Common Command Line Interface to Instruments
A Proposal

Mark Könnecke
Laboratory for Neutron Scattering
Paul Scherrer Institut
5232 Villigen–PSI
Switzerland

November 2, 2004

Abstract

With this contribution I want to propose a collaboration whose aim it is to develop a common command line interface to instruments. Such an interface would provide for a common user experience across sites and would provide a basis for common graphical user interface developments. Some thoughts will be presented about the scope and content of such a standard, a possible path to an implementation and why command line interfaces are not yet obsolete.

1 Introduction

Currently, the travelling scientist finds different instrument control systems at different sites. In order to perform her experiments, she has to learn all of them. This wastes a lot of time on the learning curve and beam time spent on failed experiments due to usage mistakes. There is also no real need why the world needs x different ways to express say a scan. With this paper I propose to define a common command set for instrument control. Users would only need to learn one system and can use it wherever they go.

2 Why Command Line Tools Are Still Useful

One common argument against this proposal will be that user want graphical user interfaces today. Command line interfaces are viewed as old fashioned. But there are good reasons why text based tools may still have a business case in data acquisition.

According to experience, only 50-80 percent of measurements performed at large facilities are standard measurements. For standard measurements it is easy to provide a nice and friendly graphical user interface. Though it takes time to develop such interfaces. Interesting science can be done with standard measurements if interesting samples are provided. However, there remains the realm of non standard measurements. In this realm, full control of the instrument is required. This is most easily be achieved with the expressive power of a command line based tool. It is not sustainable to tell a user: you cannot do the experiment, there is no button in the graphical user interface for it. And it is with those non standard measurements where the best science is made.

There are types of instruments where the user interacts very heavily with the instrument during experiments. Typically the user performs some measurement and based on the results, adjusts parameters or decides upon a new measurement strategy. This style of instruments are called exploratory style instruments. A good example is the neutron triple axis machine. The versatility of such instruments is very difficult to capture in a graphical user interface. For such instruments the full expressive power of the command line is still needed.

Another good reason is that, in the humble opinion of the author, instrument control IS programming. At the end of the day, a user wants to write a batch file and have the instrument do the work for her while she is off for dinner and a good nights sleep. Users and instrument scientists alike wish to write little programs which automatise all or parts of the instrument operation. Users wish to write ad hoc measurement procedures in order to do that elusive special scan. There is also the need to integrate some hardware a user brought very quickly. Again, a command line tool which can communicate with the device and wrap it with a useful procedure excells.

A common command line language would also follow a common design pattern for good software: the domain language design pattern. CCLI would provide a basis on which higher level functionality can be built.

Summing it up, any time we do something more complicated then selecting something, hitting a button or fill a form the expressive power of language is required. A visual programming tool may be equally expressive. However, visual programming has not really picked up. This may be caused by the fact that a graphical represenation of a program may not be compact enough when the program becomes to complex.

3 The Common Instrument Control Command Set

The main part of the proposal is to define a common command set independent of any actual implementation. This command set should include commands for:

- Driving real and virtual motors. Virtual motors are a convenient and elegant way of expressing coordinated movements of several motors.
- Counting
- Scanning in one, two or mode dimensions. Propbably a variety of scans has to be implemented on top of a more general scan facility.
- A flexible and user controllable logging system. The amount of logging necessary varies strongly with the preferences of the people involved with the measurement.
- We need to save our experimental data.
- When aligning an instrument it is quite useful to have procedures for the automatic location of peaks and for optimizing them. Also rough estimates of intensity and full width at half maximum are useful in many cases.

- Triple axis spectrometer and four circle diffractometer calculations
- It would be useful if plots of the current data can be generated and exported in some standardized format to a display system.
- Another common occurrence is that a user brings a piece of hardware and wishes to control it during the experiment. Most people are ever so happy when they can send ASCII commands to the device from the control software through some bus, be it RS-232, TCP/IP, GPIB or whatever. Therefore commands should be provided to initialize and communicate through some bus. The buses available will naturally vary from site to site. But there is no need to have very different interfaces to each.
- There must be commands for controlling batch files. This should not only include simple batch file execution but also informational commands which allow to find out what exactly is executing at any give moment.

