MULTI-PLATFORM PROCESSOR FRAMEWORK FOR DATA ANALYSIS, DATA ACQUISITION AND SIMULATION

N. Xiong, P. Hathaway, T. Lam, N. Hauser, ANSTO, Lucas Heights, Australia

Abstract

The multi-platform processor framework (MPF) is a model-based environment for developing data acquisition, data analysis, and simulation applications for neutron scattering facilities in the Bragg Institute, ANSTO. This open-source project is designed to help developing, integrating and reusing implementations from multi-domains. The processor framework has a data-centric architecture which helps to maintain quality and integration. It provides templates for developers to contribute modules in different domains and in different programming languages. These modules can be put together by recipe files in the deployment or can be created at runtime by the users to perform different tasks. For the user, it provides a convenient way of reusing module blocks. Users that familiar with different programming languages can work together on the same project supported by this framework.

INTRODUCTION

The multi-platform processor framework is a feature project under the Gumtree application software at the Bragg Institute, ANSTO. The Gumtree software is an open source and multi-platform graphical user interface application for performing neutron scattering experiments [1]. As an important feature, MPF provides service to both developers and users in helping them to design data acquisition and data analysis procedures for the scattering experiments. The implementation of the application is in Java language. It is also based on Eclipse RCP* technology. Such plug-in architecture simplifies software development and deployment.

THE MISSION

The users of the neutron scattering facilities can be from different research fields. They can be physicists, chemists, biologists, and so on. They may not be familiar with how to use the instruments, or not even know what is going to turn out as a result. Usually there will be an instrument scientist to help them carrying out the experiment. But it is still a difficult task due to the complexity of the experiment and that the preparation time of each experiment should be very short. Thus, the software we provide to do the experiment must be very easy to learn. On the other hand, because each user will carry out the experiment for a different purpose, there must be a big flexibility in the software.

There are two types of tasks in the neutron scattering experiments, the data acquisition tasks and the data analysis tasks. Each type of tasks has different requirements.

- In the data acquisition tasks, the user interface needs to be simple and neat. It is necessary to use extra protection to prevent the user from making mistake.
- In the data analysis tasks, the user interface can be complex and flexible to meet different requirements. Users should be able to easily contribute their own code to perform data analysis. A powerful mathematics library must be provided. Users should be able to create their own graphical user interface easily as well.

PROCESSOR FRAMEWORK

Concept

The MPF is a project to match the requirements described in the above section. MPF is a number of processors linked together to carry out an experiment task. Each processor of the MPF does a single small work. If chained together, they can carry out a complex task as designed. It is very easy to enable or disable a single processor so that it is masked out from the whole task. There are processor libraries contributed by the software developers and users. A user can simply pick the processors he need, chain them together and run it.

Using MPF, the software application can create certain number of framework instance to carry out different tasks. Some of the framework instances are designed by the software developers, and configured by the users. Such MPF instance can be well tested so that run with higher reliability. Using this type of MPF instances in data acquisition tasks can prevent the user from making mistakes. A fixed graphical user interface is usually coupled with such MPF instance so that the user always has a clean, familiar environment to work with.

Other MPF instance can be designed by the user themselves. A design interface is provided to the users to easily contribute their code. Following the templates, the user simply chooses the type of language he is familiar with. And he can write software code to do analysis on the inputs of the processor, by using the mathematics library provided by the platform. Such analysis procedure creates outputs for the processor. The outputs will then be passed to the next processor in the chain. Such feature of MPF is mostly used in data analysis tasks. Users can use such way to carry out analysis procedures that are not provided at the software deployment time.

Structure

There are three types of basic components in MPF, the processor, the port, and the connector.

* Rich Client Platform, http://wiki.eclipse.org/index.php/Rich_Client_Platform.

- The processor is the holder of the processing code. The processing code can be either generated at deployment time, or at runtime. Code generated at runtime is usually contributed by the user. The processors can also be classified into two types based on their complexity, simple processors and composite processors. A simple processor is designed to only carry out a unit task. Such processor has properties of sharable and reusable. A composite processor is a processor that contains other processors. Use such type of processors to carry out a more complex, but also sharable task.
- The port is the interface to access the fields of the processor. When the processor is processed, the status information such as the inputs and the results are held by the fields of the processor. Use ports to expose these fields so that they become accessible. There are three types of ports. The IN ports, the OUT ports, and the VAR ports.
  - The IN ports access the fields of the processor that are used as inputs. A processor can have as many as possible IN ports. When all the IN ports are set with a value, the processor starts processing.
  - The OUT ports access the fields of the processor that are used as outputs. After the processor gets processed, it will set the OUT ports one by one.
  - The VAR ports access the fields of the processor that are used as arguments. These arguments have default values and can be modified at any time.
- The connector is the link between two ports. The connector has direction. The port that the connector starts from is called *Producer*. The port that the connector ends with is called *Consumer*. When the value of the *Producer* is set, it immediately passes the value to its *Consumers*. A *Producer* can have as many as possible *Consumers*. A *Consumer* can also have as many as possible *Producers* linked with it. Both the *Producers* and the *Consumers* can be any type of ports.
- The framework is a special type of composite processor. It is the instance of the MPF that holds other processors. Only the ports attached to the framework are exposed to be accessed from outside. Figure 1 shows an example of MPF instance. The framework has two simple processors included, P1 and P2.
- Recipe file is the XML description of the MPF instance. The file is human readable. It contains information about the framework structure and the default values of the VAR ports. In the MPF application, the application can create multiple instances of framework from a single recipe file.

