A GRAPHICAL SEQUENCER FOR SOLEIL BEAMLINE ACQUISITIONS

Gwenaëlle Abeillé, Majid Ounsy, Alain Buteau
[1]Synchrotron SOLEIL, Saint Aubin, France, http://www.synchrotron-soleil.fr

Abstract

Addressing batch-processing and sequencing needs are fundamental for daily beamline operation. The SOLEIL [1] control software group proposes two solutions.

Firstly, the PYTHON [2] scripting environment, for which a dedicated TANGO [3] binding is available, has been proven to be a powerful resource, but is limited to scientists with good programming skills.

Secondly, we provide the PASSERELLE software, developed by the ISENCIA [4] company and based on the PTOLEMY [5] framework. Within this environment, sequences can be designed graphically by drag and drop components called actors (representing elementary sequences steps). The process execution can be easily "programmed" by graphically defining the data flow between actors. On top of this framework, an existing generic GUI application allows users to configure and execute the sequences. A dedicated GUI application can also be developed to provide the beamline's end-user a completely integrated acquisition application.

The work organization, software architecture, and the design of the whole system is presented, as well as its current status of deployment on SOLEIL beamlines.

THE CONTEXT

The SOLEIL light source is a new low emittance 2.75 GeV electron storage ring which was commissioned in 2006 at Saint-Aubin near Saclay, not far from Paris (France). This facility provides high-intensity photons covering a wide spectral range from ultraviolet light (UV) to hard X-rays. SOLEIL will serve an international community of scientists from many fields including physics, materials science, chemistry, and biology. Ten beamlines are being commissioned and will start regular user operation by the end of 2007.

Under normal operations, external users will be on site from 8 hours to several days to perform data acquisitions to provide structural information concerning their samples. Upon arrival, most users will have no knowledge of the beamline and its control system. Thus the challenge is to provide them a software tool:

- for performing acquisition sequences in a simple and flexible way,
- that conceals the complexity of the beamline and its control system,
- that helps beamline automated alignment to shorten the beam reconfiguration between successive experiments,
- that guarantees "high availability" since no beam time should be lost because of software problems.

To summarize, have only to click a "Start Acquisition" button to collect data.

OUR VISION

Intrinsic Problems of Scripts

Using scripting languages is the common way in Synchrotron facilities to address all previously evoked needs, but in reality has a number of severe limitations.

First, users have to learn a script language which can be quite painful if they are not software developers. Since users stay only a short time on a beamline, it is impossible to relearn a new language each time they go to another Synchrotron. Some collaborative work on "scripting language homogenization" has been occasionally tried by various institutes. Unfortunately, the same language does not always translate into the same commands behaving in a standardized way across sites. For instance behind a single command line like "scan 100 200 1" that launches a script (or several), many different behaviours may arise depending on the underlying Control System, equipments, execution context, etc ..

Moreover each time this script(s) is changed by someone (*i.e.* to add a new functionality) the command line behaviour is potentially and often inevitably altered. This apparent flexibility quickly raises stability problems. What was working before the modification may fail during command execution. Precious resources are thus dedicated only to keeping the control system in a stable state.

Another limitation of scripting languages is that it is quite difficult to follow what the script is currently doing or what has been done in a previous execution. It is also impossible to remotely perform batch-processing administration.

Last but not least, scripting languages do not natively provide services for: logging, execution simulation modes, error management, and context checking. Users have to handle these tasks by hand, generally waiting for their scripts to complete execution and checking if anything is wrong with their collected data.

This model of operation maximizes the risk of lost beam time.

Our philosophy: leave scripts in experts hands and only for limited applications

Script languages are nevertheless used at SOLEIL. For instance, the TANGO Python binding is widely used by the software team to carry out unit and regression tests. It is also used on two beamlines for commissioning applications. In both cases, the individuals using these Python scripts have good programming skills, skills which are not available within each beamline team. Using python at SOLEIL over the past six months has demonstrated that it was not manageable to use it as an

integrated environment for hosting beamline operation by external users.

