EDNA ½ Day Introduction

• Agenda:

14:00 - 15:15 (Olof)
- Why EDNA?
- Collaboration
- Data model framework
- Modularity / plugins
- Workflows
- Testing framework

15:15 – 15:45 - Pause

15:45 - 17:00
- Azimuthal integration and diffraction tomography using EDNA (Jérôme)
- Workflow tools for EDNA (Matthew)
- MX applications (Olof, if time allows)
Why EDNA?

• EDNA is the best answer we (developers) have come up with so far for answering these questions:

  • How can we “pipeline” existing scientific software programs/packages for (online) data analysis workflows?
  • How can we make workflows that is easily adapted to new versions of scientific software packages?
  • How can we abstract certain calculations to be “generic”, e.g. indexing of a diffraction pattern?
  • How can we make “flexible” workflows, i.e. workflows that can be changed rapidly depending on the scientific needs?
  • How can we automate data analysis workflows?
  • How can we make these workflows robust?
  • How can we collaborate efficiently?
  • How can we re-use code developed by another facility without breaking existing functionality?
So – what is then EDNA?

• Short answer:
  
  A tool for Online Data Analysis
  
  and
  
  A Collaborative Effort

• Long answer :
  
  This introduction!
Online Data Analysis

• Data analysis performed as quickly as possible in order to provide feedback to the user(s):
  • Quick calculations for data visualisation
  • Quality of the data generated by the experiment – is the experiment successful?
  • Use of results for planning the experiment: e.g. MX strategy calculation

• Combination of data acquisition and data processing

• Automation - Workflow

• Meta data definitions important (data manager)

• Impact on beamline efficiency:
  • Should introduce added value
  • No slow-down of experiments
  • No or little impact on beamline scientists / users workload
Challenge for the Upgrade: Massively Automated Sample Selection Integrated Facility

- Baskets from Lab DS loaded on screening carrousel
- Screening station(s)
- From screening to sorting
- Samples taken from Lab baskets & put back after screening
- Sorting/Cleaning station
- From sorting to cleaning
- Manual transfer to Data collection station(s)
- Data collection station(s)
- From sorting to Data collection
- From Data collection to sorting/cleaning
The EDNA foundation – the project management
EDNA Project Management
To be revised on September 23rd 2010

• Executive Committee:
  • Alun Ashton, DLS, UK
  • Gérard Bricogne, Global Phasing, UK
  • Andrew Leslie, MRC LMB, Cambridge, UK
  • Andrew McCarthy, EMBL-Grenoble, France
  • Sean McSweeney, ESRF, Grenoble, France
  • Thomas Schneider, EMBL-Hamburg, Germany
  • Andrew Thompson, Synchrotron Soleil, France

• Other members from:
  • BESSY, Berlin, Germany
  • MAX LAB, Lund, Sweden
  • NSLS,Brookhaven, U.S.
  • SLS, Villigen, Switzerland
  • University of Sydney, Australia
  • University of York, UK
EDNA Project Management

- Project agreement
- Coding conventions
- Code reviews
- Development tools
  - Eclipse
  - Enterprise architect
- Project portal
  - http://www.edna-site.org
  - Wiki documentation
  - Bugzilla server
  - Subversion server
  - Discussion forum
- Executive committee
- Video conferences
- Developers’ meetings & workshops

Marratech video-conferencing tool (now replaced by EVO)
The first pillar – Data Model Framework

Data Model / UML → Code

Project Management
EDNA Data Model Framework

• What is a data model? From wikipedia:

A data model in software engineering is an abstract model that describes how data are represented and accessed. Data models formally define data elements and relationships among data elements for a domain of interest.

Communication and precision are the two key benefits that make a data model important to applications that use and exchange data.

• Since we want to make workflows → communication between programs → data modelling is important
How are Data Models used in EDNA?

• The “common” data model:
  • This data model defines a set of simple basic types (e.g. double, string etc) and some more complex (3x3 matrix) which can be used by all other EDNA data models.
  • The common data model is a part of the EDNA kernel.

• The “specific” data models:
  • Data models which are specific for a certain task or program, e.g. data models for MOSFLM, XDS, FIT2D etc.
  • The specific data models are typically used only by a few EDNA plugins (modules).

