EDNA Training

- Agenda Monday November 15th:

  09:30 – 10:00 – Introduction to EDNA

  10:00 – 10:30 – Installation of Eclipse (TopCased, UML2XSD)

  10:30 – 10:45 – Pause

  10:45 – 11:15 – The photov1 project, conceptual design, data model

  11:15 – 12:00 – Eclipse pydev, plugin generation, tests

  12:00 – 13:30 – Lunch

  13:30 – 15:00 – The photov1 project: Execution plugins

  15:00 – 15:15 – Pause

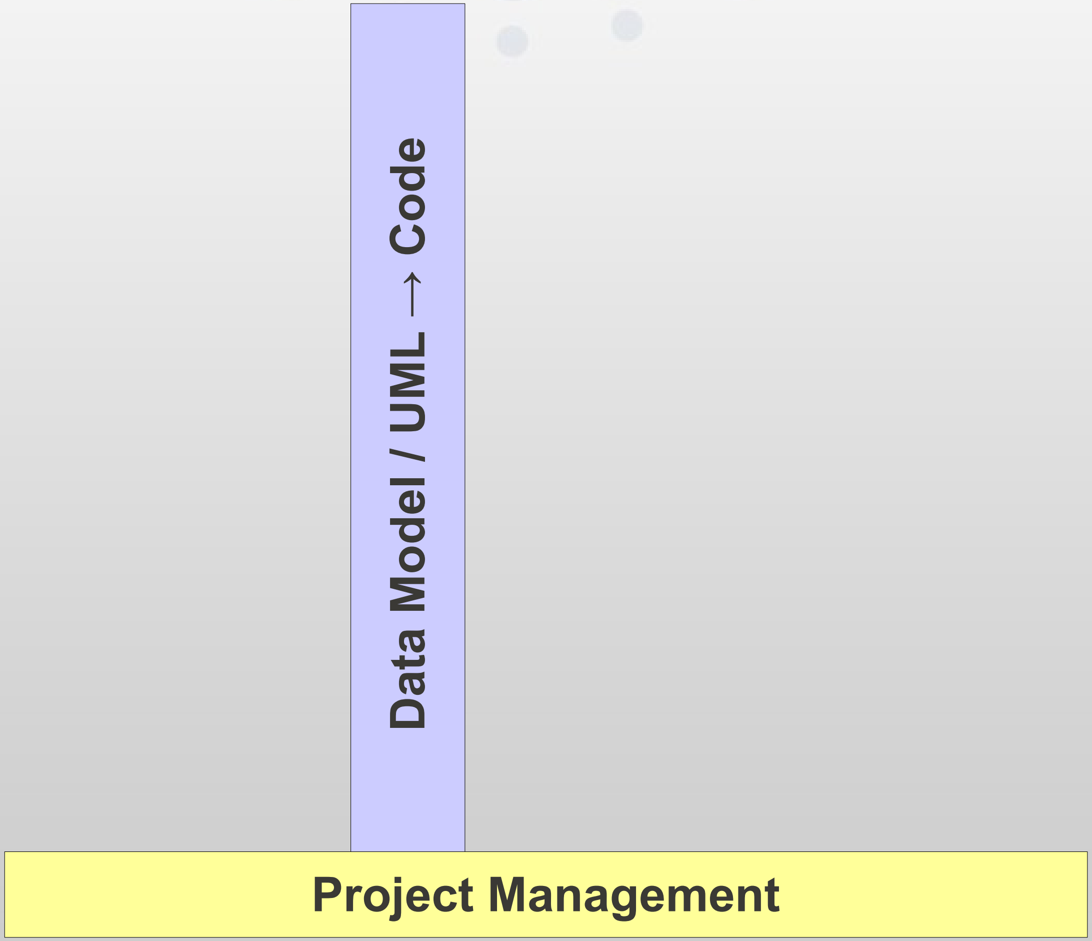
  15:15 – 17:00 – The photov1 project: Control plugins

- Tuesday November 16th : To be defined during Monday

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?

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.