Keeping in mind these intrinsic limitations, we studied the market to find a ready-to-use product that could meet our requirements: flexibility, robustness and GUI integration for the development of acquisition sequences. This led us to the PASSERELLE framework from ISENCIA company.

PASSERELLE OVERVIEW

Passerelle is a toolkit for designing sequences (and more generally data workflows) in a "drag and drop" graphical environment. Its core functionalities are based on the Java technology standards.

To design/develop/program sequences, called models, ISencia provides a graphical IDE (Integrated Development Environment) (see Fig. 1). The IDE offers an execution engine as well as and a number of essential framework services which will be described bellow.

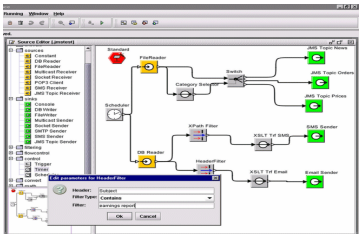

Figure 1: The Passerelle IDE

Using Passerelle graphical IDE, a given process can be easily mapped from its functional design onto a graphical model of inter-connected components. The solution model can be gradually refined, starting from high-level composite components, to define how each composite component can be assembled from more elementary building blocks. Models can be immediately tested inside the Passerelle IDE, after which they can be directly deployed to the Passerelle model executors.

Passerelle: the basics

Within Passerelle, the user designs models that are composed of "boxes" and "wires". The boxes are called "Actors", and they execute an action. The wires are called "Messages", as they transfer data between actors. The fig. 2 is a very simple model that calculates the sine function of a value and writes the result to a file.

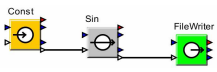

Figure 2: A very simple sequence.

Each actor can be configured with some "Parameters". Fig.3 shows the "value" parameter of the "Constant" actor that acts as an input generator for other actors.

Figure 3: Example of an Actor's Parameter.

Each Passerelle model must be populated with one instance of a special actor called the model Director. The Director handles communication and data transfer between all the other actors. Different types of Directors exist to implement different strategies of communication (e.g. synchronous flow, asynchronous queues, rendezvous...).

The user can also define its own actors inside the Passerelle IDE with a Composite Actor (see Fig. 4) which is a composition of several basic actors.

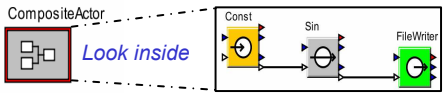

Figure 4: Composite actor

The Execution Control

After developing a model, the user is able to test it in the stepwise execution mode.

A very useful execution environment is the simulated mode. Each actor can be executed live, to control the beamline or in simulation, to check or test a model.

A Passerelle model is able to natively manage errors. The user has just to choose among a predefined set of error strategies:

- Abort on error,
- Pass the failed step,
- Retry n times the failed step.

A context validation service is integrated. This gives the ability to check at each step of execution if some critical conditions are verified; is the beam still present? Is the source power not too low? The user can then decide which action to take: Continue? Stop? Pause?

The Execution Environments

A Passerelle model can be used in various execution engines. The first one, the Passerelle's IDE, supports the design, definition and testing of new solution assemblies. It provides a sophisticated graphical model editor, including the means to pick actors from the library, to interconnect them in an assembly and to configure them. The embedded executor ensures an easy iterative work-flow to define, test and refine solution assembly models.

Another Passerelle execution environment is provided by a GUI (Graphical User Interface) called PasserelleHMI (see Fig. 5). This GUI is able to display a panel on top of any model, giving access to all its configurable parameters.

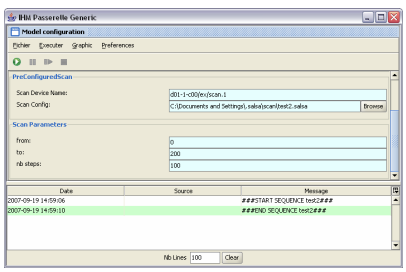

Figure 5: Passerelle HMI.

