Software for Neutron and X-Ray Scattering
NOBUGS 2008 Report
Final Version, July, 27, 2009

Why Should I read This?

Between November, 3 to 5, around 80 software professionals from neutron and synchrotron sources around the world met in Cronulla, Sydney for the NOBUGS 2008 conference. The purpose of the NOBUGS conferences is to foster collaboration on software and to encourage the creation of standards for exchanging programs and data. NOBUGS is an acronym for New Opportunities for Better User Group Software. This report will bring you up to date about the current trends in software for neutron and x-ray scattering. This report has been written by the NOBUGS International Organising Committee and thus represents not only a cross facility viewpoint but also the collected experiences of the NOBUGS participants.

Executive Summary

Software does matter. Adequate, fast and user friendly software for data acquisition, data reduction and data analysis increases the scientific output of a given facility and makes it attractive to users in an increasingly competitive scientific market. Sadly, scientific software for neutron and x-ray scattering is frequently in a sorry state. This is to a large extent due to the fact that no budget for software is set aside for e.g. the commissioning of a new instrument and thus gets left to the last. Clearly some software construction is required. But this has to be managed right. According to the experiences of the NOBUGS participants successful scientific software projects have to be managed along the following guidelines:

- Users need to be involved in the software construction process at all stages.
- An iterative software construction process with frequent releases and feedback loops.
- The project team ideally should consist of scientists and software engineers.
- The software has to be designed in an extendable and modular way.
- Software is not complete without adequate user and developer documentation.

A way to save resources is to collaborate with other like minded facilities. This can happen for data reduction and data analysis in technique specific clusters. For data analysis there is a trend to component based frameworks. This means that algorithms are split into sensibly small parts which are glued together with a scripting language, mostly python. The advantage is that it becomes easier to replace and modify components when this becomes necessary or to rearrange and re-use components in different algorithms. Collaboration in data acquisition systems is more difficult. This can be attributed to the fact that some 50% of the value of a data acquisition system is concentrated in the driver layer to access the hardware. And few sites use the same hardware. Nevertheless two patterns for how to do data acquisition are emerging. The first is that most data acquisition systems contain a scripting environment. Scripting languages allow to program instrument control on a day

to day and experiment to experiment basis easily. The preferred scripting language among NOBUGS participants is python. The synchrotron community converges towards a common architecture: for device access either TANGO or EPICS are used, with a preference for EPICS. On top of this there is some script server which executes python scripts on behalf of terminal of graphical user interface clients. This opens a new opportunity for collaboration through the exchange of python scripts. New facilities, or facilities undergoing an upgrade, are well advised to apply this pattern for their DAQ software. Experiences in the community show that commercial software and operating systems, especially Windows, are problematic for data acquisition. Commercial systems were found lacking in multi user capabilities, security, interoperability and transparency.

There is a strong trend in the community to give users access to data, training, computational resources and even instruments through WWW or grid technologies. While the Grid is primarily geared to provide access to large amounts of data and massively parallel computation, both the synchrotron and neutron community can benefit from the virtual organisation and authentication infrastructure developed for the grid to allow users access to facility resources in a secure way.

Why Bother About Software?

There are still people who have not realized it. But the days when scientific software consisted of several hundred lines of Fortran code are long over. Today's windows pampered users ask for sophisticated software preferably with a graphical user interface. Moreover the spectacular advances in computer power now allow sophisticated data treatment algorithms which simply were not practical just a couple of years ago. Consider only the field of instrument simulation which has proven very useful in the design of new or upgraded instruments. But uses computer power at an unprecedented scale. Unfortunately, writing software capable of matching current requirements exceeds the abilities of scientists. They have to turn to professionals who help them get the job done. Sadly, this does not happen frequently enough and thus facilities are stuck with outdated and user unfriendly software. Thus users and instruments scientists alike have to spend a lot of time fighting with software rather then producing scientific results. This impedes scientific productivity and the scientific impact of facilities is reduced. Which in turn will make it harder to justify funding for neutron and x-ray facilities in an ever more competitive scientific environment.

In the days gone by, facilities created data which was then analysed by scientists in the field who were specialists for the method. Today users instead ask for answers to scientific questions: how does the structure of this protein look like? What happens to my samples structure when I change the temperature? What is the magnetic structure of my sample? Only few wish to know all the gory intricacies of the method. Facilities are called upon to help such users and new users with appropriate software and advice by trained personnel. Otherwise they will loose users to facilities which do just that and perish in the worst case.

Collaborative Software Development

Before we consider software it is worth to consider first who our users are. For data acquisition and and data reduction software these are mainly the instrument scientists working at the facility and the visitors to neutron and x-ray facilities who perform experiments. For data analysis software it can be any scientist using the method. The latter two groups are very numerous and thus very difficult to communicate with. Therefore we frequently have to rely on the instrument scientists as their proxies to tell the software people what is needed. But this approach has its limitations: instrument scientists might become so entrenched in the ways of their facility that they fail to realize the benefits of different approaches.

