF), are designed to be easily adaptable universal controls for a wide range of synchrotron experiments, while *JBluIce* is a dedicated system for crystallography. The more generic *OpenGDA* and *BLISS* have an abstraction layer between the GUI and the hardware servers, which simplifies porting the software to different beamline environments. For example, in *OpenGDA* a GUI client communicates *via* the Corba protocol with one or several Object Servers, which in turn communicate with *EPICS* (Gibbons, 2008). This scheme allows *EPICS* to be replaced by another type of hardware server, *e.g. TANGO*. In *BLISS* a beamline-specific GUI, for example *MxCube*, communicates with *BLISS* hardware objects, which communicate with *TANGO* and *TACO* (Guijarro, 2004; Papillon, 2010; Gabadinho *et al.*, 2010). The scripting capabilities also differ: *OpenGDA* includes an embedded Jython script interpreter and a script editor, which is necessary for generic beamline software. *JBluIce* has a lesser need for scripting because it targets crystallographic experiments, in which the procedures are well established. Nevertheless, *JBluIce* contains a script launcher and much of *JBluIce* functionality relies on supplied scripts, but the scripts are executed as separate processes and can be written in any language supporting *EPICS*.

3. *EPICS*-level solutions

Most parts of the *EPICS* layer in the *BluIce–EPICS* system were available at the start of this project, including the basic infrastructure for booting *EPICS* I/O controllers (Motorola MVME6100 and Linux Virtual IOCs), *EPICS* database managers and the CA servers. A number of standard *EPICS* device drivers were also available, including those for the Struck multi-scaler serving as an X-ray intensity counter, the Canberra multi-channel analyzer (MCA) used with the fluorescence detector, the Acromag ADC and Digital I/O, the Systran and Xycom DACs *etc.* The remaining *EPICS* drivers were developed in-house by modifying, combining or extending the capabilities of existing *EPICS* device interfaces. For example, the *EPICS* CCD detector interface (Rivers, 2004) was updated to the latest remote-interface protocol of the Rayonix CCD detectors and embedded in the *JBluIce–EPICS* Frame-Processing Engine (described below). The *EPICS* driver for the Delta Tau PMAC motion controllers (Coleman, 1995) was extended to work with the 32-axis Turbo PMAC2-VME Ultralite line of controllers, while reducing the respective *EPICS* DB by a factor of ten per axis and supplying a convenient interface for on-the-fly scanning. Finally, the *EPICS* driver for the Hytec 8402 digital-to-analog converter (DAC) controller was upgraded to allow on-the-fly DAC scanning.

Several *EPICS* state-notation servers were implemented. These include the Frame-Processing Engine, which provides collection of diffraction images, the Robot Server, which carries out sample mounting/dismounting operations using the ALS-style pneumatic sample automounter (Snell *et al.*, 2004), the Keithley Server, which controls the parameters of the Keithley 428 current amplifiers, and the Sample-Annealing Server, which controls the sample-cryocooling stream.

3.1. Distributed motion controls with PMAC

Motion controls constitute the core of any beamline-control system. The minimum requirements for a motion controller at a modern crystallography beamline include the following. (i) A capability to move groups of drives synchronously, for example for fast on-the-fly scanning. (ii) An I/O functionality, for example to open and close the shutter at specific angles while rotating a sample on the goniometer. (iii) Support for different types of drives, including servo, stepper and piezo drives. We found that the Delta Tau Turbo PMAC2-VME Ultralite satisfied these requirements fully. The 32-axis controller provides a 2 kHz internal PID loop, a programmable capacity to arrange several drives in up to 16 coordinate systems with user-supplied motion programs and optional digital and analog I/O extension boards. An additional advantage is the distributed architecture, in which 'Macro Stations' are connected to the central unit *via* a fiber loop. Placement of the Macro Stations near motors eliminates the need to lay long cables from the IOC to the drives, makes the layout much cleaner and reduces crosstalk and noise in the encoder and limit-signal lines. Of the VME, PCI and stand-alone networked versions of PMAC, we chose the VME version owing to the availability of a prototype *EPICS* driver: the driver for PMAC1 VME (Coleman, 1995). The use of a single universal type of motion controller created a uniform motion-control environment throughout the beamline from tiny sample piezo positioners to powerful motors supporting the CCD detector. We used the PMAC systems to control as many as 89 motors per beamline.

The challenge of *EPICS* support of such a versatile device as PMAC is choosing what functionality to map into *EPICS*. Our implementation followed the paradigm of Coleman (1995) in which the *EPICS* driver serves as a carrier allowing any *EPICS* PV to link to a PMAC parameter, rather than providing a single *EPICS* motor record for each axis. This constitutes a more flexible control interface compared with a

motor-record approach and is more appropriate for a device as versatile as PMAC. Since each PMAC has thousands of parameters and not all of them are used frequently, the database designer must select which controls to map directly into *EPICS*. We restricted the DB to about 100 PVs per PMAC axis, an approximately tenfold reduction compared with Coleman's PMAC1 interface. These included the requested and actual motor coordinates, status bits for amplifiers and limit switches, coordinate-system status words, motor speeds and a few more. Other parameters that do not require frequent access can be interfaced *via* a dedicated pair of text-string-type *EPICS* PVs, one to send a command to PMAC and the other to receive the controller response. The open-source DB and driver are available for download from the GM/CA CAT website at http://www.gmca.anl.gov/TPMAC2. The PMAC driver was developed in collaboration with the Diamond Light Source, where it is used with dozens of PMAC boards serving as the major motion controllers around the storage ring (Rees *et al.*, 2007; Alcock *et al.*, 2010).

3.2. Fast on-the-fly scanning

A capability for fast on-the-fly scanning, *i.e.* scanning with continuously moving motors, is essential for speedy beamline operations. We implemented an on-the-fly scan functionality using a combination of PMAC and a Struck multiscaler. One of the 32 axes in each PMAC controller acted as a virtual axis and was dedicated as a link to the Struck scaler. The pulse and direction output from the virtual axis were converted into pulse-up and pulse-down signals and fed into two of the 32 Struck inputs (Fig. 5). Before starting a scan, the PMAC is commanded *via EPICS* to link the virtual axis to a coordinate system as a combination of real axes, *e.g.* $v = x$ or $v = (x + y)/2$. The scan involves recording the initial axis position, starting the scaler and starting motion. Data, in the form of intensity *versus* time and motor positions *versus* time, are stored in the scaler input arrays, each with 4K memory. The scaler advance can be based on either the internal clock (fixed time per point) or on the pulses from the virtual motor (fixed step per point). With this simple technique, we have achieved time resolutions as low as 0.5 ms and can profile very fast processes such as opening a fast shutter. It is important to note that although scan preparation is performed in software, which makes it flexible, scanning synchronization is performed solely in hardware so that any latencies are negligible.

