Choosing an instrument control system for ANSTO

Nick Hauser, *Andy Gotz

Abstract #92

In July 2002, the software development lifecycle for ANSTO's instrument control system began. Requirements were gathered from the instrument scientists and their user communities through interviews, a web based survey and a workshop http://www.ansto.gov.au/ansto/bragg/workshops/comp_report.pdf with internationally recognised invited speakers. Current solutions in the X-ray and neutron scattering communities were surveyed by visiting a selection of institutes. It was found that there were pockets of collaboration, but not nearly as extensive as one might expect.

This paper analyses what components are required in a generic instrument control system, describes briefly what the ANSTO team is implementing and suggests there are benefits to increasing the level of collaboration between facilities.

*European Synchrotron Radiation Facility

Scope

The first step in any engineering project is to propose a scope of work. The scope of work referred to here was to provide an instrument control system for the neutron beam instruments at ANSTO. This scope ends where data reduction and data analysis begins. This appears to be an accepted boundary in the neutron and X-ray community.

Market survey

The next step was to survey the market to see what control system were already being used. This survey was easy to conduct because the beam-line control community is open and helpful. The survey included visits to six neutron and X-ray facilities in North America and six in Europe. A report on those visits was generated with recommendations. The recommendations were based on

1. Fit for use
2. Development and maintenance effort required per instrument

'Fit for use' was based on the experience of the developers and instrument scientists at the facilities. It was found that there was a wide variation in the development effort required, from 3 person months per beam-line to 5 person years. These numbers were estimations for developers experienced in the systems, so an estimate was made to allow for the ANSTO team's inexperience with the chosen system. There was also a wide variation in the learning curve of systems, from 2 weeks to 6 months per developer.

This was not an exhaustive survey of all available custom developments and commercial products. There may be value in compiling such a survey for use in the data acquisition community to help institute match products to their requirements. Industries including the process industry and sciences including astronomy have mature products for batch control. Whilst solutions may not come from these market sectors, their ideas may be useful.

Requirements gathering

Requirements must take into account all major stakeholders. The requirement gathering process was designed to gather information from the 3 major stakeholder groups: instrument users, instrument scientists (the customer) and management.

1. Web based user survey

The survey focussed on user experiences at neutron facilities[1].
The weaknesses users found with control systems included long data storage times, new functionality is difficult to add, poor default settings and poor integration of multiple computers.
The strengths included system stability, integration of user supplied ancillaries and user involvement in software development process.
Users said that system could be improved by

- well tested software
- infrequent changes to software
- fast processing
- modular design
- additional support personnel
- good documented

50% or more of respondents requested

- to be involved in the software development process
- remote (Internet) access to instruments and data

Respondents asked for interface standardisation between the Australian Synchrotron and ANSTO beam-line software without compromising either system.

Regarding the human interface, respondents requested

- GUI or GUI and command line
- good defaults
- real time visualisation, reduction and high level analysis
- standardised component names, commands and data files
- similar interface for all instruments
- simulation mode for training

2. Data Acquisition Workshop. Users and instrument scientists

This workshop was a gathering of representatives from the three major stakeholder groups. The workshop

[1] http://www.ansto.gov.au/ansto/bragg/2005/comp/anbug_survey.html

lasted two days, and was divided into seven sections for the seven instrument classes being built ie. powder diffraction, SANS, single crystal, three-axis, residual stress, polarisation analysis and reflectometer.
35 participants attended. Three overseas guest speakers were invited to provide their experience in running a facility and their perspective on solutions. Results of the market survey and web survey were also presented. A report[2] was generated which was open to participant review. The conclusions of the workshop supported the findings of the web and market surveys and added

- reactor variables available to instrument control
- NeXus as the data format
- convert NeXus to other formats
- GUI that guides new users
- user training

3. Management's requirements

ANSTO's management required a system that has

- low development cost
- low maintenance cost
- high reliability and availability

Our solution to keep development cost low was to find an instrument control product that fulfilled most or all of the essential user requirements. Our solution to keeping maintenance cost low is a generic system that can be customised to any instrument ie. a single code base that can be used to control a reflectometer, SANS, X-ray powder diffractometer etc. with minimal customisation. There were several candidates for system that fulfilled this requirement. Our solution to high reliability and availability is to keep the system as simple as possible.

