Treepath Based Instrument Control

Mark Könnecke, Markus Zolliker
Paul Scherrer Institute
5232–Villigen–PSI
Switzerland
Nick Hauser, Ferdi Franceshcini, Tony Lam
ANSTO
Menai, NSW 2234
Australia

October 29, 2008

Abstract

In the process of the development of a graphical user interface to an existing command line based control system, SICS, the authors developed a system for communication between an instrument control system and a GUI. This system has the potential to become a platform on which collaborations on GUI's for instrument control can be founded. Moreover our solution can easily support command line interfaces too. Our solution maps the four aspects of instrument control: parameters editing, command execution, data for online graphics and batch processing into a tree structure and operations on the tree structure. A protocol for interacting with this tree structure over the network is also described. Client software to this system only needs to know how to handle the tree and the protocol without further knowledge of the underlying control system. Means of introspection are provided which allows client software to discover the available tree structure and adapt itself to the instrument being controlled. On top of this two different Eclipse-RCP based graphical user interfaces have been developed.

1 Introduction

Both sites (PSI, ANSTO) involved in the collaboration leading to this paper use SICS (SINQ Instrument Control System) as their control system for controlling neutron scattering instrumentation. SICS is a client server system with a server written in ANSI-C doing all the hard work and clients, mostly written in Java, contributing the user interface. Client and server communicate with each other through TCP/IP sockets and a simple ASCII command line protocol. Our new solution was developed from three lines of thought.

When instrument scientists at PSI were asked about their requirements for a new GUI, they were initially not very enthusiastic; they prefer the time honoured command line and batch files. But a parameter editor which would save them from memorising the plethora

of parameter names seemed like a good idea. A cursory look revealed that even a simple instrument like a neutron powder diffractometer can easily have in excess of a hundred parameters. So there was the problem of displaying all of this information in a sensible way. Fortunately an instrument breaks down into components and into devices; thus a hierarchical tree structure seemed like a good idea.

Another source of inspiration was Darren Kellys research on using XPath technology to connect a user interface to an instrument model.

The next line of thought came when we tried to develop a GUI for SICS. We quickly realized that the ASCII protocol was designed for human users. But it was not friendly for a computer to parse. Especially it was not always clear which message from SICS belonged to which action. Clearly some work was required. Moreover, there is the problem of propagating parameter changes to the client. Having the client poll would create a lot of network traffic and load on the server. But the SICS server knows best when new data is available, because may be a motor is moving or a scan point has finished and should rather push changes to the clients.

After we implemented our solution we realized that it may have wider applicability. Especially in the light of the insight from NOBUGS 2004 that a common interface to control systems would be of great benefit to users. Our solution is a working proposal for such in interface.

2 The Treepath Concept

The treepath concept consists of the following components:

- A mapping of instrument concepts into a tree structure
- A communication protocol to interact with a tree
- A network protocol to allow asynchronous I/O

There is a clear divison of labour in the treepath concept: The user interface, be it graphical or not, only interacts with the tree. The underlying control system (in our case SICS) translates tree operations into changes to hardware and software parameters, executes commands, writes data files etc. It is advantageous if the underlying control system also makes sure that the parameters in the tree are updated at sensible time intervalls. However, the treepath concept would also extend to a client which regularly polls the tree for parameter updates.

2.1 The Tree Structure

In instrument control several aspects have to be addressed:

- A lot of parameters need to be edited
- Certain commands, like count or scan commands, can be executed against the instrument.
- An instrument provides data for online graphics which can be used by the scientist to judge the progress and the quality of the experiment

- Many instruments need to be programmed to perform a series of measurements unattended

Let us start by addressing parameters first. Any parameter is stored as a node in a hierarchical structure. The exact hierarchy used is actually unimportant and up to the instrument scientist to define. However, a hierarchy organized along major instrument components and devices appears to be a sensible idea. Tree nodes have types: the usual suspects: integers, doubles, strings and integer and double arrays are supported. Tree nodes can also have properties. Properties are key value pairs associated with a given tree node. Such properties are not meant to be shown in the user interface but are a means to communicate additional information to the user interface. Examples of properties used are:

values A comma separated list of values possible for this parameter. Example: monitor, timer for a count mode

priv The privilege required to modify this parameter.

viewer A hint to the UI that this node may benefit from a special viewer.

type Indication for special node types.