The scanning scheme is used in *JBluIce–EPICS* for fluorescence scanning as a function of energy, fluorescence rastering as a function of sample position and various beam-optimization procedures.

3.3. Frame Processing Engine and other state-notation servers

Collecting diffraction frames is the core activity of macromolecular crystallography beamlines. A frame is a diffraction image recorded by an area detector while the sample crystal is rotated on a goniometer through a certain angular range from A to B, where typically $|A - B| = 0.1$–1.0°, over a time interval of 0.5–5.0 s. A hardware-level synchronization of goniometer motion, beamline fast shutter and detector is required in order to measure the frames reproducibly. At GM/CA CAT, the goniometer is an Aerotech ABR1000 air bearing controlled by a PMAC motion controller, the shutter is a fast stepper drive with an opening time of ∼2.5 ms also controlled by PMAC and the detector is a CCD-based Rayonix MX-300. The frame controls are delegated from *JBluIce* to an *EPICS* state-notation server running on the Linux computer attached to the MX-300. This server, named the Frame Processing Engine (FPE), initializes and starts the goniometer and the detector, but the synchronization is performed in PMAC. To start the frame, *JBluIce* maps all frame parameters entered by users into *EPICS* PVs and raises a flag for the FPE (writes '1' into the respective PV). On receiving this flag, the FPE programs the PMAC to open and close the shutter when the goniometer passes angles A and B, respectively, and then sends a 'start acquiring' command to the CCD using the socket interface (Rivers, 2004) updated to the latest Rayonix remote protocol, sets the goniometer speed to match the requested frame duration and starts the goniometer rotation from $A - \delta$ to $B + \delta$, where δ is a small interval allowing the goniometer to accelerate before reaching angle A and to decelerate after passing angle B.

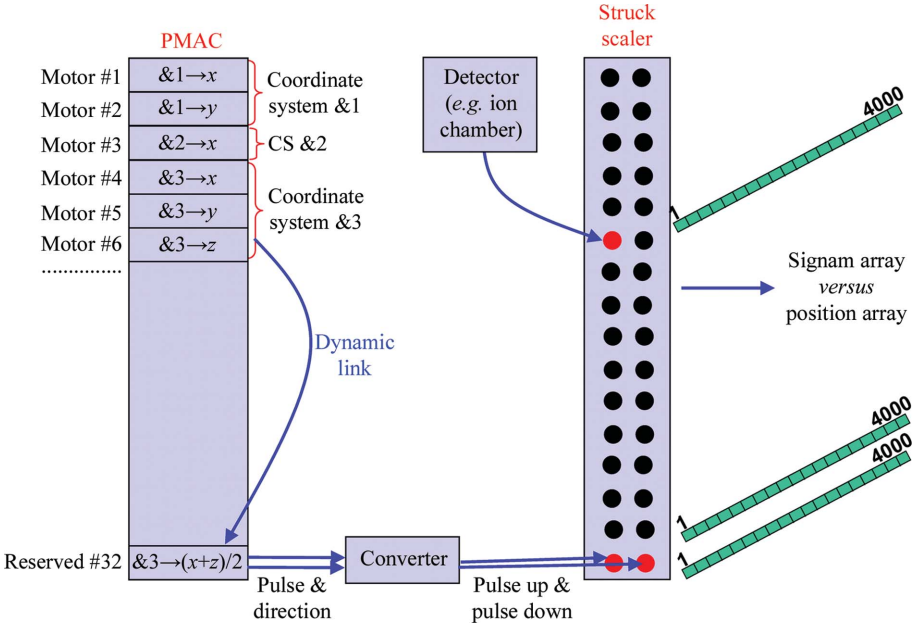

Figure 5
Schematic implementation of on-the-fly scanning with PMAC controller and Struck scaler.

Since the PMAC runs an internal PID with 2 kHz frequency, the jitter of opening and closing the shutter should not exceed 0.5 ms. After the shutter is closed, the FPE sends a 'readout' command to the CCD, restores the goniometer speed and rewinds the goniometer to $B - \delta$.

In addition to the FPE, *JBluIce* interacts with another *EPICS* server called Frame Audit. This server, which is independent of the FPE, monitors the same 'start frame' flag and on receiving it triggers the Struck multiscaler. The Struck multiscaler records X-ray intensity downstream of the shutter as a function of time. Frame Audit integrates the intensity over time, which can be used to calculate the radiation dose received by the sample during the frame collection, and provides the information to *JBluIce via* a PV. The intensity profile recorded by the multiscaler has a rectangular shape, with the FWHM corresponding to the actual shutter-opening interval. This FWHM is analyzed by Frame Audit and in the rare case of any deviation, *e.g.* if the goniometer speed is non-uniform owing to a misbehaving goniometer motor, an error message is sent to *JBluIce via* a PV and the beamline staff are notified by e-mail.

With small modifications, the FPE can be applied to other hardware. For example, when the CCD or shutter needs to be triggered *via* a TTL pulse, one can use the PMAC digital I/O extension board. However, fast detectors that allow shutterless data collection of multiple frames per second may require a different solution.

Other *EPICS* servers were designed using the same approach: the commands, parameters and responses are passed between servers and *JBluIce via* PVs. For example, the Robot Server (Makarov *et al.*, 2007), which supports the Berkeley-style automounter, receives string commands of the type 'load A 7' or 'unload', where 'A' and '7' identify the position of a sample in the robot Dewar. To execute these commands, the server coordinates multiple pneumatic actuators and Dewar motions of the robot *via EPICS* commands to digital I/O and the PMAC motion controller. Some controls, such as processing of beam-attenuation requests and clicks on the video to move samples, have both internal *JBluIce* and external server implementations. This is a manifestation of the flexibility of our design, in which multiple solutions are possible and the choice of implementation depends on convenience and local expertise. On the whole, using external servers for complex operations makes the *JBluIce* structure modular and simplifies the task of porting *JBluIce* to another facility.