The benefit of spending effort to gather requirements from the major stakeholders was stakeholder ownership and feedback from the beginning of the project, before one line of code was written. The requirements documentation also provides a clear roadmap for project management, including determination of feature priorities, the richness of each feature and project tracking [1].

The chosen ones

The best fit of the control systems survey to fulfil the requirements was the SINQ Instrument Control System (SICS) [2]. SICS has a client-server architecture and is object oriented.
A high priority user requirement was a graphical user interface (GUI). Hence the 3 major project tasks are

1. the development of an object-oriented rich-client GUI codenamed GumTree [3],
2. customisation of the SICS server for our instruments with the addition of a configuration database, and
3. Visualisation of data using ISAW [4].

It was envisaged that four products, SICS, TANGO [5], ISAW and GumTree would fulfil the majority of the user requirements.

[2] http://www.ansto.gov.au/ansto/bragg/workshops/comp_report.pdf

Minimising development effort

To fulfil management's requirements, it was necessary to review ways of minimising development cost (effort) and provide high reliability and availability (quality). There are three ways proposed to fulfil these requirements; design patterns, frameworks and collaboration.

Design patterns

To minimize the code required to control an instrument, code reuse should be maximized through the use design patterns [6] and object oriented programming. Gotz has described the building blocks for instrument control system for neutron or X-ray beam-lines [7]. Koennecke has applied object oriented design principles to build SICS.
We have found that whilst object oriented programming is naturally applied to instrument control features, applying design patterns to design an instrument control system is not a trivial problem [8] [9]. Some obvious patterns have been used in the GumTree development, including the Observer (for efficient communication), Façade (to allow for different control systems) and Proxy (to provide a simulation mode) patterns between client and server. The design of batch control and editing to manage the creation and control of a command queue was a major challenge.

Frameworks

During the scoping phase of the project, it became apparent that frameworks have become a widely accepted tool for accelerating development and fostering collaboration. Frameworks can reduce the effort required to develop software by up to 80%. The Eclipse [10] Rich Client Platform (RCP) was the framework chosen for the RRR project, and is based on a plug-in architecture. This architecture allows extensibility and configurability. It is suggested that Eclipse be reviewed by the NOBUGS community as a framework for collaboration.

Collaboration

Collaboration was found to be the most effective way to reduce development effort. The neutron and X-ray community are fortunate that source-code sharing appears to have a precedent in many institutes. The question that came to mind early in the selection process was "Why have data acquisition systems been designed for each neutron and X-ray facility, with many facilities supporting multiple systems?" The wheel has been reinvented so many times at significant cost, and without significant benefit.

Benefits of collaboration

Collaboration fulfils two requirements that ANSTO users requested, and that we suggest may be representative of the wider user community

1. User familiarity with interface across neutron and X-ray facilities – NeXus paradigm
2. Better quality software. Tested across instrument classes and across facilities

It also fulfils the intent of NOBUGS - New Opportunities for Better User Group Software. Resources can be released to explore new opportunities and exciting features for interacting with devices and data that are available now and are waiting to be implemented including

- Speech recognition and synthesis
- Intelligent instrument control (requested by NeSSI)
- Virtual reality (3D rendering of instruments and data)

Most facilities have limited resources for software development and would like to provide better service for users.
Increasing collaboration will require that you put effort into dealing with your collaborators. The benefit will be leveraging human resources, development, testing and support.

Collaboration and personnel

In general, the development teams for these products are small, and often one person is responsible for many beam-lines. The developers often have to be generalist, and will always be limited by their skill-set.
The developers are expected to do everything in the development lifecycle, including testing and support. This mixing of roles is not acceptable in most software engineering philosophies. Collaboration allows developers, tester and support staff to fulfil roles that best fit their skill-set and interests.