In future examples of tree structure listings required properties will be given as (key=value) in parantheses.

Commands are also stored as nodes structures in such a tree. A command node is a special node which has the parameters of the command as children. As an example, the tree structure for a count command would look like this:

- count (type=command)
  - countmode
  - preset

Command invocation is then accomplished by first setting all the parameter nodes and then the command node itself. Which will be translated by the underlying control system as an invocation of the command with the parameters given.

Data for online graphics is also stored in a tree structure. As an example, consider a powder diagram:

- powder-diagram(type=graphdata)
  - rank
  - dim
  - two_theta (type=axis) (dim=0)
  - counts (type=data)

Data for online graphics will be stored under a node with the type property of the node set to graphdata. This node is supposed to have a rank child node which must be initialized as an integer holding the dimensionality of the data. Another child node, dim, is an integer array holding the size of the data along each dimension.Plot axis are described by

nodes which have the name of the axis and hold the data for the axis. They are identified to the UI through the property type being *axis* and the property dim indicating to which dimension the axis belongs. The actual plot data is stored under a node with a freely given name. But the node must have the type property initialized to *data*. Thus the node structure supports any dimensional data to be stored. Optionally, a child node title can be used to define a sensible plot title. Otherwise the graph node name is used.

Support for batch processing can be achieved by recording a sequence of tree operations and replaying this sequence either in the UI, or after a suitable translation, as a batch file at the instrument server.

2.2 Communication with the Tree

The user interface needs to communicate with the instrument tree. The nature of the transport between the user interface and the actual instrument is not important. The solution presented implies a client program communicating with some instrument server. This communication can happen through TCP/IP, through pipes or by forwarding commands to an internal interpreter. Tree operations through any kind of transport are done through an ASCII command protocol.

Tree nodes are identified through unix-like path strings. For example /instrument/monochromator/w identifies the wavelength in the monochromator subtree of instrument.

A user interface only needs to deal with a small set of commands to interact with the instrument tree:

getgumtreexml path Returns a rendition of the tree under path in XML format. The user interface can initialize its internal tree from this description. The XML format used is IBM's SDO (Service Data Object) format. SDO was developed by IBM and Bea to transfer data between differerent enterprise systems. Each SDO object provides a data graph which in turn can hold data and further objects. Tools exist to automatically convert SDO XML into objects in various programming languages.

hset path value value value... Set a nodes values(s).

hget path Returns a nodes value in the form, path = values value value... If the transfer property of the node is set to zipped, data is returned in a compressed binary format. This allows for the efficient transfer of array data.

hnotify path enables automatical notification of changes to node values of all nodes underneath path.

hlist path Lists the children of the node described by path.

hlistprop path lists the properties of the node identified by path.

hgetprop path key returns the value of the property key on the node identified by path.

If an operation on a tree node yields an error the underlying system is supposed to return a string describing the error with the prefix ERROR. Operations on tree nodes may also result in additional logging text data to be sent. For instance during scans.

In order to be able to unanimously assign incoming messages to the tree nodes causing them the simple command protocol is wrappped with an ACT (asynchronous completion token) mechanism. Thus the final form of a complete hset command looks like:

```
contextdo 122 hset /instrument/monochromator/wavelength 3.5
```

where 122 is an arbitray identification number generated by the client. The server then sends responses in the form:

```
122::>response<::
```

This is any message is prefixed with the identification number followed by `::>` and terminated with `<::`. Thus all communication happens asynchronously. This is advantageous because responses may be delayed as certain operations (driving motors for example) take some time to complete. As an alternative messages may be wrapped into json format. Json is a simplified notation to encode structured data into ASCII.

An important point is that the treepath protocol is more a message protocol rather then a request-response protocol. For example, an hset on an motor may yield an reponse or not, but may also send a message later on that it ran into a limit switch. An hset on a reciprocal space motor may cause a whole flurry of update messages from the actual motors implementing that movement in reciprocal space.