3.4. Operations tracking and error handling

The *JBluIce–EPICS* software provides an extensive set of facilities for operations tracking and error reporting. These facilities are embedded into each instance of the GUI as well as into the helpers. Each operation is tracked to the extent permitted by hardware. The use of advanced motion controllers, such as Turbo PMAC, facilitates tracking capabilities because the controllers track and report to *EPICS* any subsequent errors in encoded motions, amplifier faults and the status of limit switches. The tracking messages are printed in the log window located at the bottom of each *JBluIce* tab. They are color-coded as in SSRL *BluIce* (green for info, gray for warnings and red for errors). All helper scripts and servers contribute to this log by writing into a 256-character *EPICS* string PV, which is monitored by *JBluIce* and copied to the log window upon any change. All stdout and stderr output of helper scripts is also forwarded to the log. The full copy of the log for each *JBluIce* execution is stored in a dated text file that can be used for post-analysis. Given the size of these files, they are compressed after closing. The server-type helpers, such as FPE and Robot Server, keep additional detailed logs. As explained above, the *JBluIce* Frame Audit process monitors data collection, contributes to the log and reports suspicious data frames to beamline support staff by email. In addition to the extensive logging, critical errors are reported *via* popup dialogs that require user acknowledgement.

4. *JBluIce* controls for macromolecular crystallography

JBluIce–EPICS functionality at the user level is organized as tabs on the *JBluIce* GUI, preserving the task-oriented philosophy of SSRL *BluIce* (McPhillips *et al.*, 2002). Eight tabs are available: Hutch, Sample, Collect, Scan, Raster, Screening, Log and Users. The Hutch, Collect, Scan and Users Tabs resemble those of SSRL *BluIce*, while the others are specific to new functionality developed within *JBluIce*. Unlike SSRL *BluIce*, there is no Setup tab because all staff functions for setting up the beamline were moved out of *JBluIce* to standard *EPICS* clients (MEDM, Striptool *etc.*) combined with beamline-specific automation scripts. The *JBluIce* tabs are written in Java using Eclipse RCP (Eclipse, 2004) and they communicate with *EPICS* using CAJ, which is the *EPICS* Channel Access API for Java (COSYLAB, 2004).

4.1. Hutch tab

The Hutch tab contains controls for general parameters of the beamline and end station (Fig. 6). Users can change the X-ray beam energy and attenuation, set the size of collimating slits, position the detector, set the beamstop distance, rotate and center the crystal on the goniometer *etc*. They can also control several video cameras, including a high-resolution on-axis sample-visualization camera, a lower resolution sample-visualization camera pointing at the sample from below and a camera surveying the end station.

Although these controls look similar to those on the Hutch tab of SSRL *BluIce*, there are essential differences behind the scenes. Two examples of differences also illustrate *JBluIce–EPICS* organization. X-ray energy changes at our undulator beamlines require changes not only to the monochromator Bragg angle but also to the undulator gap and potentially to the undulator active harmonic and reflective lanes of the focusing and deflecting mirrors. Relevant parameters and calibration values are taken from a lookup table stored in the MySQL database. The beam attenuation is preserved by calculating the best-matching combination of attenuating foils

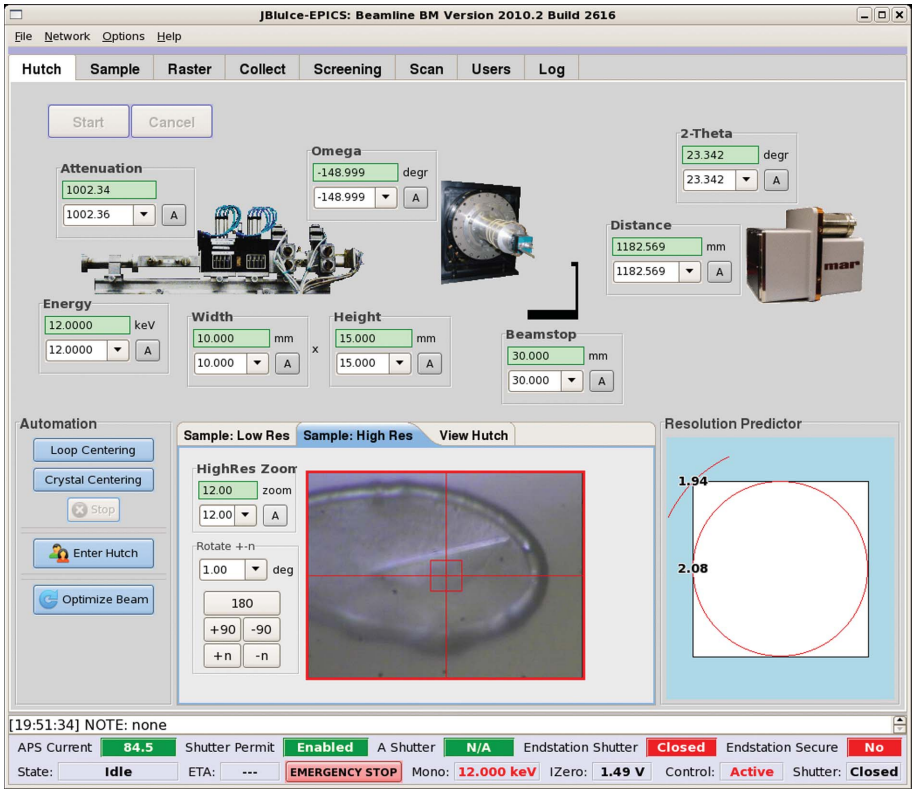

Figure 6
JBluIce Hutch tab.