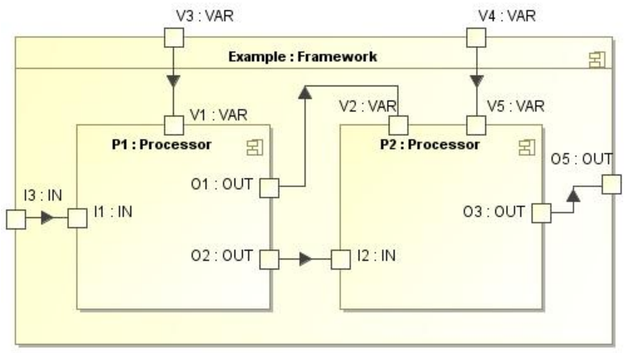


Figure 1: MPF example.

RUN THE PROCESSOR

The processing of an MPF instance is in a single thread. However you can run multiple MPF instances at the same time. In the example shown in Fig. 1, setting a value to port I3 will trigger the processing of the processor chain. If there are more than one IN ports attached to the framework, all of them need to be set in order to trigger the running of the processor chain.

In the above example, port I3 passes its value to port I1, which triggers running of P1. As a result, it will output values to port O1 and O2, separately. Once port O1 gets a value, it passes the value to port V2. That changes the argument field of P2. After port O2 gets the value, it passes the value to port I2, which triggers running of P2. Finally, P2 will create result to port O3. The final result can be reached by acquiring the value of port O5.

In a more complex example, as shown in Fig. 2, there is a composite processor, P4 in the framework. Since there is no IN port exists for the framework, running of the MPF will start from the first processor listed in the recipe file. In this case, it is P4. This task is finally assigned to P1. After processors P1 and P2 get processed, it outputs the results to port O03 and port O05. Since port O05 is the producer of port I01, this will trigger the processing of P1. The chain of P1 and P2 becomes a loop. In each run of the loop, port O04 will pass the result of P4 to port I03. This triggers the running of P3. Port O08 gets a result.

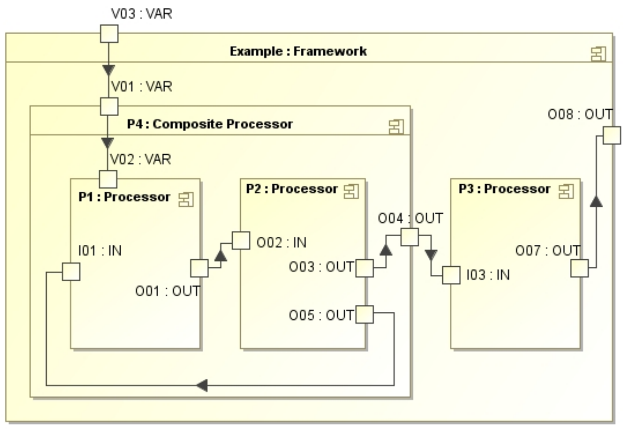
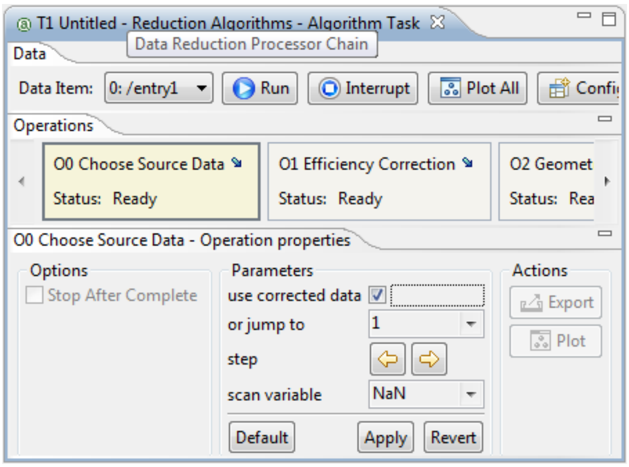


Figure 2: Processor loop example.

GRAPHICAL USER INTERFACE

Figure 3 shows an example of using the generic graphical user interface of MPF. It is about a processor chain to carry out data analysis on neutron scattering experiment data. The chain consists of processors that do the data collection, efficiency correction, geometry correction, and so on. User can use this interface to configure and run the MPF instance. In the GUI, the operation blocks are automatically generated for the processors in the MPF. To change the value of the VAR ports, simply click on the block. The user interface for each VAR port gets created based on the type of the value.