3 Command Line Interfaces to the Tree

The tree access protocol given above is already a command line interface to the treepath concept. This interafce is a clumsy one and requires a lot of typing by the user. But it is in fact very easy to write a generic wrapper which translates the treepath protocol to the widget command, function call syntax preferred by one of the authors of this paper. This then looks like:

object parameter to retrieve a parameter

object parameter value to set a parameter

object function par par ... to invoke an object method

function par par ... to invoke a general command

It is also very easy to convert the tree structure into an object model in the OO programming language of your choice. This can either happen through introspection on the tree or by directly converting the SDO-XML description into objects. The latter method has already been demonstrated to work for python.

Code genereration for procedural or functional languages is als an option.

The treepath concept can be easily adapted to fit into the ReST concept for WWW-services. ReST already uses hierarchically structured URI's to address objects. Objects can also be treepath objects which understand certain messages, namely set and get.

4 Implementation into SICS

The SINQ Instrument Control Software SICS has been extended to support the configuration of an instrument tree and the tree communication protocol through ACT as described above. This extension is called Hipadaba for hierarchical parameter database. Parts of these extensions, namely the tree node code, can be reused in the implementation of other ANSI-C based control systems.

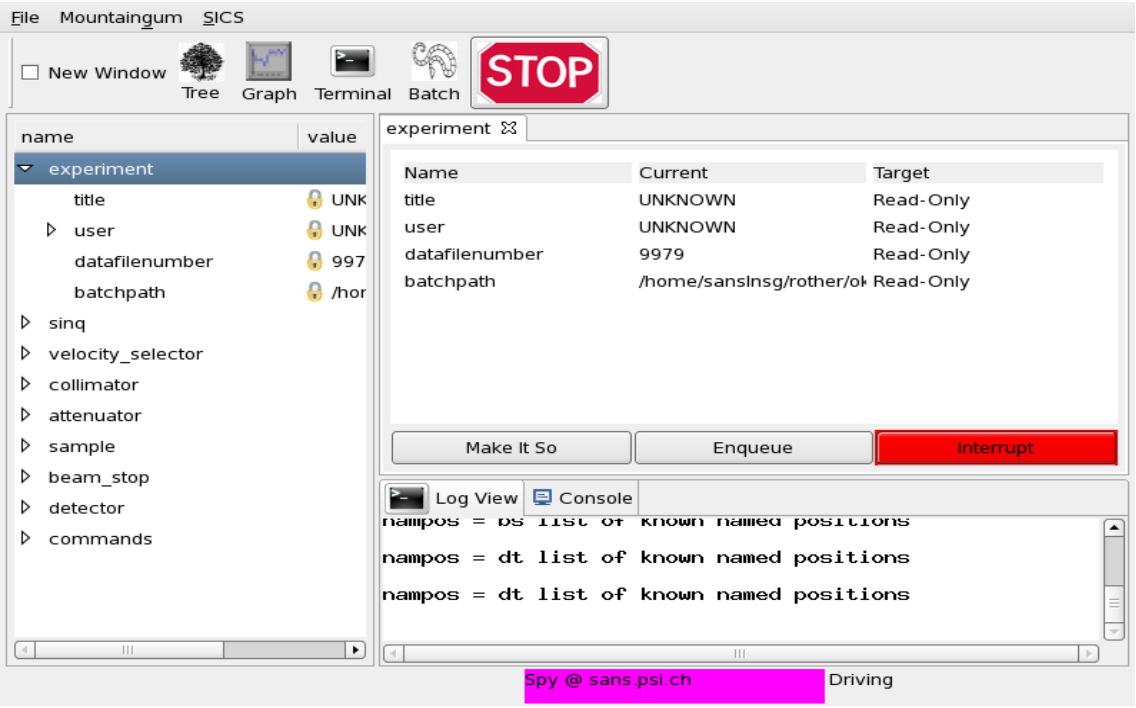

Figure 1: Gumtree SE Parameter and Command Tree Display

5 Graphical User Interfaces

On top of the Hipadaba enabled SICS two graphical user interfaces have been developed. Both are written in Java and are based on the Eclipse Rich Client Platform (Eclipse-RCP). Eclipse-RCP provides an application framework and for extending applications by contributing plugins in a standardized way.