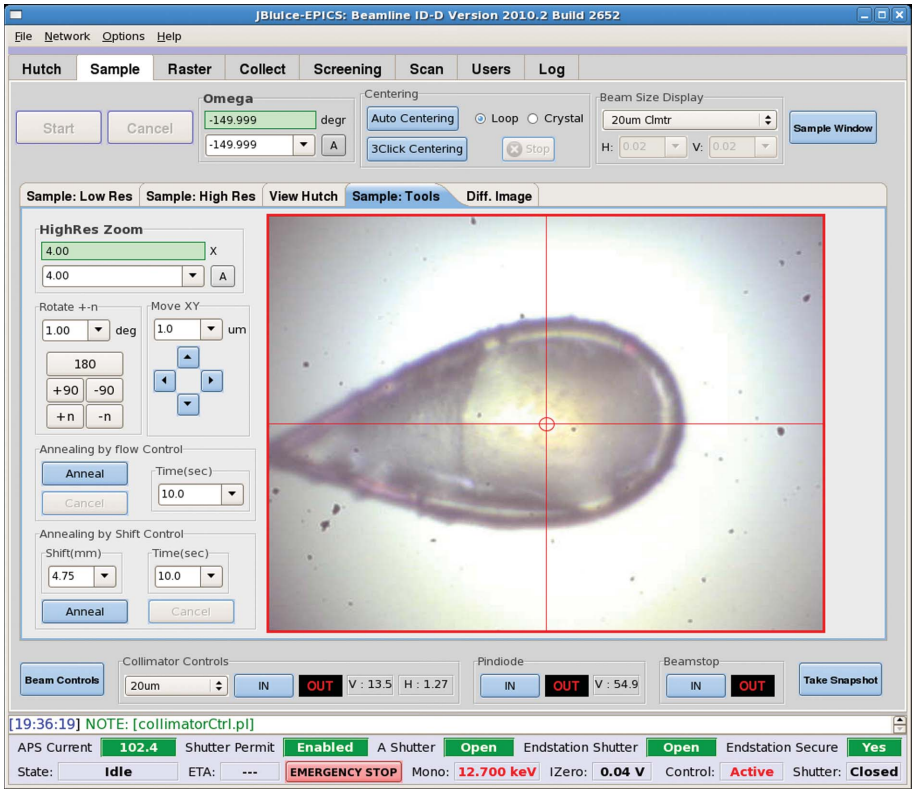

Figure 7
JBluIce Sample tab.

for the new beamline energy. For large energy changes, when the monochromator Bragg angle alters by more than 1°, the beamline is re-optimized by automatically putting into the beam a small pinhole with a PIN diode behind it and scanning the beam in two directions with the help of DAC-controlled piezo actuators on the horizontal and vertical focusing mirrors. The whole operation is outsourced to a helper script so that a facility-specific procedure can be substituted as needed.

The second example is the calculation of X-ray beam attenuation. At SSRL the attenuating foils were made of Al of geometrically progressing thickness, which allowed easy determination of the best foil combination for the requested attenuation. Also, attenuation by a single foil was calculated using a simple formula for the dependence of foil absorption on X-ray energy, which was appropriate for the SSRL X-ray energy range. At the GM/CA-CAT beamlines, which are designed for a wide energy range from 3.5 to 35 keV, the foil set is five Ag and 11 Al foils of nonregularly progressing thickness. To find the best combination of foils for the X-ray energy and requested attenuation, *JBluIce* applies a simplex algorithm and the absorption from a single foil is calculated by interpolating data from the tables of Henke *et al.* (1993) and Brennan & Cowan (1992).

The video from the low-resolution and hutch-surveillance cameras is provided by an interface to AXIS networked video servers (as at SSRL), while the high-resolution video is from a 5 Mpixel IQeye networked video camera. The loop- and crystal-centering buttons spawn a helper script that interfaces with the *XREC* autocentering package (Pothineni *et al.*, 2006). The optimize-beam button calls another helper to execute a beam-position optimization script that is also used after large energy changes.

4.2. Sample tab

The Sample tab (Fig. 7) is an addition to the Hutch tab created in response to requests by users. It gathers in one place the controls of beamline components

around the sample, provides a larger window for video streaming and most notably is detachable so that users can have a large live video of the sample and sample controls.

A highlight of the Sample tab is the collimator control, which allows one-click switching between the 5, 10 and 20 µm pinholes or a 300 µm scatter guard in the quad minibeam collimator installed at the GM/CA-CAT beamlines (Sanishvili *et al.*, 2008; Fischetti *et al.*, 2009). The pre-aligned positions for each pinhole are stored in *EPICS* PVs and upon a click on the beam-size pull-down menu they are sent to the PMAC motion controller driving the collimator in vertical and horizontal directions.

The Anneal controls allow temperature annealing of the sample by either temporarily closing the cryostream through a command to the respective state-notation server or temporarily moving the sample out of the cryostream.

The Beam Controls button opens a popup window with two sliders for vertical and horizontal steering of the X-ray beam by changing the voltage on the piezo positioners of the focusing mirrors. The collimator, PIN diode, beamstop and backlight in/out controls are implemented in a single helper that prevents collisions of these components with each other. In addition, the CCD detector is protected from accidental radiation damage by a low-level *EPICS* PV lock that blocks shutter opening when neither the beamstop nor the PIN diode is in the beam. The three-click centering button launches a helper for sample centering that uses an algorithm developed at the ESRF (Gabadinho *et al.*, 2010). Briefly, the vertical coordinate of the crystal on the screen is presented as $y = r\cos(\varphi - \varphi_0) + y_0$, where r is the radius of sample deviation from the rotation axis, φ is the goniometer angle, φ_0 is the angle of maximum deviation and y_0 accounts for a shift of the rotation axis with respect to the crosshair. The three clicks determine r, φ_0 and y_0 and hence the vertical shift needed to center the sample on the rotation axis. The horizontal shift is taken as the average horizontal coordinate of the three clicks.

