NEW DEVELOPMENTS FOR THE JCNS NEUTRON SCATTERING INSTRUMENTS

M. Drochner, A. Erven, L. Fuss, H. Kleines, M. Wagener, FZ Jülich / ZEL, Germany
J. Heinen, D. Korolkov, S. Mattauch, M. Monkenbusch, FZ Jülich / IFF, Germany

Abstract

Since the Juelich 'Dido' research reactor was shut down 5 years ago, the larger part of its neutron scattering instruments was moved to the new FRM2 neutron source at Munich, and some newly constructed.

While we used a set of standard components as far as feasible, we had to accept some compromises due to the time pressure to get things going. In particular, we have a number of different graphical and command-line user interfaces now which is difficult to handle for users who work on multiple instruments.

After the initial rush, we are now working towards some consolidation, both on a common GUI philosophy and a high-level scripting language.

The design of the new user interface will be presented.

INTRODUCTION

Almost fife years ago, the Jülich research reactor "DIDO" was shut down and the bigger part of its neutron scattering instruments was moved to the new "FRM2" neutron source at Munich within about two years. Also, some new instruments were constructed (as the "SPHERES" backscattering spectrometer) or are just beginning construction (as the "MARIA" spectrometer).

At that time, we decided on a modular toolbox to use for all control systems under our responsibility, called "Jülich-Munich Standard" [1]. This covers the computing architecture (PC compatibles running Linux), a middleware framework ("TACO" from ESRF), process control frontend (eg. PLCs, CompactPCI, ProfiBUS) and process control clients (Qt, Python).

But as the term implies — this is just a box of tools, and there are various ways how to use them. In particular the UI philosophies, and the look and feel of graphical control programs differ massively now. Reasons contributing to this might be:

- Due to the number of instruments and the not always realistic schedule, all energy was needed to get the instruments running at all. UI consistency is considered second priority in such a situation.
- While only few people got the technical skills to judge about control system internals, many more have a funded opinion about user interface design. This makes the decision process time consuming.
- Some instrument responsibles wanted to keep the familiar menu structure of old programs which ran the instrument at Jülich.

It was not until recently that discussions about a common look and feel of the user interfaces started to take place in a bigger circle of intrument responsibles and software experts. Now we are reaching consensus, with the script interface as a first step.

EXISTING SOFTWARE

Figure 1 shows the software modules covered by the "Jülich-Munich standard" mentioned above.

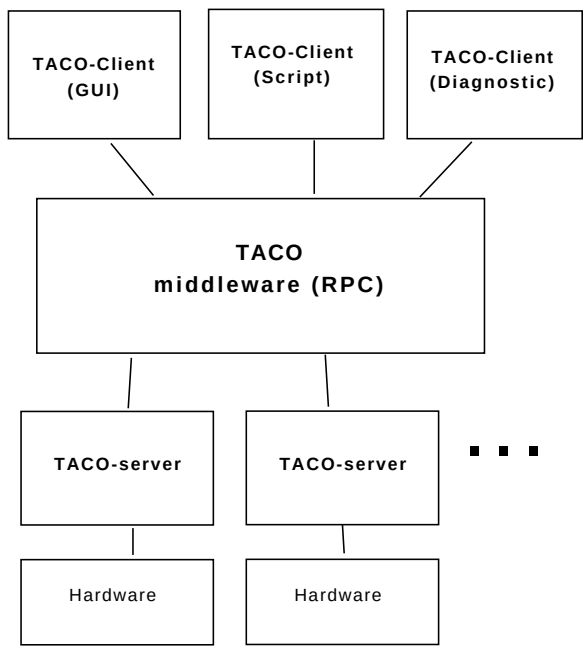

Figure 1: General structure of the neutron scattering instrument control software — Servers are reused among instruments but clients are mostly instrument specific.

The "TACO" framework is identical for all instruments, and there is a set of "device servers" which is used all over the place, due to the standardized frontend hardware and some abstraction work done by the PLCs. The situation is quite different at the client side, in the programs which comprise the user interface. Even while restricting ourselfes to the Qt and Python tools we've standardized to, we now got:

- a manual control program for each of about five instruments, implemented in Qt, used for commisioning, repair and various adjustment tasks. These programs visualize the instrument or parts of it; they are individually tailored (except where instruments are very similar which is the case for two small-angle scattering machines). Figure 2 shows an example of such a visualization.

- For the two small-angle scattering machines mentioned, a triple of "configuration", "definition" and "control" programs. The first one is used by instrument responsibles to define allowed and calibrated states of the machines, e.g. detector positions and setpoints of temperature controllers. The second allows users to define series of measurements (so-called "scans"). It works offline (i.e. not connected to the instrument hardware): Parameters can be chosen from those specified by the "configuration" programs; the result is put into XML database files which can be used later when the machine is available. The last one controls the real measurement. It interprets the XML definition files (with help of a Python script which will be mentioned later again), displays basic machine state, progress within a "scan" and estimated time to finish.

- Most if not all instruments use a graphical program to provide a live display of the spectrum currently measured by the detector(s). These programs are also tailored to the properties of the individual hardware.

- Python is used as scripting language at different levels of abstraction: Some scripts use the low-level TACO binding which comes with the software distribution, some use intermediate Python libraries which hide the details and provide eg. user friendly axis names and transformations.

- Python is also used as programming language where the code remains fixed and gets invoked either as a shell command or within a bigger framework as in the small-angle scattering case described above.

In addition there are some user interface programs which are neither Qt nor Python — either originating from software older than our standardisation efforts, or work of individual instrument scientists with their private ideas and preferences. It is not always feasible to rewrite those programs just for the sake of uniformity, but we should try to get them under the umbrella of a common interface philosophy.