This example shows an important feature of the MPF application. Users only need to contribute code for analysis algorithm. They do not need to care about how to code the graphical user interface, because the application will create it automatically. A design interface is also provided for the user to contribute his analysis code and wrap it with a processor. The GUI also provides tools for user to chain the processors together.



Data Item: 0: /entry1

Status: Ready

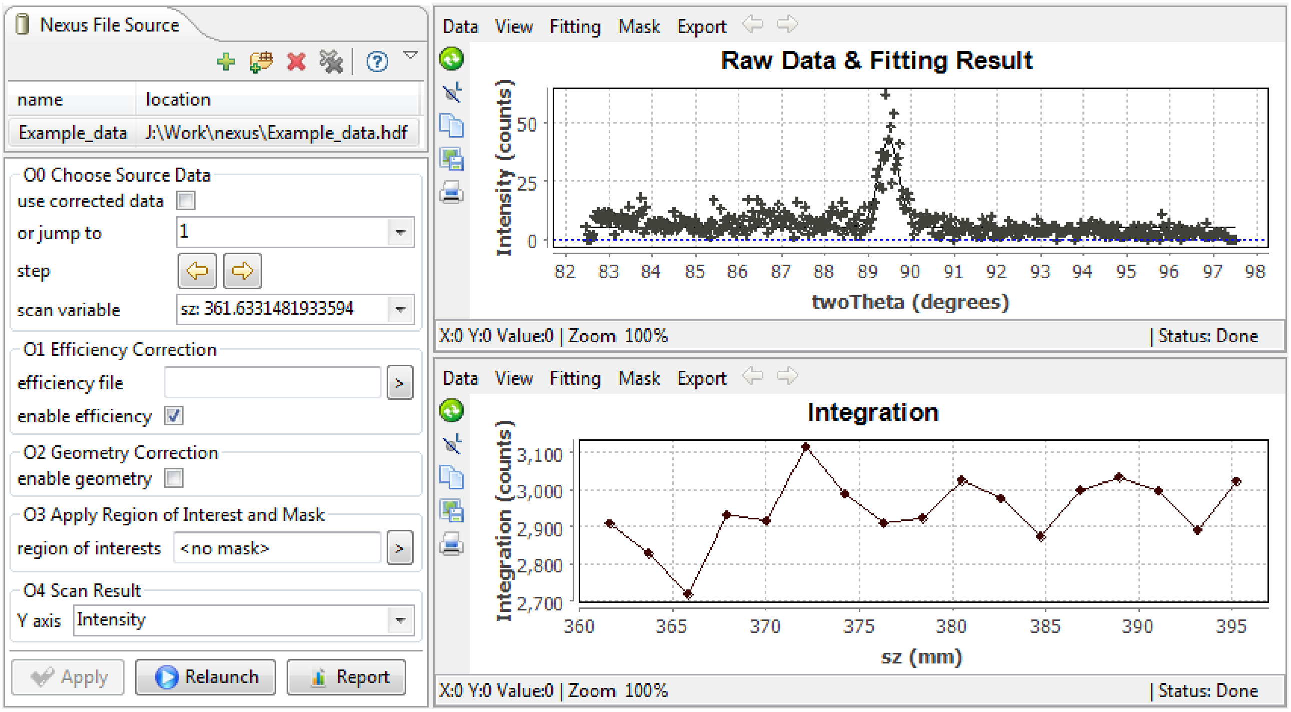
use corrected data ✓

or jump to 1

scan variable NaN

Figure 3: MPF graphical user interface.

Figure 4 shows an example of a customised user interface, which performs data correction and reduction.



name	location
Example_data	J:\Work\nexus\Example_data.hdf

00 Choose Source Data
use corrected data ☐
or jump to 1
scan variable sz: 361.6331481933594

01 Efficiency Correction
efficiency file
enable efficiency ✓

02 Geometry Correction
enable geometry ☐

03 Apply Region of Interest and Mask
region of interests <no mask>

04 Scan Result
Y axis Intensity

X:0 Y:0 Value:0 | Zoom 100% | Status: Done

Figure 4: Customised graphical user interface.

SUMMARY

The MPF is an application to simplify the software designing for neutron scattering experiment. The developers can use this module to provide reliable experiment control and analysis algorithms. It also integrates the user’s contribution easily. This application has been applied in the software for carrying out experiments in four different neutron scattering instruments at ANSTO. Future work about this project is to provide a better design GUI for the user, and to support more programming languages.

REFERENCES

[1] T. Lam, N. Hauser, A. Gotz, P. Hathaway, F. Franceschini & H. Rayner, “GumTree-An integrated scientific experiment environment”, Physica B 385-386, p. 1330-1332 (2006).