Instrument control software development at FRM-II

Jens Krüger

October 19, 2004

Abstract

The standard instrument control software at FRM-II is highly modularized and based on TACO (http://www.sf.net/projects/taco) and NICOS (http://www.sf.net/projects/nicos).

This design allows the instrument scientists to adapt their instrument control software for special cases very easy.

TACO is used for the access to the hardware and NICOS works as glue between the hardware modules.

It will be explained how simple extensions to the instrument control software may be added and how new hardware may be included.

1 Introduction

The standard instrument control software at FRM-II has to fulfil the following paradigms:

1. it has to be network based
2. it has to be flexible in type of hardware and type of instrument
3. it has to use only a reduced set of device types with a common set of commands for each type
4. the hardware should be replaceable without any changes in the instrument control software

The software is based on two components:

- TACO device drivers

- NICOS methods

Since both systems are client/server systems the instrument control software is naturally network based. The network based architecture gives the flexibility to choose the right hardware components for the system. The only boundary condition we have is the existence of a network interface. Hardware may itself have the interface, or it is integrated in a computer which have a network interface.

The TACO device drivers are directly connected with the hardware, and are implemented as RPC-Servers which may run on different machines. Due to the standard TACO command interface you may interact with the drivers from different clients and you may move the drivers from one computer to another without changing the Software. The only think you have to do is a small change in the configuration.

The TACO device drivers implement only a special set of commands which should be executed by the hardware. From the point of view of the instrument control software it is not useful to implement all features of the hardware in commands. In praxis it has been shown, that not all features are really used. It is much more helpful to restrict the number of commands and implement the features internally, or make them available by configuration parameters.

2 Set of Functions

We, at FRM-II, decided to define a set of commands/functions of each device type which are typically for this device type. So has a device for output of digital signals really two commands:

- write a discrete value
- read the written value

From the view of software it makes no difference what the actual hardware is. It may be a builtin computer card or a fieldbus module or whatever. The difference in access to the hardware is hidden in the implementation of the drivers. If the current hardware breaks and has to be substituted by another you have only to change the setup and the perhaps to write a new driver. There's no rebuilding of software necessary. There is no really need to store a pool of hardware components which are builtin into the experiments.

Not only the TACO device drivers are working in this way, also our NICOS works in this way. It has itself a set of commands, so called "NICOS

methods", which handle the communication between all hardware components. In difference to the TACO commands the NICOS method should primarly handle more complex components of our instrument control like monochromators and detectors. These components are intrument specific, but the set of commands isn't it. A monochromator has only some functions:

- selection wavelength or energy
- focussing the beam in some directions

The way to get this may differ from instrument to instrument, but with the help of this unique set of commands it will be possible to write more generalized function like scans.

3 Why TACO and NICOS?

Whereas the TACO devices implement the hardware access the NICOS methods are used to develop more complex devices like monochromators and detectors, in which a lot of components work together.

NICOS is written in Python (http://www.python.org). So gets the instrument scientist the possibility to integrate such complex systems into his specific software directly with out the help of computer specialists, and may also change some things are not working correctly in a simple way.

4 Development of a new TACO device driver

The development of a TACO device server from scratch is an exhausting job, since you have to write a lot of lines of code for the TACO framework before you could concentrate to the programming of the device specific parts. The second problem you have is the definition of the command numbers and parameters of the command. To overcome this problem we developed a database driven system "tacodevel" for defining and creating new TACO device servers. This system helped us to write and test about 10 new TACO device drivers per week with 2 persons.

The "tacodevel" system consists mainly of three parts:

- database
- GUI
- code generator

The GUI allows to define:

- commands with its paramaters (including the type)
- devices with a set of commands (which have to be defined in the database) and the client counter part
- device servers with a set of provided devices

From this definitions a code generator creates a code framework (including all files necessary for autotools) which may be directly compiled and distributed. The only thing is missing is the code inside the commands. This has to inserted in some functions and after them you may test the TACO device driver.

Since the client normally already exists on your target due to the standard device command setup, you have only to install the driver on the host machine and configure it.

The use of "tacodevel" at FRM-II has reduced the zoo of device types and uncontrolled growth of commands.

5 Development of NICOS methods

The term "NICOS methods" suggest a set of functions, but it is more in a sense of classes. The term method comes from the time were NICOS was born. The idea was to create a set of functions but it was shortly after clear a set of classes would be better, but the term was kept.

All objects in NICOS are derived from a base class "Xable". The next layer of classes are the "Switchable", "Moveable", "Countable", and "Readable". In this categories may all of our compononents to be devided.

The instrument scientist has to overwrite the corresponding default class methods by its own and all should work fine after fixing the bugs.

6 Software development tools

The software development at FRM-II uses mainly the following tools:

- autotools for software projects mainly written in C/C++/FORTRAN
- cvs as software configuration management tool
- Gnu Compiler Collection (GCC)

- Python
- Make (GNU, BSD)
- Qt framework to write portable software for Un*x/Linux, Mac OS-X, Windows

7 Conclusion

The instrument control software at the exististing facilities has to be flexible, simply to modify. It was show that with the current environment at the FRM-II this may be reached.

The definition of sets of devices with a fixed set of commands may improve the quality of development and maintainence of the instrument control software. Not each feature of hardware should be integrated in the set of commands. It is better to hide the feature in the setup and code.