ADOPTION OF THE "PyFRID" PYTHON FRAMEWORK FOR NEUTRON SCATTERING INSTRUMENTS

M. Drochner, L. Fleischhauer-Fuss, H. Kleines, M. Wagener,
S. v. Waasen, FZ Jülich / ZEA-2, Germany
D. Korolkov, Bruker AXS GmbH, Germany

Abstract

To unify the user interfaces of the JCNS (Juelich Centre for Neutron Science) scattering instruments, we were adapting and extending the "PyFRID" framework. "PyFRID" is a high-level Python framework for instrument control. It provides a high level of abstraction, particularly by use of aspect oriented (AOP) techniques. Users can use a builtin command language or a web interface to control and monitor motors, sensors, detectors and other instrument components.

The framework has been fully adopted at two instruments, and work has been started to use it on more.

INTRODUCTION

After the initial construction phase at FRM2, where we had to get the instruments into operation quickly, instrument operators and users expressed the wish to have a more unified user interface at the various instruments. As described in [1], a kind of poll was taken, and a simple scripting language was defined which is powerful enough to do common measurements yet almost intuitively to handle. As an implementation of that language, the "PyFRID" [2] project was started, implementing not only the command line parser but also the semantics of the commands and other components which are needed to build an instrument control system. The "PyFRID" system is deployed on the "BioDiff", "MARIA" and "KWS3" instruments.

In the meantime, another instrument control project called "NICOS2" was started by a competing group at the same facility (actually based on an old and abandoned project called just "NICOS"). Both "PyFRID" and "NICOS2" are modern, Python based implementations of a control system framework. Functionality is comparable, as are their logical positions within the overall control system structure.

Recently, a decision was taken, on grounds which are not technical but mostly based on developer manpower, in favour of "NICOS2". While there are no immediate plans to migrate the three instruments mentioned above to "NICOS2", there will be no further developments for "PyFRID" beyond basic support. So this article can be seen as a concluding report about our "PyFRID" activities.

APPROACHES FOR MODELING COMPLEX SCATTERING INSTRUMENTS IN CONTROL SOFTWARE

Scattering instruments as we are concerned with generally consist of a large number of mechanical and electrical components which need to be accessed at a level determined by engineering. Examples are stepper motor or encoder pulses (for positioning of components), voltages or currents (for magnetic fields), or speeds and phases (for choppers). For practical use however, units need to be used which make sense to the physicist operating the instrument. Here we can have various levels of abstraction, e.g. common units of measurement (millimeters, or Tesla), or application specific terms like momentum transfer and reciprocal lattice coordinates. Also, simple components are often grouped into more complex subsystems, as a collimation line comprising multiple elements and apertures, each aperture in turn made of four individual blades, driven by a stepper motor each. This suggests a hierarchical, tree-like view of the instrument, with the leaves being elemental components and increasing abstraction towards the root. Since control systems are usually distributed, with some network transparency, the control system developer has some choice at which levels of the hierarchy network protocols are used, and where just programming APIs.

TACO [3] and TANGO, which are used as middleware in our instrument control systems (see [4] for a higher level overview) encourage "stacking" of device servers. This means that servers which implement more abstract functions (e.g. the aperture in the example above) are by theirselfes clients to one or more other device servers which operate logically closer to the hardware. User interface programs typically use the topmost layer of servers, but for diagnostic purposes it is possible to access any layer. This scheme allows for modularity and avoids duplication of efforts, a function implemented in a TACO server can be easily reused. The downside is additional complexity. A user action leads to multiple network requests which are translated into each other; diagnostic and error handling becomes challenging.

In our systems, we have thus avoided complex server stacks and implemented device interaction, calculations etc. in the client programs. Duplication of efforts can hardly be avoided this way – each client dealing with axes does the translation from machine units into physical coordinates itselfes. This is somewhat mitigated by use of common libraries for these functions, and instrument specific parameters located in the TACO/TANGO database. If

ISBN 978-3-95450-139-7

multiple programming languages are used, multiple implementations of analogous functions cannot be completely avoided though.