Software for neutron and x-ray facilities is frequently underfunded. It is hard to allocate funds for infrastructure projects such as software and especially for the necessary maintenance. One way out of this is to be more efficient. We can be more efficient if we collaborate and reuse software developed elsewhere to a larger extent or share the burden of software construction with other facilities. Fostering such collaboration is the purpose of the NOBUGS series of conferences.

But collaboration in software construction needs to be done right. It requires proper management. And this is often lacking. From experiences from both failed and successful collaborations on scientific software some management guidelines have become clear.

Software construction benefits from clear requirements. Now, scientists are notoriously bad in specifying requirements. They usually know what they do not like but cannot always tell you what they want. Not only dues to this, requirements frequently change during the course of the project. Often, there is a communication and cultural barrier between scientists and software engineers. These difficulties can be overcome by a couple of measures:

- Involve users as much as possible with the construction of the software. This is done best by using an iterative software construction process. This involves that the users gets to see and use prereleases of the software frequently. Feedback from the users is used to steer the further course of the project. Another aspect of an iterative software process is that management gets early feedback about the progress of the software too and can enact appropriate measures if things do not advance as they should.
- Design the software in such a way that it is modular and allows for easy extension and change. A framework which ensures architectural consistency is a worthwhile trait.
- Mix scientists and software engineers in the project team.

Another aspect is documentation. No software is complete without an adequate set of documentation. Both user and developer documentation are required. While the need for user documentation is immediately clear, the case for developer documentation may be less so. Developer documentation means that the design and the internal structure of the system is described in a legible way. In text, through UML diagrams, Use cases or whatever else is appropriate. This documentation becomes necessary when there are changes in the project team, when the software has to be adapted for use at another facility by another person or when the software has to be maintained in a later stage by a person who is not among the original authors.

Of course software collaborations require adequate resources. A software collaboration which is run by people in their spare time between other jobs is doomed to fail. Software projects also require managers who understand the business at hand. Real collaboration on software goes beyond: I give you a copy of my source code. Real collaboration does include consultations of partners already in the design phase, user training, documentation and maintenance of software.

Software maintenance is often overlooked. In any software project the work is only half (or less) done after the initial construction of the software. Afterwards maintenance is necessary: bugs need to be fixed, new requirements need to be accommodated, people may need help with the use or extension of the software. This requires dedicated personnel in hard to get by permanent positions. The NOBUGS community has identified a need for the maintenance of some common shared software packages. Examples include the McStas simulation package for neutrons, the EPICS and Tango device access software, the NeXus data format and some more. Unfortunately no suitable management structure exists which supports this necessary work across facilities worldwide.

It is also interesting to look at the programming languages used for current projects: most projects use Java. C++ is used when performance is an issue. When it comes to scripting languages, there is a strong preference for python among NOBUGS participants.

Data Acquisition

Data acquisition is probably the area where the little collaboration is happening. This is a consequence of the fact that more then 50% of the value of a given DAQ system lives in the software interface to the hardware. And of course everyone has different hardware. But some patterns still reoccur:

- Most data acquisition systems have some scripting support built in. This reflects the fact that the requirements for instrument control may vary from day to day and from experiment to experiment. Scientists can program the necessary adaptions themselves if they are provided with a documented and reliable scripting environment by the software professionals. The preferred scripting language is python.
- At this NOBUGS conference it became clear that the synchrotron sources converge to a very similar architecture. Most use either TANGO or EPICS for device access, with a preference for EPICS. On top of this there is a script server which executes scripts on behalf of the user. Terminal and graphical user interfaces written in Java or QT interact with the system though the execution of scripts on the server. The preferred scripting language is python. This opens an opportunity for collaboration through the exchange of python scripts. Neutron facilities might consider to adopt this well proven pattern for new installations or major upgrades.

For neutron scattering facilities there is a trend towards event mode data acquisition. With modern computing power this becomes possible. In traditional hardware based histogramming techniques, the binning has to decided upon in advance and the gating, i.e. the interruption of data collection if something is wrong, has to be done in hardware. In event mode both these tasks are handled in data reduction which is more flexible. With event mode data, even some new science is possible.

One interesting report highlighted serious problems with the use of closed source applications and operating systems in data acquisition. The reason is that DAQ software has to run very stable over weeks and possibly months. This involves that all the hard bugs, those which occur may be once a day, have to be fixed too. This is often not possible with closed source systems because the source code cannot be inspected nor changed. Windows was also found to be lacking in security features in a multi user environment. Windows in special also tends to attract unwanted applications, like games. The facility is now switching to Linux which offers source code access and more capable multi-user support. There is a strong preference in the community to use open source solutions for DAQ systems where possible.