ONGOING WORK AND GOALS

During discussions, it became appearent that it is much easier to reach a consensus for a scripting language than for graphical user interfaces. Also, most instrument scientists are willing or able to control their machines by scripts.

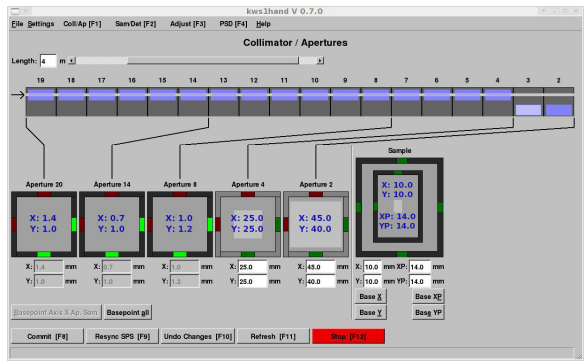

Figure 2: Example of an existing GUI TACO client: manual control of the collimation line of a small angle scattering instrument, implemented in C++ and Qt3.

Thus the scripting language definition is almost completed, and a preliminary implementation was done, while we still have varying ideas and ongoing discussion about the GUI design.

To some surprise (and disappointment for programmers wanting to bring in modern technology) instrument scientists agreed on a macro language which simply consists of fixed command words and their arguments. No control structures were considered necessary, and syntactic elements should be reduced to a minimum. (So the technically simplest idea – use of Python as primary macro language – was out of question. There is the "cmd" module which allows to integrate such a simple language with Python, but implementation details are out of scope for this article.)

All axes, temperatures and other parameters should be handled similarly: They have a logical name, can be moved to some abstract position, and the status can be read. This brings some object oriented semantics, and allows to have only a handful of primary command words.

One syntactic finesse was necessary due to a specific requirement: Because mechanical axes of neutron scattering instruments often take a long time to position, it should be possible to move more than one in parallel. This lead to

```
move omega 7.5; move detector 5
```

meaning successive positioning, while

```
move omega 7.5 detector 5
```

moves both simultaneously. (Semicolons are command delimiters just like line ends.)

As mentioned above, measumements are often done as "scans" with some varying parameters. For simplicity, it was considered sufficient to limit the ability of the macro language to scans with at most two dimensions. This can be handled implicitely within a command word, without need for loop constructs at language level. So

```
scan <device> <from> <steps> <to> <t>
```

would describe a one-dimensional scan, with <from>, <steps> and <to> being the loop parameters and <t> the measurement time for each position.

This implies that other parameters, e.g. for the detector to use, are given by the context, by a previously issued

```
configure <device> <args>
```

command. (Obviously, the “configure” command is massively polymorph. because the required parameters differ.) Since configuration commands can be harmful to the instrument if done wrong, there will be “access levels” to restrict such actions to experts.

For GUI development, some proposals are being circulated. There are different approaches: Some prefer a uniform top-level structure leading through a number of panes, from instrument configuration through sample setup to measurement. Figure 3 shows hoh this could look for an existing instrument.

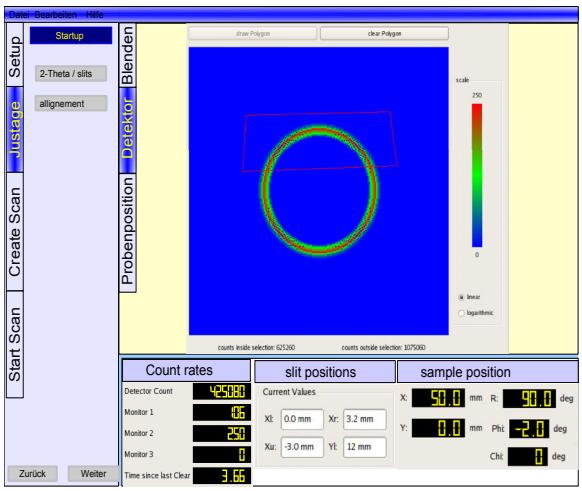

Figure 3: A proposed layout for the future instrument control GUI for all Munich JCNS instruments. The user is guided through the measurement by vertical menu bars at the left side.

Others like a big canvas which allows to combine a number of interface widgets, so that the user has everything needed in view at the same time. Figure 4 shows an example of the latter kind.

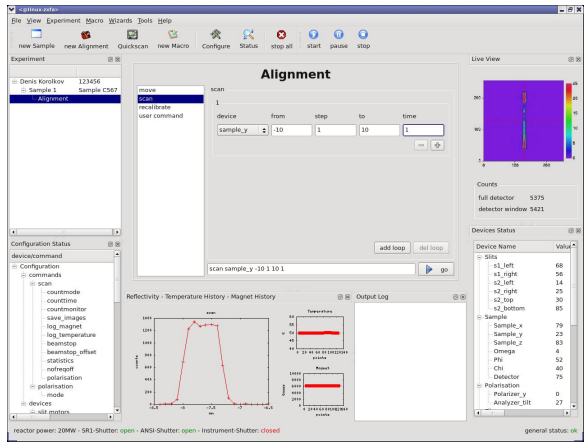

Figure 4: Another proposed GUI layout. Various components can be arranged within a window.

There is no final decision yet, and implementations are still at mock-up level.

REFERENCES

[1] M. Drochner et al., Relocation and Reconstruction of the Jülich Neutron Scattering Instrumentation - Challenges and Plans, PCaPAC2005, Hayama, Japan, 2005.