A. Pazos[1], P. Duval[2] and S. Fiedler[1]

[1]EMBL, Hamburg, GERMANY, [2]DESY, Hamburg, GERMANY

SCRIPTING TOOLS FOR BEAMLINE COMISSIONING AND OPERATION

Abstract

Scripting tool capabilities are a valuable help for beamline commissioning and for advanced user operation. They are the perfect complement to static Graphical User Interfaces allowing one to create different applications in a rapid way. A light middle-layer for scripting support has been foreseen for the EMBL structural biology beamlines at the PETRA III synchrotron to provide 'controlled' rather than 'direct' access to the control system devices. This prevents conflicts with the control system and allows control of the supported operations. In order to account for the wish of different scripting languages by the beamline scientists an extension of the scripting capabilities of the TINE [1] control system has been implemented. To the existing shell support, a Python extension (PyTine) has been added and a Perl wrapping has been also prototyped (tine4perl).

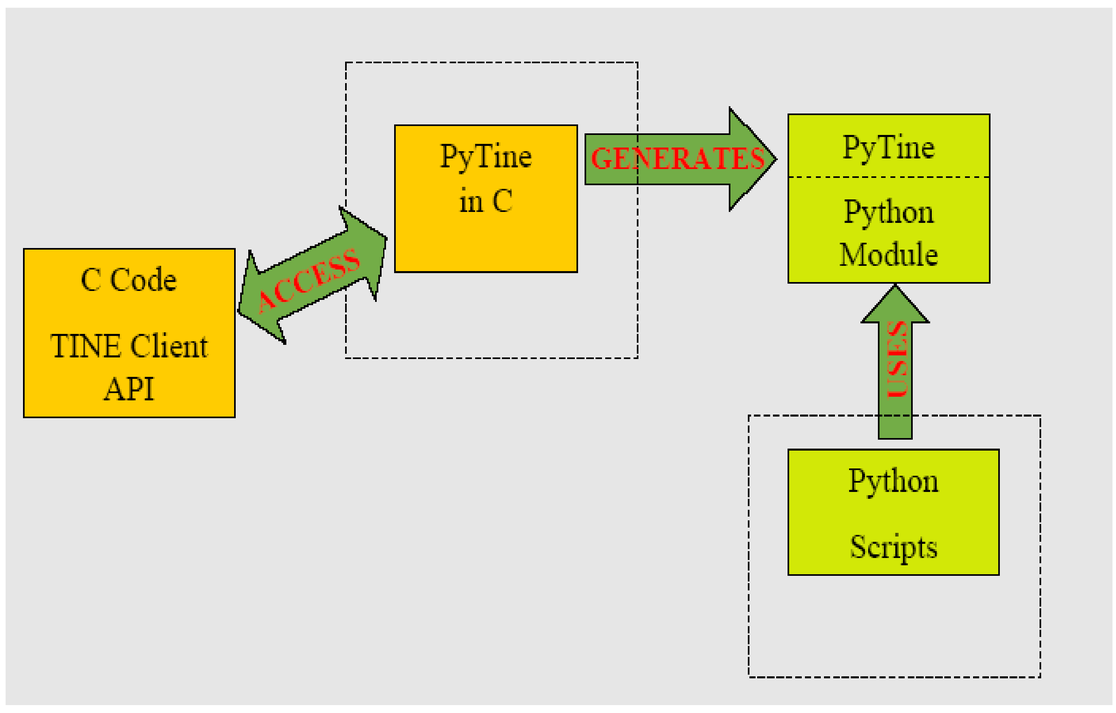

Figure 1. PyTINE implementation overview. The PyTINE library wraps the TINE C API. Different Python modules are used as middle layer for the user scripts.

Requirements for a Scripting Tool

- Easy to learn (for the developers and for the users)
- Easy to maintain
- Flexible (in contrast to a GUI)
- Dynamic (does not need variable declarations)
- Well defined syntax
- Well documented
- Possible to control the accessible functionality
- Separated of the device specific layer
- Command-line support
- Sequencer support
- Reliable
- Secure
- User proof
- Multi-platform
- Open-source

Why Python?