• The “generic” or “project” data models:
  • These data models should not be dependent on a single program but rather be developed for a certain scientific area, e.g. MX, tomography etc.
The EDNA Data Model Framework

- From UML diagrams to generated code (data binding):

```xml
<xs:element name="XSDataSample" type="XSDataSample"/>
<xs:complexType name="XSDataSample">
  <xs:complexContent>
    <xs:extension base="XSData">
      <xs:sequence>
        <xs:element name="absorbedDose" type="XSDataAbsorbedDose" minOccurs="0" maxOccurs="1"/>
        <xs:element name="shape" type="XSDataFloat" minOccurs="0" maxOccurs="1"/>
        <xs:element name="size" type="XSDataSize" minOccurs="0" maxOccurs="1"/>
        <xs:element name="susceptibility" type="XSDataFloat" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
The second pillar – modularity / plugins
Why do we want modules / plugins?

• Again from wikipedia:

In computing, a plug-in is a set of software components that adds specific capabilities to a larger software application.

Applications support plug-ins for many reasons. Some of the main reasons include:

• to enable third-party developers to create capabilities which extend an application
• to support easily adding new features
• to reduce the size of an application
• to separate source code from an application because of incompatible software licenses.
EDNA Framework: Kernel + Plugins

• The EDNA kernel contains:
  • The common data model and data binding generator code
  • Base classes for all EDNA plugins
  • Base classes for EDNA applications
  • Some utility/helper classes
  • The testing framework
  • The plugin generator
  • Plugin and test launcher scripts
  • The EDNA kernel is written in pure Python
  • No dependency on AALib any longer

• An EDNA application consists generally of:
  • One or several data model based on the common data model
  • A set of plugins derived from the kernel plugin base classes
  • One or several application classes
  • One or several scripts for launching the application
EDNA Modularity: Plugins and their hierarchy

- **Plugin base class:**
  - Configuration, working directory, etc.

- **Execution plugins:**
  - Execution of external programs, e.g. (bash) scripts

- **Controller plugins:**
  - Control of execution plugins
  - Parallel execution
  - Synchronisation
EDNA Plugins Features:

- Self-contained plugin structure:
  - Data model(s)
  - Plugin source code
  - Data binding objects
  - Unit and execution tests
  - Data for tests
  - Documentation

- Fast dynamic plugin loading (cache)
- Plugin execution and synchronisation (threadsafef)
- Plugin configuration
- Handling of input and output data
The third pillar - workflows

- Modular / Plugins
- Data Model / UML → Code
- Workflow model

Project Management
Example EDNA workflow: MXv1 Characterisation (1)

- MX sample characterisation taking into account radiation damage
- Indexing using MOSFLM or Labelit
- Parallel integration of reference images
- If flux + beamsize:
  - RADDOSE for estimating radiation damage
- BEST strategy calculation
  - taking into account radiation damage
  - multi-subwedge data collection strategies
Example Characterisation Workflow (1)

- Characterisation XML Input
  - Plugin for preparing indexing input
    - Indexing plugin
      - Plugin for preparing integration input
        - Integration plugin
          - Plugin for preparing strategy input
            - Strategy plugin
              - Plugin for assembling characterisation results
                - Characterisation XML results
MXv1 Characterisation (2)

- MOSFLM indexing
  - Indexing Evaluation → Ok
    - Labelit indexing
      - Indexing Evaluation → Ok
        - Failure
      - Indexing Evaluation → Failure
  - MOSFLM Predictions
  - MOSFLM integration
    - [Raddose]
      - Best
Why use a workflow tool in EDNA?

- Implicit documentation of workflow
- Implicit parallel workflows
- Possibility to “easily” modify / construct new workflows
- Possibility to debug workflows
- Possibility to restart a stopped workflow
The fourth pillar – the testing framework
EDNA Testing Framework

- The EDNA testing framework consist of three layers:
  - Kernel Unit tests
  - Plugin Unit tests
  - Plugin Execution tests

- Example of EDNA Plugin Execution tests result:

