Minutes Small EDNA workshop

2008/09/22 - 2008/09/23

Presents: Gleb Bourenkov (EMBL-Hamburg), Sandor Brockhauser (EMBL-Grenoble), Pierre Legrand (Syncrhotron Soleil) and Olof Svensson (ESRF)

Table of Contents

1 Introduction

This small workshop was organised, as decided in the EDNA prototype demonstration meeting, with the aim to advance the XDS plugin. The workshop took place at the EMBL Grenoble. The first part of the workshop was dedicated to a general discussion around the EDNA project, in particular the generic data model. In the second part of the workshop we developed together from scratch a new plugin which uses XDS for performing a spot search.

2 General discussion

2.1 Generic data model

2.1.1 Coordinate system

After some initial discussions via email, we took the decision to propose the following coordinate system for the generic data model:

- Fixed on detector
- Provide templates for known detectors
- Provide input (and output) “plugins” for different coordinate conventions

2.1.2 Convenient functions

Sandor suggested that the EDNA framework should provide “convenient functions” such as transformation between two measurements of the same system e.g. between two close but not necessary identical experimental systems

2.1.3 Crystal orientation

In the current version of the generic data model (XSDataCharacterisatiov01, revision #466) both the A and the U matrix are stored. We agreed that only the U matrix is needed in this data model, since the cell is stored as well in the data model.

2.1.4 Indexing solution

We agreed that the generic data model indexing solution should contain the following information:

- The reduced cell
- The Bravais Lattice Symbol
- The lattice character, or a lattice reduction matrix (IT 1989, Volume A, p746), set of equivalent indexing transformations
- A distortion index - generic penalty (IT 2006, Volume F, p226)

2.2 XDS Plugin

Gleb suggested the following workflow for the XDS indexing plugin:

- 1. Spot Search
- 2. Indexing N solutions with penalty <
  - a) known symmetry
  - b) unknown symmetry
- 3. Refine N solutions
- 4. Choosing a solution (like MOSFLM?)

2.3 Persistence

The current EDNA framework data binding (based on “generateDS” version 1.11b) allows data to be saved as XML into a file, and later be retrieved. The current version of the prototype “persists” its meta-data in XML files in the working directory hierarchy.

Sandor argued in favor of a “dynamic workflow” where the EDNA system could freely navigate this directory hierarchy in order to obtain results. This would correspond to loosely coupled plugins. This dynamic workflow is not envisaged in the near future but could be interesting for later developments.

2.4 Plugins

We discussed briefly the two main types of EDNA plugins: execution and control. We agreed that documentation of the plugin hierarchy is both urgent and essential.

2.5 Prototype improvements

During the discussions we came up with this list of suggestions for improving the EDNA framework and the prototype. The idea is to convert each point to a bug in the EDNA bugzilla repository:

- In EDPlugin : loadPlugin method with error handling, in the current version of the prototype each plugin implements the same error handling for checking if a plugin has been correctly loaded.
- How to improve translation to and from generic data model : in the current version the translations are made by “handlers”. The MOSFLM handler could for example be split into several handlers, one for indexing, one for integration etc.
- Organisation of the test cases : the current organisation is very flat, all the test cases are in the same directory. A nicer organisation would be to move the plugin specific test cases to their corresponding plugin directories.
- Remove empty calls to super methods
- Configuration : possibility to name plugins which would allow different configurations for plugins with the same class.
- Make “abstract” method in EDPlugin (or higher). This is due to a limitation of Python which doesn't allow a method or a class do be declared abstract.
- Add “equal” method to data binding (with tolerance). Sandor also suggested to be able to control the tolerance from Enterprise Architect.
- Sooner or later we will need some kind of workflow tool for EDNA. For the moment, at least an XML editor would be very useful.
- New plugin class : “wrappers”. The idea is to “wrap” each specific execution plugin in a “wrapper” which would take care of the necessary translations of the data model. The wrapper plugins would then be used in the higher-level control plugins. For example, MOSFLM indexing plugin and XDS plugin would use the same wrapper.

3 Development of the XDS Spot Search plugin

This part of the workshop resulted in the following “How to write an EDNA plugin” tutorial that will be published on the EDNA Wiki.