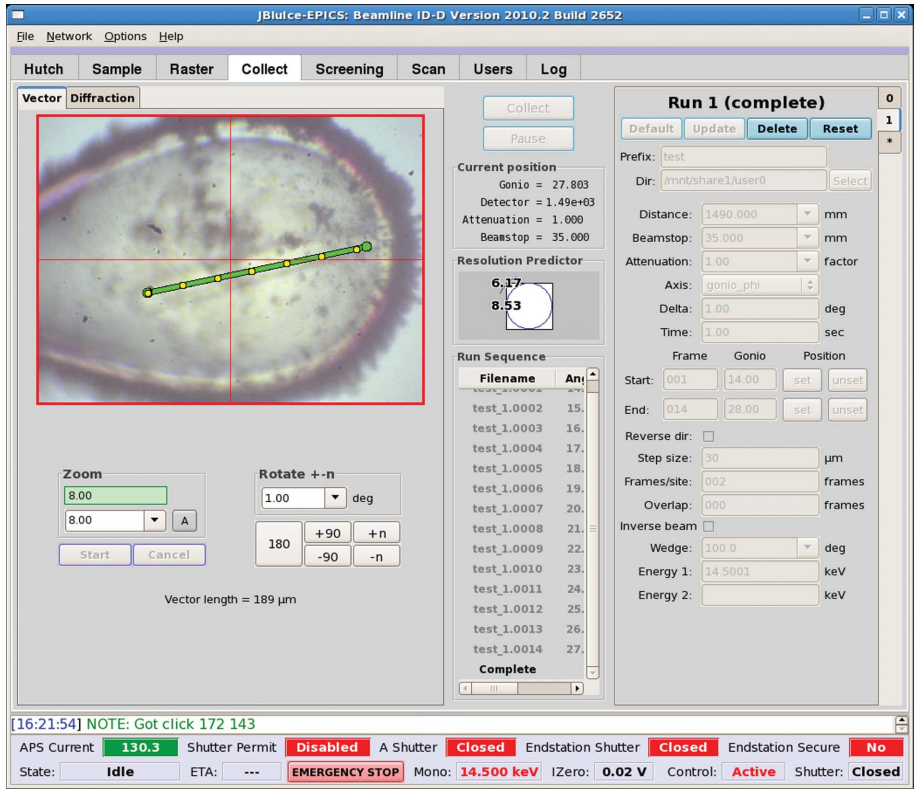

Figure 8
JBluIce Collect tab with controls for data collection along a vector.

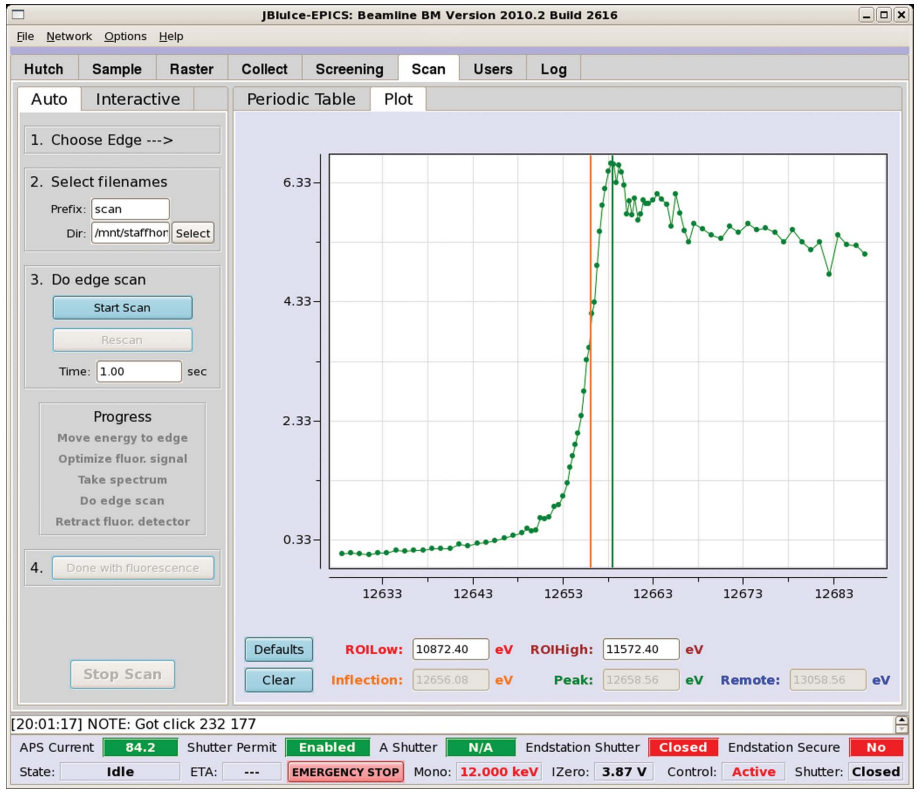

Figure 9
JBluIce Scan tab with controls for automatic scanning.

4.3. Collect tab

The Collect tab (Fig. 8) contains all the controls available in SSRL *BluIce* with the single exception of dose mode, plus the possibility of collecting data along a three-dimensional vector in the sample. Users can set frame sequences by specifying exposure time, starting goniometer angle, the angular width of each frame ('delta') and either the number of frames or the end angle. There are also options for 'wedge' and 'inverse-beam' data collection. Up to 16 sequences can be specified by adding sub-tabs at the right pane. For convenience, the detector and beamstop distances, beam attenuation and resolution predictor are also provided on this tab. When data collection starts, each frame is processed by the FPE as discussed in §3.3.

The image panel on the Collect tab displays either diffraction images as they are collected or the high-resolution camera view for setting up vector data collections (*i.e.* along the green line in Fig. 8). The option to collect data along a vector helps to reduce the effects of radiation damage. With this option, the user defines the ends of a three-dimensional vector, the distance between intermediate points along the vector and the number of diffraction images per point. Movement of the crystal along the vector is managed internally by *JBluIce*, although it also could be outsourced to a helper.

4.4. Scan tab

The Scan tab (Fig. 9) provides a means of recording fluorescence from a sample as a function of X-ray energy, typically in order to find the peak and inflection points of an absorption edge for multi-wavelength anomalous diffraction (MAD). The tab includes options for automatic or interactive scanning.

In the automatic mode, the user selects the absorption edge of interest on the periodic table sub-tab. The edge energies and scan intervals are retrieved from the MySQL database, the beamline is re-tuned to the X-ray energy of the requested absorption edge, the fluorescence detector is moved close to the sample, the beam attenuation is set to the maximum value for which a fluorescence signal is measurable (to minimize radiation damage to the sample), the fluorescence spectrum is recorded using the multi-channel analyzer (MCA), the region of interest (ROI) on the MCA is set to the theoretical fluorescence peak center and FWHM for the selected absorption edge and the spectrum is displayed in the *JBluIce* plot window for the user to check the ROI and intervene if needed. A helper script then runs the three-band scan with fine energy sampling at the central interval around the absorption edge and coarse sampling at the tails. In the interactive scanning mode the user performs each step explicitly. This mode also offers manual selection of the ROI on the fluorescence spectrum, which is especially helpful for infrequent samples with several close fluorescence lines or with unknown fluorescent atoms.