Run-time Administration

Passerelle also offers an administration console to manage its execution environment. Passerelle's executor management is based on the standard for Java application and component management JMX (Java Management Extensions) [6]. The console is delivered as a browser-based application, implemented on an embedded Java web application server. Thus a Passerelle runtime environment can be easily managed remotely using a standard browser.

Passerelle Integration with our Control System

The Soleil software team has developed a library of actors that are able to communicate with the control system. This library contains actors which abstract all the complexity of the different controlled systems within sequences. For instance, for a complex system like a monochromator (used to configure the energy of the beamline and composed of at least 6 motors) a simple actor with only one parameter, the desired energy, has been developed. All the processes of configuration including the waiting for the end of movements and the error management are hidden. Actors for acquisition processes such as scans, detector acquisition… have also been developed.

PASSERELLE STATUS AT SOLEIL

Current Deployment

Of the first ten beamlines of SOLEIL, eight are currently under commissioning. One beamline is not yet started. Five beamlines are still doing unitary tests of their equipments. Four are beginning to automate some alignment and acquisition and three are using PASSERELLE.

Acquisition Sequences Already Developed

Below is an example of a sequence to perform a scan in fluorescence (see Fig. 6). In this sequence, the beamline is first configured (monochromator and detector) then a scan is performed. Finally the result of the scan is used to calculate the energy inflexion point of the sample.

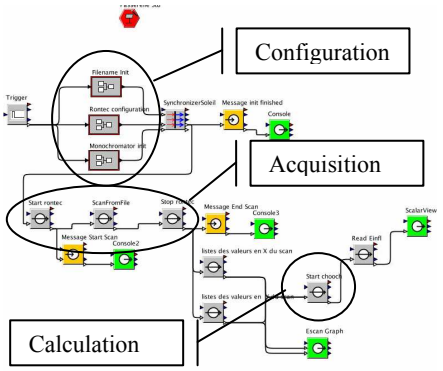

Figure 6: A Soleil sequence.

Higher Level of Integration

We also provide GUI upon Passerelle models because the generic GUI's ergonomics is not always sufficient for the end user. Figure 7 show the panel that configure the model presented in Figure 6.

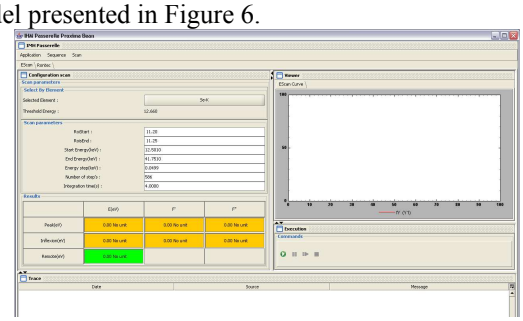

Figure 7: A passerelle GUI for energy scan.

FUTURE OF PASSERELLE AT SOLEIL

For the moment, Passerelle is only used on the beamlines. There are, however, plans to install it also on the accelerators' control system to automate the process of start-up and injection which today are handled manually by the operators.

For the moment the PasserelleHMI is quite basic and the configuration for display of information is quite poor. We provide on-demand specific GUI application on top of specific Passerelle models (programmed in Java). However, the best way should be that the designer of the Passerelle model is also the designer of the GUI. A project is under way to provide such a tool.

The ultimate goal of Passerelle is to be able to automate as much as possible, to achieve a state where all the beamlines could be pre-positioned for specific experiments and the experiments themselves are done automatically. Passerelle will also be connected with online data-analysis systems to do decision-making on the acquisition process.

REFERENCES

[1] http://www.synchrotron-soleil.fr/portal/page/portal/Accueil
[2] http://www.tango-controls.org/bindings
[3] http://www.tango-controls.org/
[4] http://www.isencia.be/content/products/products.html?content=more-passerelle.html
[5] http://ptolemy.eecs.berkeley.edu/ptolemyII/index.htm
[6] http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/