```
[UnitTest]: ###################################################################
[UnitTest]: Result for EDTestSuiteKernel : SUCCES
[UnitTest]:
[UnitTest]:
[UnitTest]: Total number of test cases executed with SUCCESS : 10
[UnitTest]: Total number of test cases executed with FAILURE : 0
[UnitTest]:
[UnitTest]: Total number of test methods executed with SUCCESS : 26
[UnitTest]: Total number of test methods executed with FAILURE : 0
[UnitTest]:
[UnitTest]: Runtime : 4.420 [s]
[UnitTest]: ###################################################################
```
To be avoided...
EDNA Documentation

• Available today:
  • Data models (png)
  • Automatic API doc generation
  • Wikipages with developers' “How-to”s
  • Minutes / presentations of previous meetings, code camps etc

• Planned:
  • Automatic plugin documentation repository (use cases etc)
  • Workflow documentation (workflow tool)
EDNA is not DNA...

• Same goals (initially)...
  • Automatic MX sample characterisation
  • Online data processing during MX data collection
  • Ranking

• ...however very different implementations
  • EDNA designed to not be specific to MX
  • No shared code base between DNA and EDNA
  • Different project management
  • Different collaborators
<table>
<thead>
<tr>
<th>EDNA Collaborators 2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander Popov(e)</td>
</tr>
<tr>
<td>Alun Ashton(b)</td>
</tr>
<tr>
<td>Andrew Leslie(h)</td>
</tr>
<tr>
<td>Andrew McCarthy(c)</td>
</tr>
<tr>
<td>Andrew Thompson(k)</td>
</tr>
<tr>
<td>Clemens Schulze(j)</td>
</tr>
<tr>
<td>Clemens Vorrin(f)</td>
</tr>
<tr>
<td>Darren Spruce(e)</td>
</tr>
<tr>
<td>Elspeth Gordon(e)</td>
</tr>
<tr>
<td>Ezequiel Panepucci(j)</td>
</tr>
<tr>
<td>Gérard Bricogne(f)</td>
</tr>
<tr>
<td>Gerrit Langer(c)</td>
</tr>
<tr>
<td>Gleb Bourenkov(c)</td>
</tr>
<tr>
<td>Gordon Leonard(e)</td>
</tr>
<tr>
<td>Harry Powell(h)</td>
</tr>
<tr>
<td>Jérôme Kieffer(e)</td>
</tr>
<tr>
<td>Johan Turkenburg(m)</td>
</tr>
<tr>
<td>Johan Unge(g)</td>
</tr>
<tr>
<td>John Skinner(i)</td>
</tr>
<tr>
<td>Karl Levik(b)</td>
</tr>
<tr>
<td>Katherine McAuley(b)</td>
</tr>
<tr>
<td>Lucile Roussier(k)</td>
</tr>
<tr>
<td>Marie-Françoise Incardona(e)</td>
</tr>
<tr>
<td>Mark Basham(b)</td>
</tr>
<tr>
<td>Meitian Wang(j)</td>
</tr>
<tr>
<td>Michael Hellmig(a)</td>
</tr>
<tr>
<td>Olga Roudenko(k)</td>
</tr>
<tr>
<td>Peter Keller(f)</td>
</tr>
<tr>
<td>Peter Turner(l)</td>
</tr>
<tr>
<td>Pierre Legrand(k)</td>
</tr>
<tr>
<td>Robert Sweet(i)</td>
</tr>
<tr>
<td>Romeu Pieritz(e)</td>
</tr>
<tr>
<td>Sandor Brockhauser(c)</td>
</tr>
<tr>
<td>Sean McSweeney(e)</td>
</tr>
<tr>
<td>Takashi Tomizaki(j)</td>
</tr>
<tr>
<td>Thomas Schneider(c)</td>
</tr>
<tr>
<td>Uwe Mueller(a)</td>
</tr>
<tr>
<td>(a) BESSY, Berlin, Germany</td>
</tr>
<tr>
<td>(b) Diamond Light Source, UK</td>
</tr>
<tr>
<td>(c) EMBL, Grenoble, France</td>
</tr>
<tr>
<td>(d) EMBL, Hamburg, Germany</td>
</tr>
<tr>
<td>(e) ESRF, Grenoble, France</td>
</tr>
<tr>
<td>(f) Global Phasing, Cambridge, UK</td>
</tr>
<tr>
<td>(g) MAX LAB, Lund, Sweden</td>
</tr>
<tr>
<td>(h) MRC LMB, Cambridge, UK</td>
</tr>
<tr>
<td>(i) NSLS, Brookhaven, U.S.</td>
</tr>
<tr>
<td>(j) SLS, Villigen, Switzerland</td>
</tr>
<tr>
<td>(k) Synchrotron Soleil, France</td>
</tr>
<tr>
<td>(l) University of Sydney, Australia</td>
</tr>
<tr>
<td>(m) University of York, UK</td>
</tr>
</tbody>
</table>

EDNA developers
Executive committee