5.1 Gumtree Swiss Edition

Gumtree swiss edition(GTSE) is PSI's version of a graphical user interface to the treepath concept. GTSE shares many traits with Gumtree ANSTO, described below. In contrast to Gumtree ANSTO, GTSE is an instrument control application only because PSI did not wish to integrate data analysis with instrument control to tightly for stability reasons. Moreover PSI had the constraint that the communication with SICS should follow closely the old command line protocol in order to reduce the confusion of veteran SICS users. GTSE assumes that the tree has the following structure:

/instrument is to contain the instrument parameter hierarchy including the commands to be executed on the instrument.

/graphics shall contain the data for online graphics and data analysis.

/batch contains helper nodes for the batch processor.

/quickview contains the configuration for the configuration of the quickview display.

Figure 1 shows the parameter tree display of Gumtree SE. To the left is the tree, the top right shows the default editor for parameters. Editors are opened by double clicking

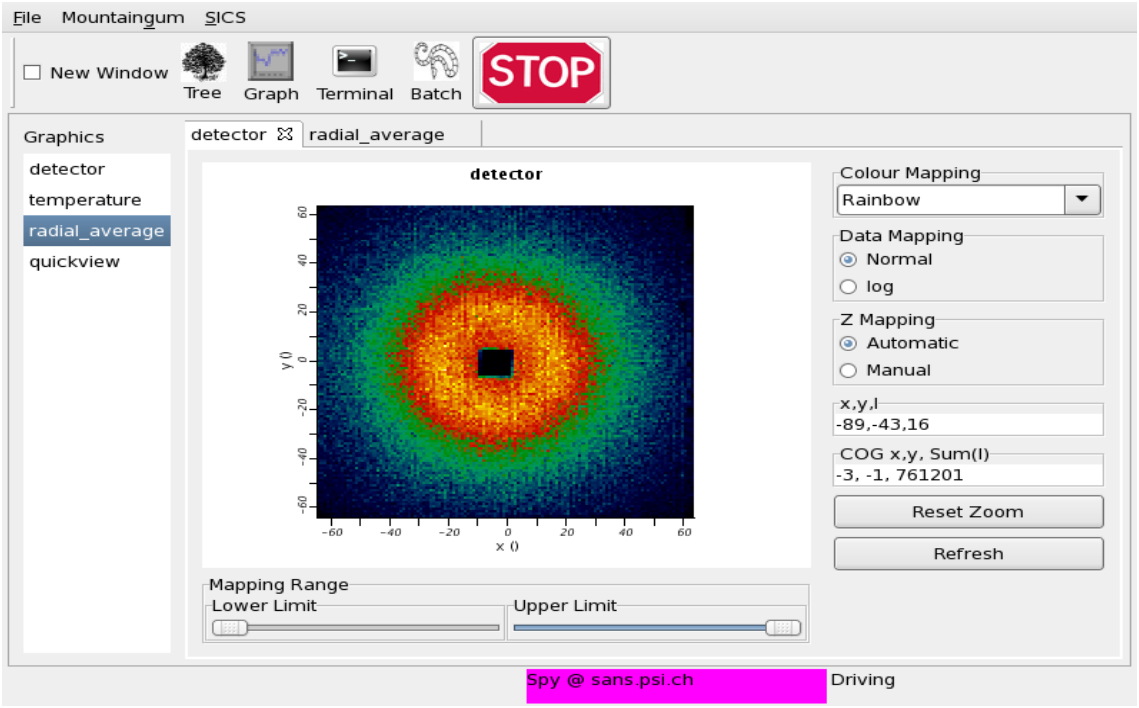

Figure 2: Gumtree SE Graphics Display

on a node in the tree display. It is possible to have node specific editors which are selected through the viewer property on the appropriate tree node. From an editor, an action can either be sent directly to the instrument (Button: Make It So) or enqueued in the batch sequence (Button: Enqueue). The bottom right panel displays SICS responses to the actions invoked.

Figure 2 shows the online graphics display of Gumtree SE. To the left is a list of available graphics. Double clicking on an item shows it in the area to the right. Displayed is the default viewer for 2D detector data.

Gumtree SE is packaged into four separate packages:

Parameter Tree The abstract classes belonging to the parameter tree and helper classes.

Graphical User Interface Everything for the GUI.