Both modes employ on-the-fly scanning, using the algorithm described in §3.2 with the addition that the signal from the fluorescence detector is discriminated according to the MCA ROI and fed directly into the Struck multiscaler, *i.e.* bypassing the MCA. In the GM/CA-CAT environment, on-the-fly scanning is about 3.5 times faster than step scanning, which is also available as an option. Scan data are automatically sent to the *CHOOCH* software (Evans & Pettifer, 2001) for calculation of the peak and inflection points. In addition, the scan may operate in an adaptive mode (currently turned on and off by staff only *via* the MySQL database) in which a first coarse pass determines the rough location of the absorption edge and a fine-step pass is then carried out around the edge, allowing automated handling of chemical shift of the edge. This is an example of how new features can be developed and made available to expert users for testing.

4.5. Raster tab

One of the main benefits of the GM/CA-CAT beamlines is the ability to target micrometre-sized crystals using easily exchangeable 5, 10 and 20 µm mini-beams (Fischetti *et al.*, 2009). However, the optical centering of small crystals is challenging since visible-light wavelengths (0.4–0.7 µm) are comparable to the crystal size. Larger crystals are also sometimes difficult to see because of parallax effects in the surrounding mother liquor. Finally, many crystals have inhomogeneous diffraction quality, which cannot be addressed by optical centering. To address these challenges, in 2008 we introduced a Raster tab where crystals can be scanned in the X-ray beam on a user-specified grid (Fig. 10).

Two modes of rastering, diffraction-based and fluorescence-based, have been implemented. In the diffraction-based rastering mode, an early version of which was described by Cherezov *et al.* (2009), the software carries out a two-dimensional step scan of the sample on the goniometer and at each step a diffraction image is taken with the help of the FPE. The images are sent to the program *DISTL* (Zhang *et al.*, 2006) for locating and counting diffraction spots and the scored results are presented in a table. By inspecting this table or examining the diffraction images, users can select optimal areas within the macroscopic sample. The multi-tabbed interface on the Raster tab allows up to 16 raster scans to be saved for subsequent retrieval. Similar diffraction-rastering functionalities have also been developed at other synchrotron facilities (Aishima *et al.*, 2010; Bowler *et al.*, 2010).

The fluorescence rastering mode has the advantage of lower radiation damage, since about 30-fold higher beam attenuation can be used compared with diffraction rastering, and fourfold faster operation, since on-the-fly scanning is applied. One can request to measure fluorescence from a particular chemical element or count any fluorescence. Fluorescence rastering is complementary to diffraction rastering because it cannot address inhomogeneities in diffraction quality. However, it is very helpful for finding certain small crystals.

4.6. Screening tab

The Screening tab (Fig. 11) is designed for automatic screening of multiple crystals with or without the sample automounter. The *JBluIce* Screening tab was developed *de novo* because SSRL *BluIce* did not have this capability at the

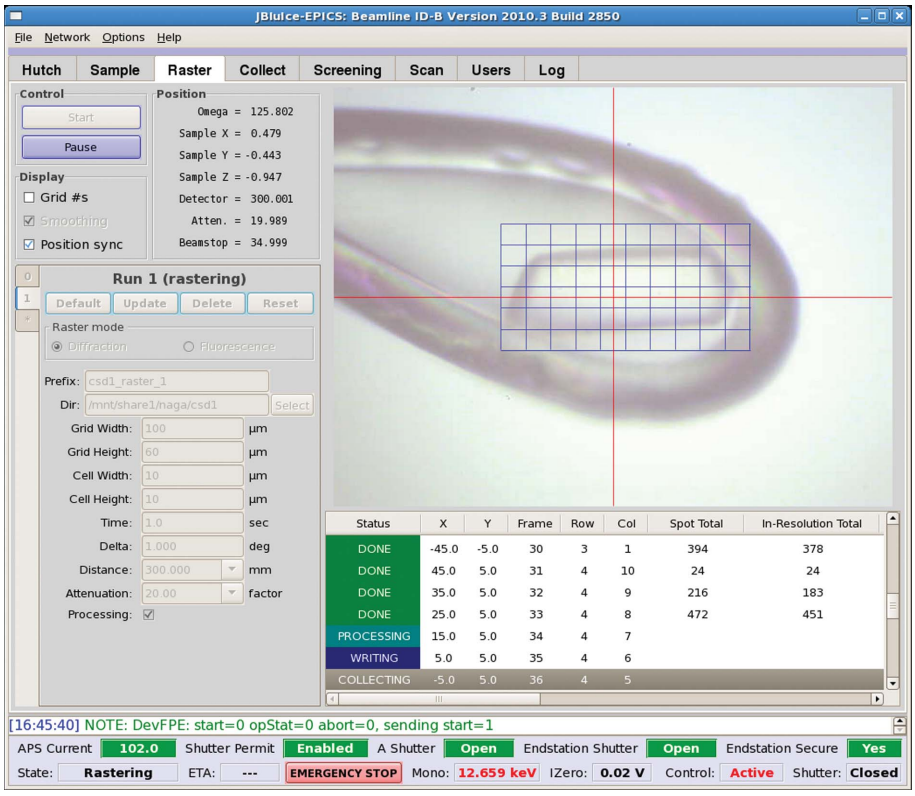

Figure 10
JBluIce Raster tab in the diffraction rastering mode.