Apart of fulfilling the list of requirements we have found a list of benefits for supporting Python [2] as our scripting language:

- It has object oriented possibilities
- Is getting more popular inside many scientific communities
- It is also a powerful programming language
- Multiple open source libraries available
- Also possible to compile and to create executables
- Extendable and embeddable
- Graphical support (PyQT [3] and others)
- The EMBL@PETRAIII GUI used at our MX beamline (MxCube [4]) is based on Python
- Possible to interface with Labiew^{TM} [5]

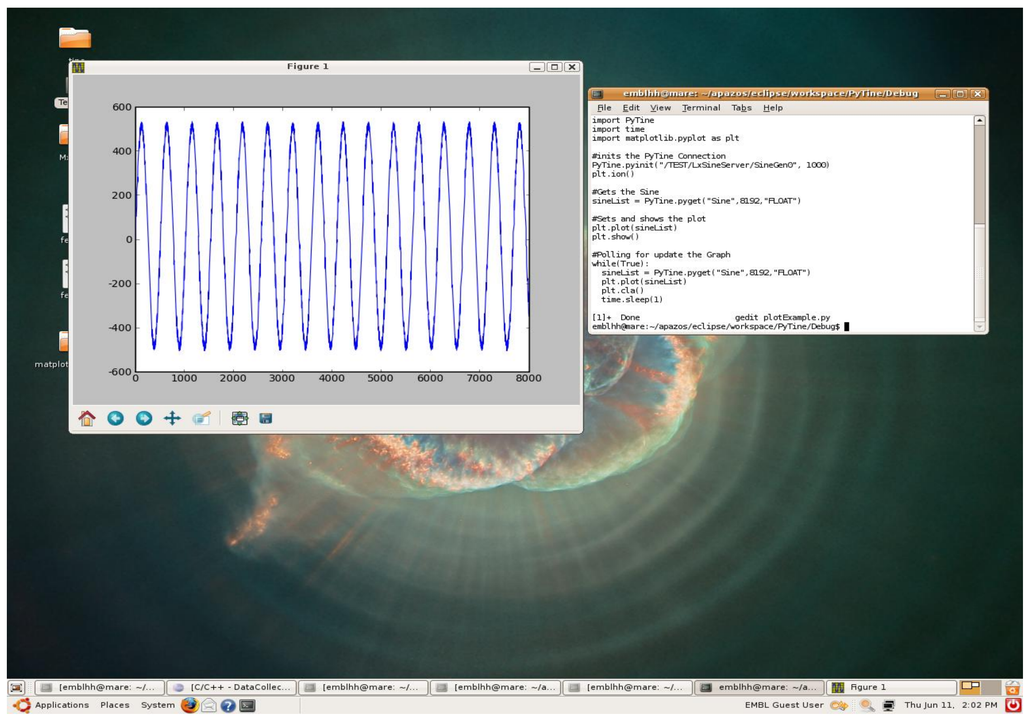

Figure 2. PyTINE example invoking the TINE sine server

Software Overview

Different scripting tools have been created and extended for the beamline commissioning and for the customized user operation:

- TINE shell tools: set of functions meant to build shell scripts both in Linux and Windows
- NEW PyTINE
  - Full wrapping of the TINE C client API (see Fig. 1)
  - Native bindings using the Python.h library
  - NEW Extended functionality available: callbacks, data structures, plot support, etc. (see Fig. 2)
- TINE4PERL
  - Interface between TINE and Perl [6] to get and set data from the device servers.
  - Implementing using the automatic translator library SWIG [7]

Conclusion

A scripting language (Python, Perl, etc.) is more suited to perform certain tasks than normal system programming (C/C++, Java, etc.). We have seen in our applications that, used together, they can create a very powerful programming environment fulfilling different kinds of requirements.
A scripting language should be as simple as possible. It is sometimes beneficial for security reasons not to provide direct access to the system but to use a middle layer controlling the access to the device servers.
It is important to evaluate very carefully the existing wrapping solutions, including automatic converters when supporting a new scripting language. Depending on the desired functionality it might be better to use one method or the other. On the one hand, the use of an automatic converter for complex implementations, that include pointers and data structures, can be tedious and can require learning a special syntax. On the other hand, it can create fast bindings for simpler wrappings.
In our environment, where all the software is integrated in a control system, flexible and open systems allow the extension of their functionality and the support of new programming languages.
This scripting concepts and architecture can be extended and applied to different environments and integrated with different control systems.

References

[1] P. Bartkiewicz and P. Duval, “TINE as an accelerator control system at DESY”, Meas Sci Technol, 18:2379–2386, 2007
[2] Python Programming Language, www.python.org
[4] PyQt White Paper, www.riverbankcomputing.com
[4] J. Gabadinho et al., “MxCuBE: a synchrotron beamline control environment customized for macromolecular crystallography experiments”, J. Synchrotron. Rad., 2010, 17, 700-707
[5] Labpython, Open Source Python tools for LabviewTM, http://labpython.sourceforge.net/
[6] Perl Programming Language, http://www.perl.org/
[7] Simplified Wrapped and Interface Generator (SWIG), http://www.swig.org/

Contact: **andres.pazos@embl-hamburg.de**

EMBL-Hamburg Instrumentation Group