At NOBUGS 2004 it was agreed that a common cross facility interface to data acquisition would be valuable. With the treepath concept such an interface has been proposed. The treepath concept maps the aspects of instrument control: parameter editing, command execution, batch processing and data for online graphics into a tree structure and operations on the tree structure through the network. This concept has the potential to enable collaboration on adaptable and generic graphical and scripting interfaces to both neutron and x-ray instrumentation.

There is a user demand to integrate online data analysis with data acquisition in order to tune the experiment to the scientific problem at hand. However, there is not yet a clear pattern how this can be done in a general way. Some more research and resources should be devoted to this issue.

Data Analysis

In data analysis and reduction collaboration can happen in technique specific clusters. State of the art in data analysis are component based frameworks. Algorithms are separated into the smallest sensible steps and bound together through a scripting language, mostly python. The advantage of this approach is that it becomes easy to modify or replace components or to rearrange components to form new algorithms. Examples of such frameworks include DANSE and mantid. However, there is no agreement yet about the optimum granularity of components and how best to organize the flow of data between components. And the only way to find out is to look at the experiences gained from the frameworks currently under development. There is also the question if data analysis needs graphical user interfaces or a scripting interface. Scripting comes in handy if a large number of data files need to be processed in a loop, possibly depending on some condition. The answer is that we probably need both, graphical and scripting interfaces, but with a common engine underneath.

Another area where NOBUGS could help is on the interfaces between the stages. For example the interface between data acquisition and data reduction is commonly a "raw data file" and that between data reduction and data analysis a "reduced data file" (e.g. S(Q,W), I(Q)). Specifying a common standard for how information at these stages is stored or transmitted would allow the sharing of programs that move data between, or modify data at, these points and could ultimately lead to collaborations on such programs.

Grid Technology, Portals and Remote Access

The high energy physics community has developed the Grid technology for storing large datasets in a distributed file system and for the massively parallel processing of such datasets. In order to do that security mechanisms had to be developed. At the heart of Grid technology thus is the concept of virtual organisations (VO) and a distributed authorisation infrastructure which allows secure and controlled access to resources and services. The neutron and synchrotron community can benefit from the Grid development in two ways.

The first is of course to use the Grid for computationally intensive tasks such as instrument simulation or the processing of large data sets. But Grid technology is complex and users need help in using the Grid. Thus some reports were given on portals which permit users a simplified access to the Grid.

The other use of Grid technology is to leverage the Grid authentication mechanisms in order to solve the security problems associated with giving external users access to facility resources such as data or instruments. An example is the IE and VCR (virtual control room) system developed at elletra in order to allow remote users access to instrumentation.

Remote access to instruments is also a concern in the community especially at synchrotron facilities and especially for the protein crystallography beam lines. It has become clear that solutions based on remote desktop protocols such as vnc or nx are deficient due to bandwidth and, more serious, latency problems. Besides IE/VCR, described above another approach to remote access to instrumentation was shown with the the CIMA infrastructure. CIMA is based on web services, the Grid authorisation system and TANGO.

It becomes common for facilities to give users access to computing resources, training materials and data through WWW interfaces. A state of the art system which allows users search and access data in a secure way is ICAT, among others. This system also allows to connect proposals, data, analysed data and publications in a central database. Other examples of similar systems exist.

However, this field is still very much in development and some more experience is required. There are also a lot of policy questions which have to be answered: who will participate in virtual organisations? Can we agree upon a common authentication mechanism? What shall be the policy for data access? If the exchange of data becomes more widespread, the need for a common data archive format like NeXus becomes more pressing.

Protein Crystallography

Since several years the synchrotron protein crystallography (PX) community spearheads a trend to ever more increased automatisation of experiments. This community also is the most open to the concept of remote access to instrumentation, discussed above. This still continues as proven by reports about even more powerful sample changers being taken into operation. A new trend is to model instrumentation in 3D and the usage of such models for collision detection and prevention. This is an interesting approach which has to be watched and may find uses outside of the PX community. The problem is however the construction of accurate 3D models of instruments. This usually requires a lot of work and can easily be broken by users who throw almost anything into the beam line.

References

- Nobugs-Wiki: http://www.nobugsconference.org
- NOBUGS-2008 on the WWW:
  http://www.nbi.ansto.gov.au/cgi-bin/nobugs2008/overview.ws3
- TANGO: http://www.tango-controls.org
- EPICS: http://www.aps.anl.gov/epics
- McStas: http://mcstas.risoe.dk
- NeXus: http://www.nexusformat.org
- DANSE: http://wiki.cacr.caltech.edu/danse/index.php/Main_Page
- MANTID: http://www.mantidproject.org
- IE-VCR: http://www.gridcc.org
- CIMA: http://www.instrumentmiddleware.org
- ICAT: http://www.metaxa.com/icat.asp

The NOBUGS International Organising Committee