3.1 General guidelines for writing an EDNA plugin

1. Define the input and result data models
   - o Create XSDataInput... and XSDataResult... classes.
2. Write the test for the plugin
3. Define the type of plugin (execution, control etc) :
   - o Derive from EDPluginExec, EDPluginExecProcess, EDPluginExecProcessScript or EDPluginControl.
4. Define and/or override methods:
   - o setInputData : must be overridden! for casting XML into the XSDataInput type.
   - o checkParameters : called befor config for assuring complete input data
   - o config : called before preProcess, used for configuring the plugin
   - o preProcess : called before process, used for setting up files etc. needed by the plugin
   - o process : The main method. Does not need to be defined if derived from EDPluginExecProcess or EDPluginExecProcessScript
   - o postProcess : called after the process, used for recuperating the data produced by e.g. external programs
   - o generateExecutiveSummary : should write a human readable result summary after the execution of the plugin

3.2 Tutorial : How to write an EDNA plugin which uses XDS for spot search

3.2.1 Write use case

This plugin finds spots on a single sub wedge using XDS.

3.2.2 Create Plugin Directories

1. In $EDNA_HOME/prototype/plugins create a new directory “EDPluginXDSSpotSearch-v0.1”
2. In this directory, create “datamodel” and “plugins” directories.

3.2.3 How to create a new EDNA specific data model

1. Open Enterprise Architect
2. File → New Project → Save project as $EDNA_HOME/prototype/plugins/ EDPluginXDSSpotSearch-v0.1/datamodel/XSDataXDSSpotSearchv01.EAP

3. Don't choose any option, just click on OK.
4. Open the “Project Browser”
5. Rename “Model” to “XSDataXDSSpotSearchv01”
6. Rightclick on “XSDataXDSSpotSearchv01” icon, choose “Import data model from XMI”
7. Choose to import “$EDNA_HOME/prototype/datamode/XSDataCommonv01.xmi”
8. Answer “Yes” to “Ok to import XMI file?” question
9. Click on “Close” to close the import dialog window
10. Rightclick again on the top “XSDataXDSSpotSearchv01” icon in the project browser and choose “New view...”
11. Type the name “XSDataXDSSpotSearchv01” and choose “Class view”, click on OK
12. Rightclick on the new icon “XSDataXDSSpotSearch” and choose “add → Add diagram...”
13. Choose “UML Structural” and “Class”, keep the name and click on OK
14. You should now see “Logical diagram 'XSDataXDSSpotSearch'” in the main window, if not double click on the new “XSDataSpotSearch” diagram icon
15. Select by clicking once on the “XSDataXDSSpotSearchv01” model icon (not top level nor the diagram icon, but the one in the middle)
16. Choose “View → Tagged values”
17. Click on the new tag icon
18. Add a new tag with “Tag: schemaLocation = XSDataCommonv01.xsd” and “targetNamespace = http://www.examlpe.org/edna”
19. Start to construct the data model...
    1. How to derive from XSDataInput etc.
    2. Use associations and notes for clarity
20. Save data model
21. Generate XSD file : Project → XML Schema → Generate XML Schema. Choose name “$EDNA_HOME/prototype/plugins/EDPluginXDSSpotSearch-v0.1/datamodel/XSDataXDSSpotSearchv01.exsd” - **Make sure the correct source package is selected in the project browser**.
22. Export xmi file : Project → Import/Export → Export package to XMI. Choose file name “$EDNA_HOME/prototype/plugins/EDPluginXDSSpotSearch-v0.1/datamodel/XSDataXDSSpotSearchv01.xmi”
23. Save the view as image : Diagram → Save as image..., save the image as “$EDNA_HOME/prototype/plugins/EDPluginXDSSpotSearch-v0.1/datamodel/XSDataXDSSpotSearchv01.png”.

3.2.4 Write the test for the plugin

1. Create input test data in $EDNA_HOME/prototype/tests/data/EDPluginXDSSpotSearch:
2. Write a execute test case “EDTestCasePluginXDSSpotSearchv01”

3. Add this test to EDTestSuitePluginExecute

3.2.5 Write configuration for the plugin

1. Define essential configuration information like path to executable etc.

3.2.6 Write the plugin

1. To be visualised in the wiki...