TreeAdapter The connector between the Parameter Tree and the underlying control system. In our example SICS. Exchanging this tree adapter is another means to reuse parts of the code base with other control systems.

Application a thin wrapper which binds the other packages together and configures them.

GTSE is already in use at 9 SINQ instruments. GTSE is still in active development. It will be extended to support the other SINQ instruments, too. The batch editor is likely to undergo a major overhaul. It is also envisaged to provide a new tree adapter to read NeXus files and build a NeXus file viewer/data analysis tool from the existing code base as a separate application. Another separate application to be derived from this codebase is a parameter editor for the instrument electronics.

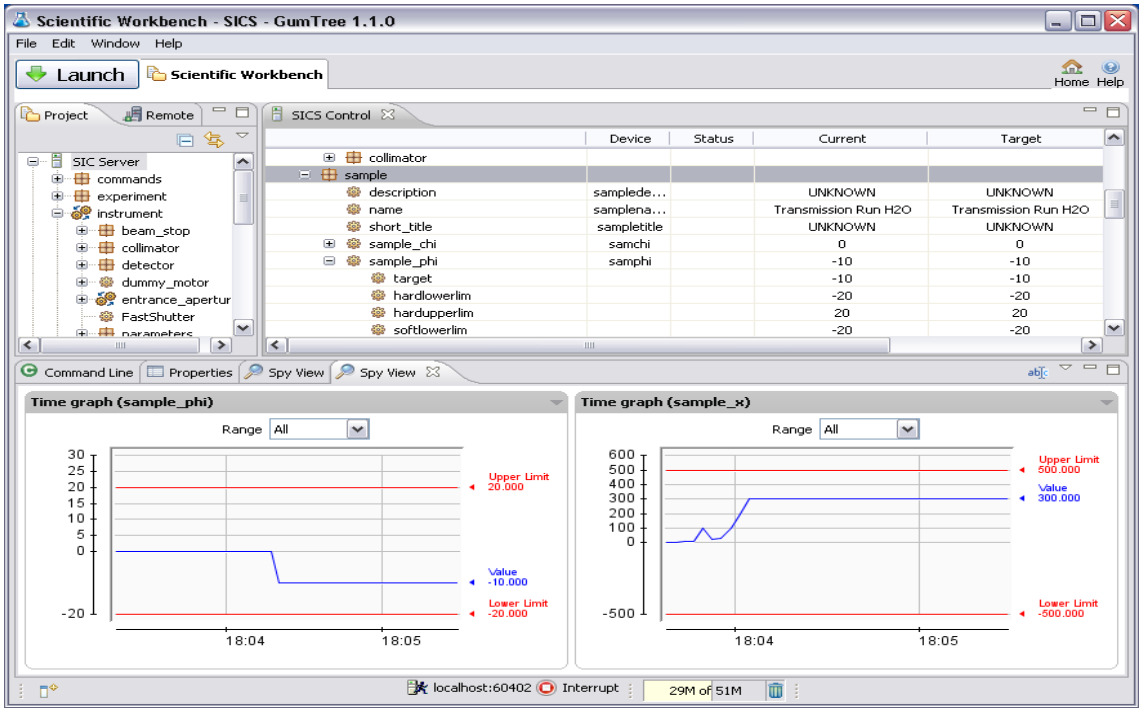

Figure 3: Gumtree ANSTO Tree Table Editor

Due to a grave lack of man power it is not possible for PSI to develop custom user interfaces for each instrument. The only possible way to a GUI for SINQ instruments is to configure different trees for different instruments (2 days/instrument) and use GTSE as a generic tool to interact with the instrument tree.

5.2 Gumtree ANSTO

Gumtree ANSTO is a sophisticated and generic workbench for scientific work. It is meant to cover all steps from data acquisition, data reduction and data analysis. Gumtree ANSTO loads the instrument tree into itself as a model. Once this model has been loaded a variety of different representations can be generated. The default representation is a table tree editor. Figure 3 shows the GumTree ANSTO table tree editor. Filtered representations of the main tree can be created which show user definable subsets of the complete tree.

Status and control dashboards are another representation of the instrument model which can be used to access often used instrument parameters quickly. Dashboard displays can be configured by editing XML files. An example of a GumTree ANSTO dashboard display is to be seen in figure 4.