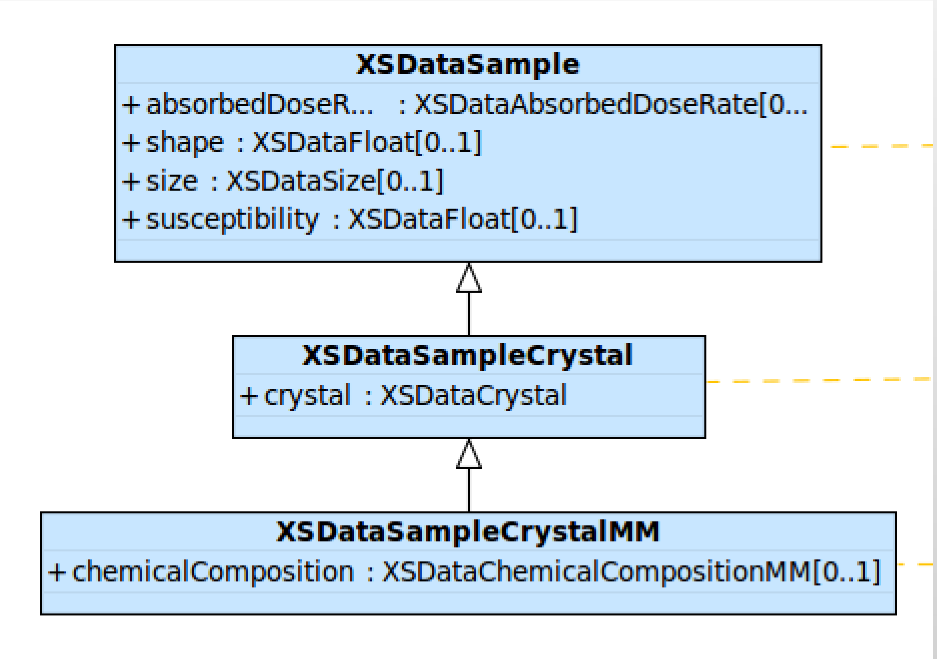
- Since we want to make workflows → communication between programs → data modelling is important

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 typacilly 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) :



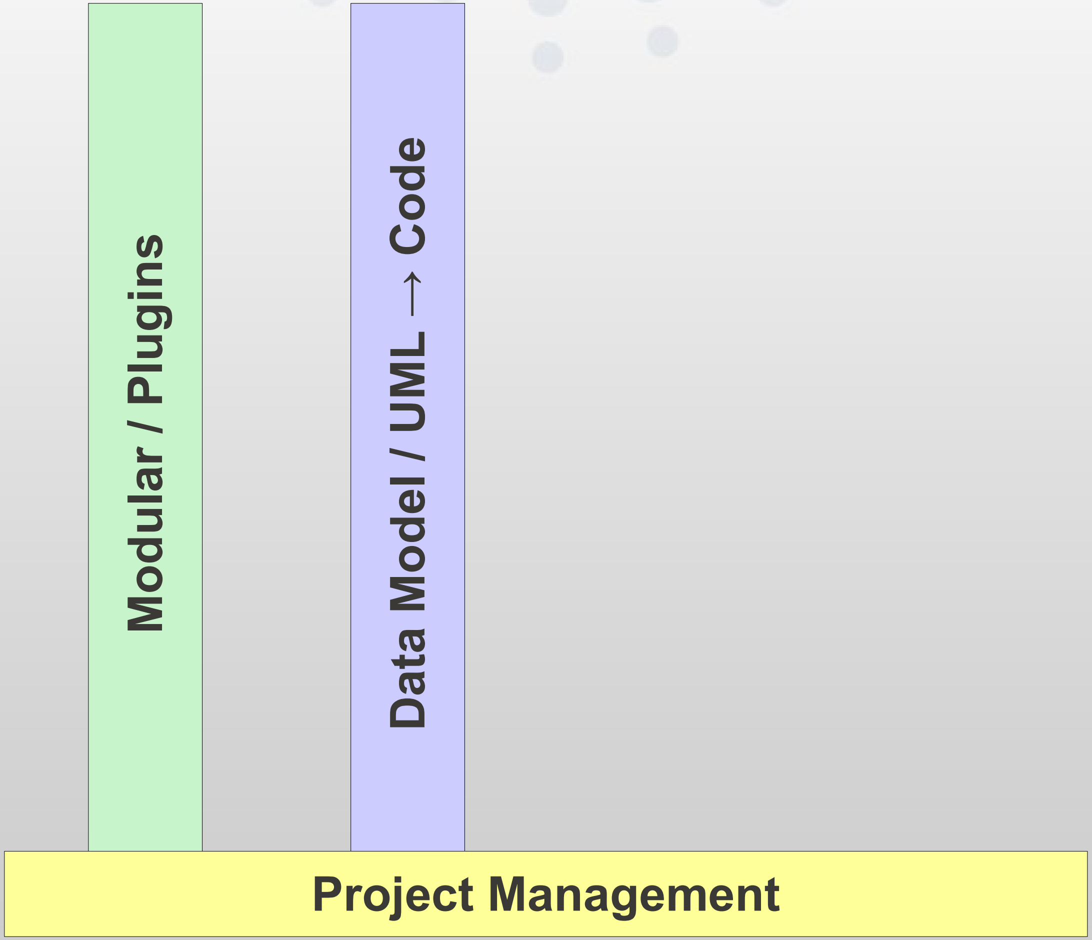
XSDataSample
+ absorbedDoseR... : XSDataAbsorbedDoseRate[0...
+ shape : XSDataFloat[0..1]
+ size : XSDataSize[0..1]
+ susceptibility : XSDataFloat[0..1]