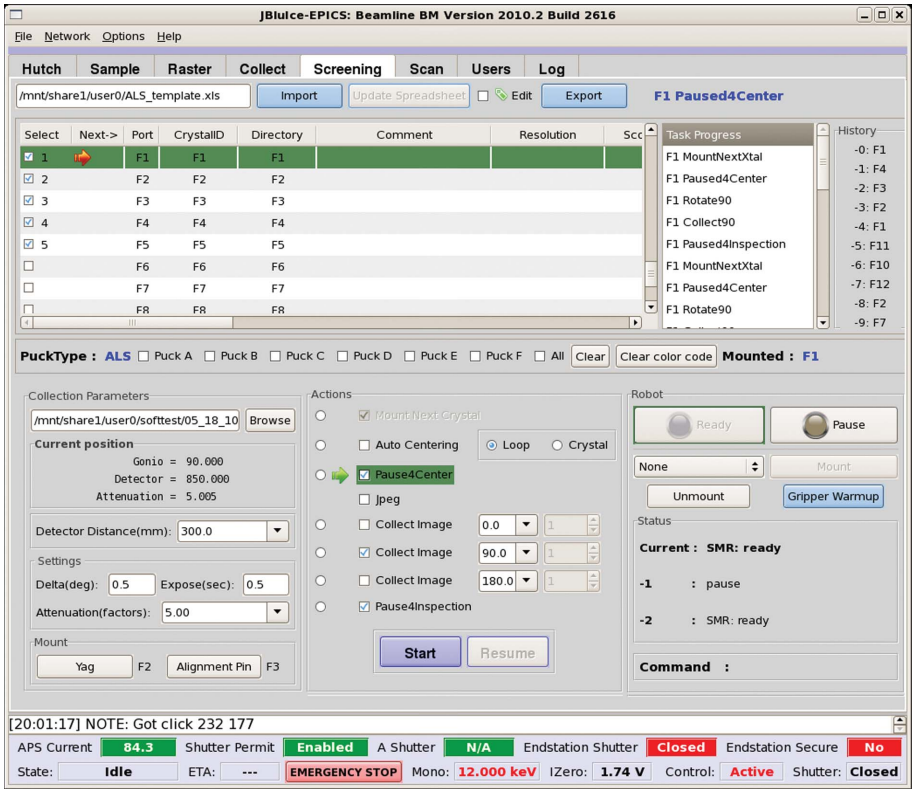

Figure 11
JBluIce Screening tab.

time that our software was forked. Later versions of SSRL *BluIce* v.5 offer automatic screening (Soltis *et al.*, 2008; Smith *et al.*, 2010). In the typical operation of the Screening tab, samples arrive in ALS-style, Uni-style or Rigaku-style pucks along with a respective description in an *Excel* spreadsheet. The pucks are loaded into the automounter Dewar and the spreadsheet is imported into *JBluIce*. The user selects which crystals to screen in what order and chooses the set of operations, including mounting with the automounter, autocentering or pausing for manual centering, grabbing a JPEG image from the video camera and the collection of up to three diffraction images at different goniometer angles (typically 0, 45 and 90°). The Screening tab can optionally send collected diffraction images to the *WebIce* software (González *et al.*, 2008) for scoring, autoindexing and strategy calculations. The results are displayed in the spreadsheet table, which can be exported in *Excel* format. The spreadsheet is editable in *JBluIce*, allowing users to add notes. Automatic sample mounting is provided by the Robot Server and the diffraction images are taken by the FPE (§3.3). This modular design allows the easy substitution of different hardware.

4.7. Users tab, Log tab and remote access

The Users tab shows all instances of *JBluIce* currently running, including the computer name, user account name and process ID for each instance and which *JBluIce* is currently a master. The Log tab generates a log file with all beamline settings and optional user notes. These parameters, along with I_0 and active-beamstop intensities, are also logged into a text file whenever a data set is collected in the Collect tab. Automatic continuous logging is planned for the future, along with a database backend and web services for convenient access to saved logs.

JBluIce does not directly target remote beamline operation. Remote access was left outside the development scope because most facilities have adopted the remote-desktop approach

using software such as NOMACHINE NX for remote presence. However, we planned for use of *JBluIce via* a remote desktop. For example, the video refresh rate is set very low (∼1 fps) while the sample is resting and is increased to 5 fps while it moves. We also placed several controls along the bottom bar of all tabs (Figs. 6–11), including storage-ring current or shutter permit, which local users can see on monitor screens. In addition, multiple safety features are also implemented with remote access in mind, such as a safety mat blocking any automounter, CCD detector or goniometer motions if anyone stands near the goniometer.

5. Conclusions

An innovative data-acquisition system for macromolecular crystallography has been implemented. It combines the classic user interface refined at SSRL over many years with powerful industry-best solutions such as *EPICS*, Java and Eclipse. The two-layer architecture makes the deployment of any advanced hardware features fast and easy. The use of completely independent plugins that can be written in any language and launched as standalone applications makes the system easily extensible.

The task-oriented philosophy embodied in the tabbed layout of SSRL *BluIce* corresponds to a typical workflow at a crystallography beamline. The user sees all controls for a particular task on one screen and therefore spends maximal time on the experiment and minimal time switching between screens, searching for proper buttons and mouse-clicking through a hierarchy of commands. Because this philosophy was so successful at SSRL, we retained it throughout *JBluIce–EPICS*. Users who are new to GM/CA CAT typically begin productive experiments immediately following a brief introduction by a staff member. Throughout the development period the addition of features was driven by close interactions with users. We collected, analyzed and scored user suggestions and implemented many of them. For example, users have been extremely enthusiastic about the *JBluIce* Raster tab (§4.5), which, together with the user-selectable mini-beam, facilitates data collection from small or imperfect crystals. The detachable Sample tab (§4.2), which was added at user request, addresses the issue that some controls should be convenient to all operations but reproducing them on each task tab would make the interface over-populated. The present control system is well settled and we do not plan significant changes in the near term. In addition to improvements inspired by user suggestions and new features of automation, we plan to implement a pipeline to automated structure determination, adapt the system to faster detectors and develop controls for submicrometre crystallography.

We thank Peter Kuhn (Scripps), Scott McPhillips (SSRL), Thomas Earnest and Carl Cork (ALS), Mark Rivers (University of Chicago) and Tim Mooney (APS) for helpful discussions and all GM/CA CAT users for providing valuable feedback and ideas. GM/CA CAT is supported in whole or in part by Federal funds from the National Cancer Institute (Y1-CO-1020) and the National Institute of General Medical Sciences (Y1-GM-1104). Use of the Advanced Photon Source was supported by the US Department of Energy, Basic Energy Sciences, Office of Science under contract No. DE-AC02-06CH1135.