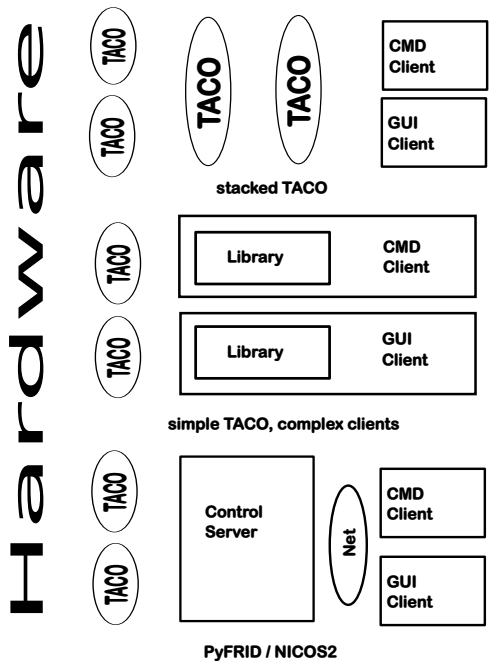

Figure 1: Different ways to organize control software.

A system built on PyFRID (or NICOS2, for that matter) uses a topology which shares parts of both these approaches, trying to avoid the problems mentioned (see Fig. 1). All instrument specific functions are dealt with within the PyFRID instance, which is a single Python program. Access to hardware uses only basic functions; besides TACO/TANGO support, it is possible to interface to other access methods. A PyFRID instance can support multiple user interfaces, for command line or web based control.

STRUCTURE OF "PYFRID"

The installed PyFRID software consists of a set of Python libraries installed at the standard location for external packages ("site-lib"), and a command line utility to generate and manage instances. The libraries contain the code which implements the script language and the standard commands within it, device handling and modules which implement authentication and logging in addition to the basic application framework. Libraries for access to specific devices and the web server component are distributed as separate packages.

A PyFRID instance for an individual instrument can be created using the command line utility ("pyfrid-admin"). This resembles the layout and philosophy of the popular web application framework "django" ([5]). As in "django", a directory structure for the instance is created in the user's current directory, containing template files and another command line utility to manage the new instance. With the new command line utility, new commands and devices can be added to the PyFRID instance; templates and glue code are added to the instance tree automatically, pulling in components of the central library as required. The user then needs to adapt the generated template code to the specific needs of the intruments.

For a simple motor device, as an example, adaption only requires to fill in the "position" and "status" member functions of the generated class, with code which connects to the actual hardware. Device accesses can be done in multiple threads. To use this, the device access code needs to be thread safe. (This turned out to be a problem with the TACO-Python binding which is part of the TACO distribution, but it was possible to fix with little effort.)

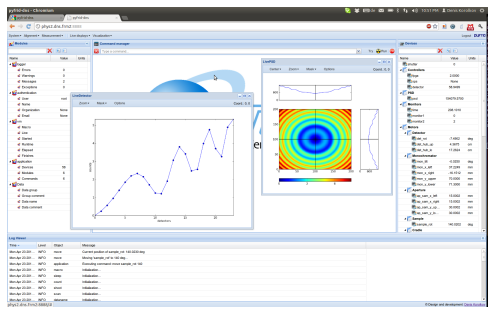

Figure 2: Screenshot of the web UI.

For the web interface (see Fig. 2), the popular "tornado" web server libary ([6]) is used. The application allows to visualize and control the devices in the system, using client-side JavaScript code. Communication between the client web UI and the PyFRID server is done through a private protocol, which can in principle also be used to connect other client UIs.

CONCLUSION

PyFRID is a novel, modern approach to control systems for spectrometers and similar instruments. It utilizes the power of the Python programming language. This leads to compact, elegant code. While it will not be further developed at our institute, its ideas are worth keeping for future developments.

REFERENCES

[1] M. Drochner et al., New Developments for the JCNS Neutron Scattering Instruments, ICALEPCS 2009, Kobe, Japan, 2009.

[2] PyFRID software, code at https://github.com/pyfrid, documentation at http://escape-app.net/html/

[3] TACO control system, http://www.esrf.eu/

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

[5] Django Software Foundation, Django Web Framework, http://www.djangoproject.com

[6] Tornado Python web server, http://www.tornadoweb.org

ISBN 978-3-95450-139-7