XSDataSampleCrystal
+ crystal : XSDataCrystal

XSDataSampleCrystalMM
+ chemicalComposition : XSDataChemicalCompositionMM[0..1]

```
<xs:element name="XSDataSample" type="XSDataSample
<xs:complexType name="XSDataSample">
     <xs:complexContent>
          <xs:extension base="XSData">
               <xs:sequence>
                    <xs:element name="absorbedDose
                    <xs:element name="shape" type=
                    <xs:element name="size" type="
                    <xs:element name="susceptibil
               </xs:sequence>
          </xs:extension>
     </xs:complexContent>
</xs:complexType>
```

The second pillar – modularity / plugins



Modular / Plugins

Data Model / UML → Code

Project Management

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:

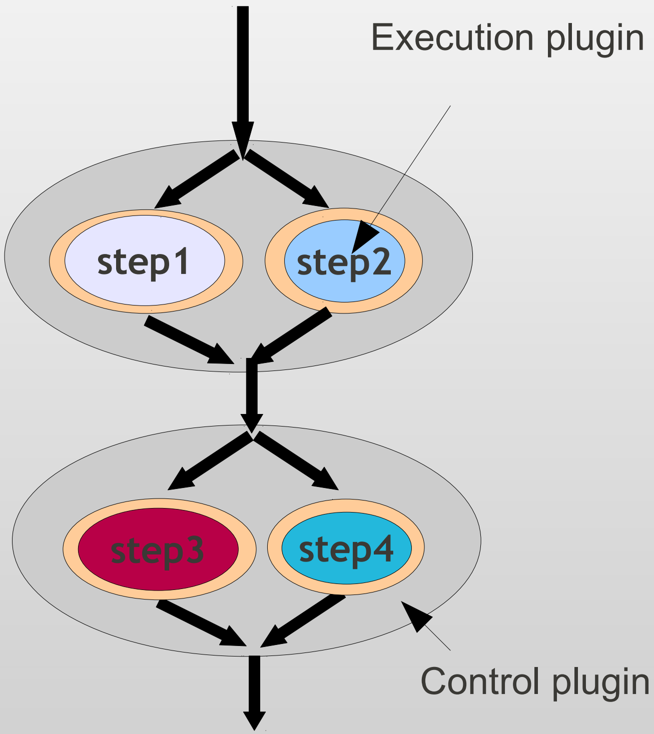
  - *to enable third-party developers to create capabilities which extend an application*
  - *to support easily adding new features*
  - *to reduce the size of an application*
  - *to separate source code from an application because of incompatible software licenses.*