GumTree ANSTO also includes a sophisticated data flow framework for data processing, called Cicada. Cicada uses as its data model netCDF GDM. GDM is a workspace like data model. GDM data is fed into processor components which optionally take parameters and generate output GDM which in turn can be fed to further processors. A UI component, Kakadu, is provided to graphically control Cicada. All this can be integrated with data acquisition. An example of such a configuration can be seen in figure 5. Data acquisition becomes just another processor in this system which yields data which is then

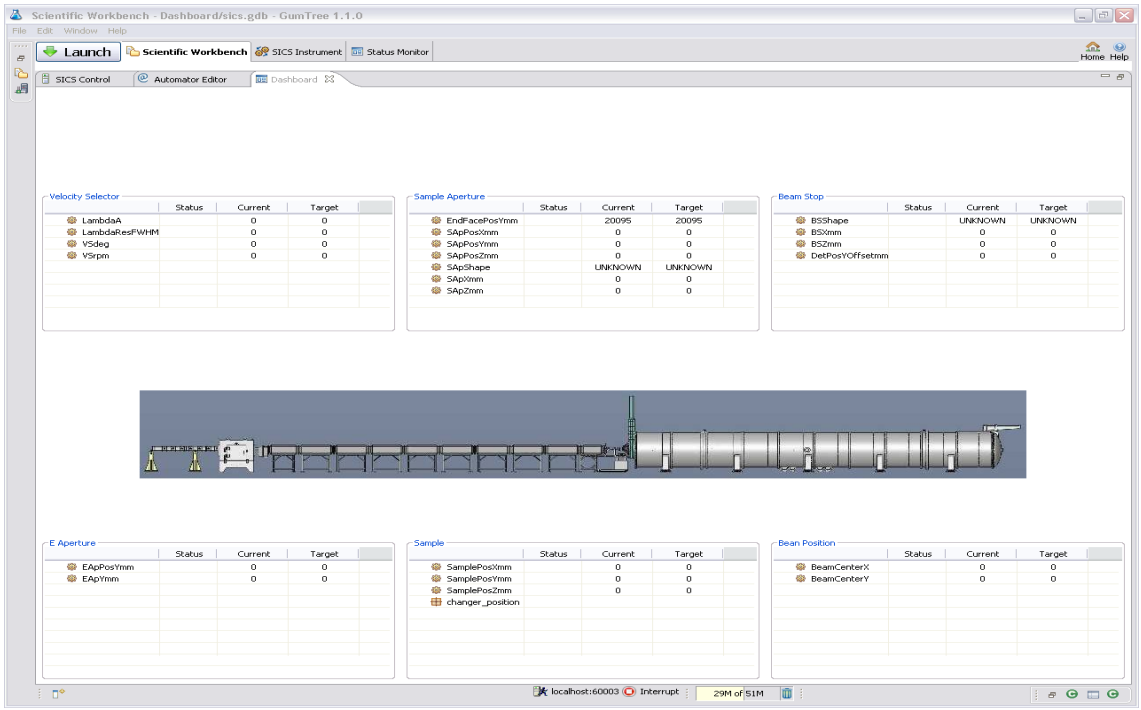

Figure 4: Gumtree ANSTO Dashboard

automatically fed into further processing pipelines.

Another GumTree ANSTO feature which uses the treepath concept are workflow components. A workflow is a sequence of operations a user has to follow to perform a successfull experiment. Much like wizard user interfaces seen in other systems. GumTree ANSTO's workflows are represented as XML files which can be loaded, saved and modified as needed.

Some instruments use customised control screens. Such screens use treepath to map UI gestures to actual changes in the instrument.

Last not least GumTree ANSTO incorporates powerful scripting support through the Java Scripting API which provides access to javascript, Ruby, Scala, Groovy and cPython. Scripts use treepath syntax to access instrument data.

6 Conclusion

The treepath concept maps the aspects of instrument control, parameter editing, command execution, online data display and batch processing into a tree structure and into operations onto this structure. An example implementation of this concept has been implemented into the instrument control software SICS and two graphical user interfaces have been designed to work on top of it. The treepath concept has a couple of interesting properties which make it a candidate for a generic approach to instrument control.