Collaboration and requirements

Requirements are similar from facility to facility. The authors believe that for requirements and implementation, there are more similarities than differences [7]. The overarching requirement is that users can do science easily and reliably.
Even where there are differences, the same software solution can be applied eg. remote access to instruments, network free solutions, large data volumes. The solutions need to be modular and extensible to allow features to be added or hidden easily.

Championing collaboration

Data analysis has NeSSI to champion collaboration. Data formats have NeXus. The data analysis community has NOBUGS.
There is an opportunity for data acquisition people to increase the scope of collaboration. This would require that the NOBUGS community becomes more active.
Does collaboration require that data acquisition software converges to a singular solution?
No. This would stifle innovation. However, there should be a focussing of effort, and this requires a reduction of the number of solutions supported, or an increase in the numbers of developers.
Does collaboration require that I retire my current data acquisition solution and use another?
No. The authors suggest that instrument control systems, like any software solution, have a finite lifetime. It may not be a requirement for you to address the question now because you have a stable system that addresses your user requirements. We propose here that when it comes time to upgrade your system, or to build a new system, that you investigate what already exists rather than developing something from scratch, and that you take into account GUM theory [7]

Existing collaborations

Collaboration exists and the numbers are growing.

NeXus is being adopted by many neutron facilities
Inter and intra facility collaboration for instrument control systems includes:
EPICS, TACO/TANGO, SPEC, Labview, MX, SICS, ISAW

These grouping should actually be made on the grounds of facilities with similar user requirements rather than technologies or solutions.

It is interesting to consider the ‘global’ user. She probably uses NeXus. She visits neutron and synchrotron facilities on three continents. She would like to use a diffractometer in Europe with a user interface that is the same as the one in the US. What is the list of her requirements?

Forum for collaboration – licensing, open source, shared source, mailing lists and e-notebooks

Would merely open-sourcing developments lead to greater collaboration? We believe that the community should ensure that their code is licensed in a way that aids collaboration eg. GPL (GNU public license) and is made available to the wider community through an easily accessible open-source repository like SourceForge. The existence and scope of the product could be announced to the NOBUGS community through Argonne’s neutrons mailing list. Overview documentation should be made available so that the community can browse information efficiently to find what is available.
The SNS have created e-notebooks for data reduction and data analysis working groups and we propose that a similar notebook be created for data acquisition.

Conclusions

Choosing an instrument control system is not a trivial effort. It is a major software engineering exercise that includes requirements gathering, market survey, collaboration, recruitment of personnel, system development and testing. Increased collaboration and the use of frameworks such as Eclipse were proposed as methods of reducing effort required developing and maintaining an instrument control system and will lead to higher quality software. Open-source and e-notebooks were proposed as methods to share source-code and documentation.

References

[1] PV. Hathaway et al. *An Integrated and Agile Approach to Delivering Extensible Instrument Control System Software.* NOBUGS 2004.

[2] M. Koennecke. *SICS Homepage*, http://lns00.psi.ch

[3] T. Lam et al. *GumTree Project Homepage*, https://sourceforge.net/projects/gumtree

[4] T. Worlton et al. *New Software for Neutron Scattering Data Visualization*. Neutron News Vol 15, Issue 3

[5] A. Gotz et al. *TANGO Homepage*, http://www.esrf.fr/computing/cs/tango/tango.html

[6] E. Gamma et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley 1995.

[7] A. Gotz et al. *Grand Unified Model for Control Systems – GUM*. NOBUGS 2004.

[8] T. Honkanen. *Design Patterns in Automation*. Postgraduate Seminar on Information Technology in Automation. Helsinki University of Technology. 2002.

[9] W. Nartz. *Design Patterns for Process Automation Systems*. University of Linz. 2000.

[10] Eclipse www.eclipse.org