References

Abola, E., Kuhn, P., Earnest, T. & Stevens, R. C. (2000). *Nature Struct. Biol.* **7**, 973–977.
Aishima, J., Owen, R. L., Axford, D., Shepherd, E., Winter, G., Levik, K., Gibbons, P., Ashton, A. & Evans, G. (2010). *Acta Cryst.* D**66**, 1032–1035.
Alcock, S. G., Ludbrook, G. D. & Sawhney, K. J. S. (2010). *Synchrotron Radiat. News*, **23**, 25–30.
Bowler, M. W., Guijarro, M., Petitdemange, S., Baker, I., Svensson, O., Burghammer, M., Mueller-Dieckmann, C., Gordon, E. J., Flot, D., McSweeney, S. M. & Leonard, G. A. (2010). *Acta Cryst.* D**66**, 855–864.
Brennan, S. & Cowan, P. L. (1992). *Rev. Sci. Instrum.* **63**, 850–853.
Cherezov, V., Hanson, M. A., Griffith, M. T., Hilgart, M. C., Sanishvili, R., Nagarajan, V., Stepanov, S., Fischetti, R. F., Kuhn, P. & Stevens, R. C. (2009). *J. R. Soc. Interface*, **6**, S587–S597.
Coleman, T. (1995). Personal communication.
COSYLAB (2004). *Channel Access in Java – Channel Access for Java.* http://caj.cosylab.com.
Dalesio, L. R., Kraimer, M. R. & Kozubal, A. J. (1991). *Proceedings of the International Conference on Accelerators and Large Experimental Physics Control Systems*, edited by C. O. Pac, S. Kurokowa & T. Katoh, pp. 278–282. Tsukuba, Japan: KEK.
Eclipse (2004). *Rich Client Platform.* http://www.eclipse.org/home/categories/rcp.php.
Evans, G. & Pettifer, R. F. (2001). *J. Appl. Cryst.* **34**, 82–86.
Fischetti, R. F., Xu, S., Yoder, D. W., Becker, M., Nagarajan, V., Sanishvili, R., Hilgart, M. C., Stepanov, S., Makarov, O. & Smith, J. L. (2009). *J. Synchrotron Rad.* **16**, 217–225.
Gabadinho, J. *et al.* (2010). *J. Synchrotron Rad.* **17**, 700–707.
Gibbons, P. (2008). *NOBUGS*2008, abstract 159. http://www.nbi.ansto.gov.au/nobugs2008.
González, A., Moorhead, P., McPhillips, S. E., Song, J., Sharp, K., Taylor, J. R., Adams, P. D., Sauter, N. K. & Soltis, S. M. (2008). *J. Appl. Cryst.* **41**, 176–184.
Guijarro, M. (2004). *NOBUGS*2004, abstract 65. http://lns00.psi.ch/nobugs2004.
Henke, B. L., Gullikson, E. M. & Davis, J. C. (1993). *At. Data Nucl. Data Tables*, **54**, 181–342.
Makarov, O. A., Benn, R., Corcoran, S., Devarapalli, S., Fischetti, R., Hilgart, M., Smith, W. W., Stepanov, S. & Xu, S. (2007). *Nucl. Instrum. Methods A*, **582**, 156–158.
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.
Mooney, T. M., Arnold, N. D., Boucher, E., Cha, B. K., Goetze, K. A., Kraimer, M. R., Rivers, M. L., Sluiter, R. L., Sullivan, J. P. & Wallis, D. B. (2000). *AIP Conf. Proc.* **521**, 322–327.
Mooney, T. M., Cha, B. K., Goetze, K. A., Reid, D. R. & Winans, J. R. (1996). *Rev. Sci. Instrum.* **67**, 3369–3373.
Nurizzo, D., Mairs, T., Guijarro, M., Rey, V., Meyer, J., Fajardo, P., Chavanne, J., Biasci, J.-C., McSweeney, S. & Mitchell, E. (2006). *J. Synchrotron Rad.* **13**, 227–238.
Okazaki, N., Hasegawa, K., Ueno, G., Murakami, H., Kumasaka, T. & Yamamoto, M. (2008). *J. Synchrotron Rad.* **15**, 288–291.
Papillon, E. (2010). *NSLS-II Workshop on Data Acquisition and User Interfaces, Upton, NY.* http://www.bnl.gov/nsls2/workshops/041910_DAC_UserInterface.asp.

Pothineni, S. B., Strutz, T. & Lamzin, V. S. (2006). *Acta Cryst.* D**62**, 1358–1368.

Rees, N. P., Denison, P. N. & Cobb, T. M. (2007). *International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS), Knoxville, Tennessee, USA*, pp. 108–110. http://accelconf.web.cern.ch/AccelConf/ica07/PAPERS/TPPA10.PDF.

Rees, N. P., Pulford, W. C., Norbury, M. A., Leicester, P. J., Jones, E. L., Heron, M. T. & Denison, P. N. (2005). *International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS), Geneva, Switzerland.* Poster PO1.048-6. http://accelconf.web.cern.ch/AccelConf/ica05/proceedings/pdf/P1_048.pdf.

Rivers, M. (2004). *synApps: ccd.* http://cars9.uchicago.edu/software/epics/ccd.html.

Sanishvili, R., Nagarajan, V., Yoder, D., Becker, M., Xu, S., Corcoran, S., Akey, D. L., Smith, J. L. & Fischetti, R. F. (2008). *Acta Cryst.* D**64**, 425–435.

Skinner, J. M., Cowan, M., Buono, R., Nolan, W., Bosshard, H., Robinson, H. H., Héroux, A., Soares, A. S., Schneider, D. K. & Sweet, R. M. (2006). *Acta Cryst.* D**62**, 1340–1347.

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

Smith, C. A., Card, G. L., Cohen, A. E., Doukov, T. I., Eriksson, T., Gonzalez, A. M., McPhillips, S. E., Dunten, P. W., Mathews, I. I., Song, J. & Soltis, S. M. (2010). *J. Appl. Cryst.* **43**, 1261–1270.

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.

Soltis, S. M. *et al.* (2008). *Acta Cryst.* D**64**, 1210–1221.

Stepanov, S., Makarov, O., Urakhchin, A., Devarapalli, S., Yoder, D. & Fischetti, R. (2006). *NOBUGS*2006, pp. 31–32. http://nobugs2006.lbl.gov.

Stepanov, S., Makarov, O., Urakhchin, A., Schwabe, U., Venkataraman, C. & Pugliese, R. (2004). *NOBUGS*2004. http://lns00.psi.ch/nobugs2004.

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

Welton, D. (2010). *Where Tcl and Tk Went Wrong.* http://journal.dedasys.com/2010/03/30/where-tcl-and-tk-went-wrong.

Yamada, Y., pHonda, N., Matsugaki, N., Igarashi, N., Hiraki, M. & Wakatsuki, S. (2008). *J. Synchrotron Rad.* **15**, 296–299.

Zhang, Z., Sauter, N. K., van den Bedem, H., Snell, G. & Deacon, A. M. (2006). *J. Appl. Cryst.* **39**, 112–119.