- The treepath system is the only concept known to the authors which allows to access a complete instrument with all capabilities in a uniform way. Systems like EPICS or TANGO only provide an abstraction for hardware access.

- The instrument becomes discoverable. It is possible to approach an instrument

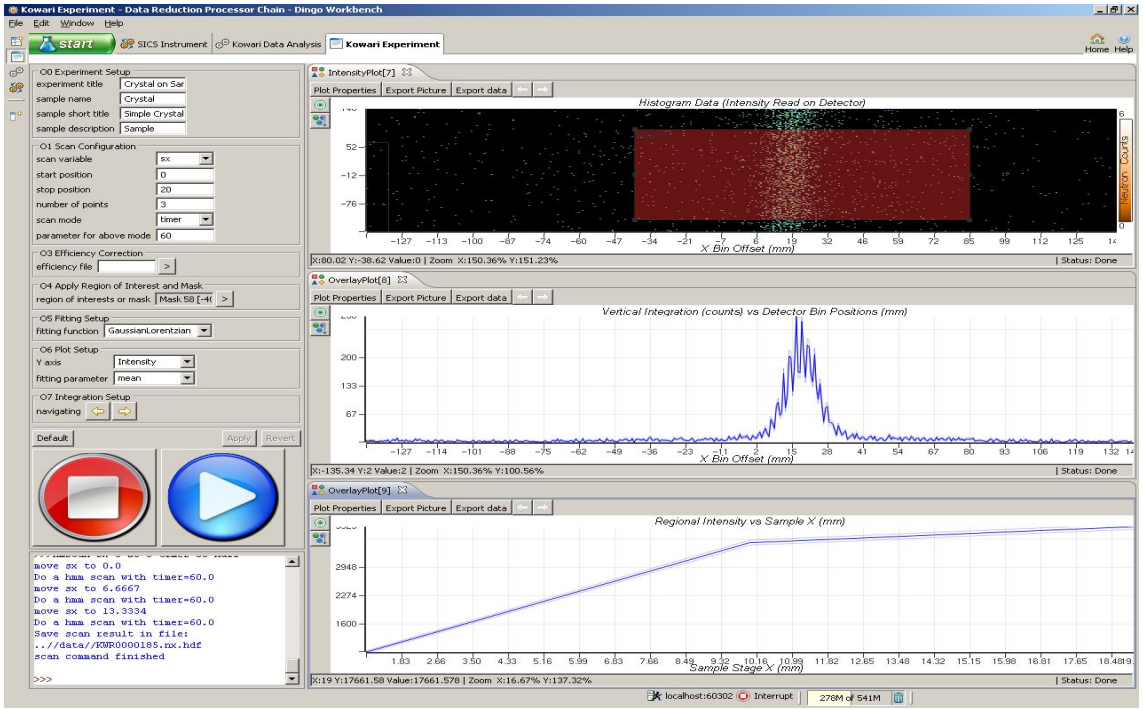

Figure 5: Gumtree ANSTO: DAQ intergated with DA

defined through a tree structure without prior knowledge and find out which parameters can be set, which commands can be run, what data can be viewed etc. Thus the user and the user interface have a chance to adapt themselves to the instrument given.

- The exact layout of the instrument tree can vary in a wide range, as defined by the facility and the instrument scientist. It is even conceivable to have different trees for casual users, instrument scientists and technical stuff, each exposing more detail of the instrument. Our implementation reflects the NeXus hierarchy.

- The treepath approach scales well to huge numbers of parameters.

- Though the concept suggests a user interface which utilizes a tree structure, it is not limited to it. Any user interface is possible, it just has to map its operations into tree operations.

- The introspection possible through the tree structure has the additional advantage that a generic interface can be written in order to cover a large variety of instruments. Given the constraints on manpower and funding at large facilities this is a great help.

- The treepath concept does not only support graphical user interfaces but command line or scripting interfaces as well.

- If we invest the work to define the tree structures and the network protocol to acces it more rigourously, we can define a standard upon which collaboration on graphical user interfaces for instrument control becomes possible.

- The concept can be extended to data analysis too: a large number of parameters is frequently involved. A data analysis procedure can also bee seen as a transformation from a source tree to a result tree.