Of course, this list is only a suggestion and has to be discussed and possibly extended.

Besides the commands a common command line interface should also provide for some other desirable feautures:

- Parameters should be persistent. Thus the machine should start up after a software or hardware failure with the same parameters as before.
- The system should allow for client server operation. This implies three different usage modes. These are:
  - Control mode. In this mode the instrument is controlled.
  - View mode. In this mode it is possible to view parameters but not to change anything.
  - Calculation mode. In this mode all calculations and checks are performed but no hardware is moved. This is useful for testing batch files.
- Currently several mature and flexible scripting languages are available. Therefore there should be no need to reinvent procedures, loops and control structures but rather use one of the existing scripting languages as a base. Which one has to be discussed. The author would prefer Tcl (Tool Command Language) because it comes with a shell like syntax.
- Of course an online help system is required.

4 CCLI Hardware Access

A common command set is only half the deal. What use is it if the command set is known, but a user cannot find out which motors and other devices are available at a given instrument? Thus the idea of a common command set would be supported very much by a convention for naming hardware objects. For most devices this can be covered by giving them descriptive names. Motors are more of a problem. For motors the author proposes:

- Define an instrument coordinate system, for instance the McStas system.

- Define a prefix according to position in the instrument
- For translations:
  - With all rotations zero
  - Name according to the axis moved
  - If more then one motor on a axis in a give position, append l for left, r right, t for top, b for bottom.
- Rotations: name according to rotation axis.

However, naming conventions for motors are difficult to agree upon. Therefore a standardized document should be available for each CCLI supported instrument which lists the devices and their function within the instrument. This information should also be available from an online help system. In order to facilitate the porting of batch files, it is desirable to define aliases for devices. Then porting a procedure involving devices would just require to assign aliases for functionally equivalent devices at the new instruments to the names used in the procedure to port.

A system which has to deal with hardware is confronted with a varying and huge number of hardware related parameters. Standardizing names for all such parameters would be a task which is next to impossible. However, if we restrict the scope of the common command set to those parameters which are usually in the user domain, the task becomes manageable. As an example, consider a motor: A user needs to see zero points, hard and software limits and the sign of the motor. Perphaps a means to reset the motor in case of errors is useful as well. But parameters such as encoder assignements, gearing ratios or ramping parameters can be left to a local system or a local command set. For each hardware device the parameter set which should be in the user domain has to be discussed.

5 CCLI Implementation

A common command line interface should be implemented on top of an existing local data acquisition system. The CCLI then will communicate through TCP/IP sockets, pipes or other inter process communication systems with the data acquisition system. Thus CCLI would be an additional layer which does not break any operational system. Of course, the choice of the underlying scripting language is crucial. In the best possible case the CCLI would just be a thin syntax adaption layer on top of an existing capable data acquisition system.

In other cases there might be functionality which is missing. Such functionality should be provided in a CCLI–library which can be integrated into the CCLI system.

In the worst case a CCLI core system must be provided which implements the command set with the help of the CCLI–Library. Then CCLI would communicate with the hardware through a faily dumb interface.

In any case a client server adapter should be provided. This adapter provides for the integration of the CCLI into integrated data analysis and simulation tool chains and the remote operation of instruments.

6 Conclusion

A common command line interface to instrument would be a worthwile effort if enough sites participate and implement it. The following benefits will be expected from a CCLI:

- Users learn one system, use everywhere.
- Reduce instrument scientist workload because users will be better trained eventually.
- Make better use of instruments because users need to spend less time on the data acquisition systems learning curve.
- A CCLI can be the foundation for shared graphical user interface developments.
- A CCLI can also serve as a foundation for exchanging beamline alignment procedures and other instrument control procedures.
- A CCLI can also serve as a means of modernizing legacy systems.
- A CCLI can also be a first step in the implementation of a new control system.

The author suggests to found a working group in order to define a CCLI. The goal is to present a completed definition and perhaps a prototype system at NOBUGS 2004. Persons interested in joining into this effort should contact the author.