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

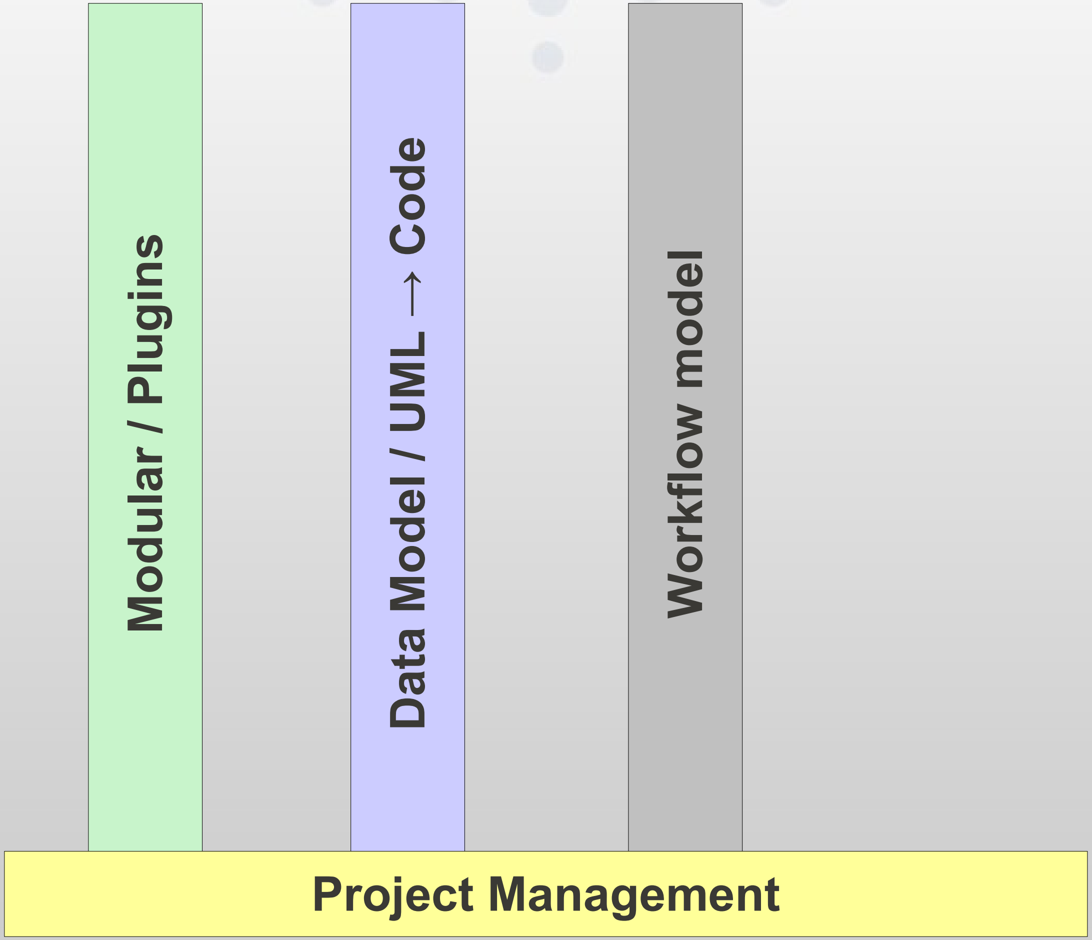
- Plugin base class :
  - Configuration, working directory, etc.
- Execution plugins :
  - Execution of external programs, e.g. (bash) scripts
- Controller plugins:
  - Control of execution plugins
  - Parallel execution
  - Synchronisation



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 (threadsafe)
- Plugin configuration
- Handling of input and output data

The third pillar - workflows



Modular / Plugins

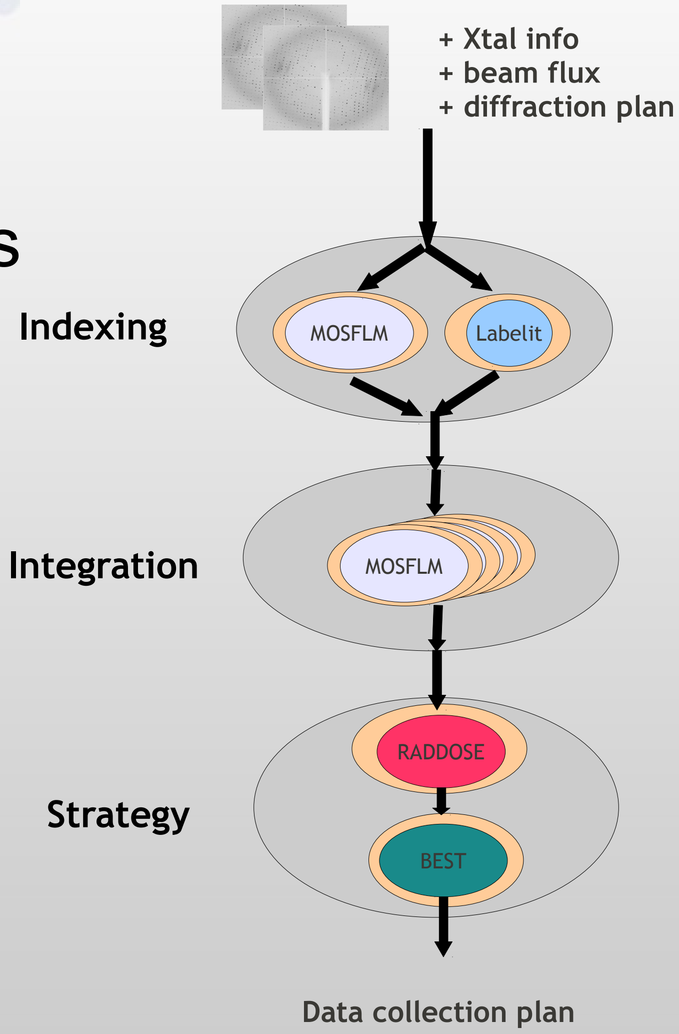
Data Model / UML → Code

Workflow model

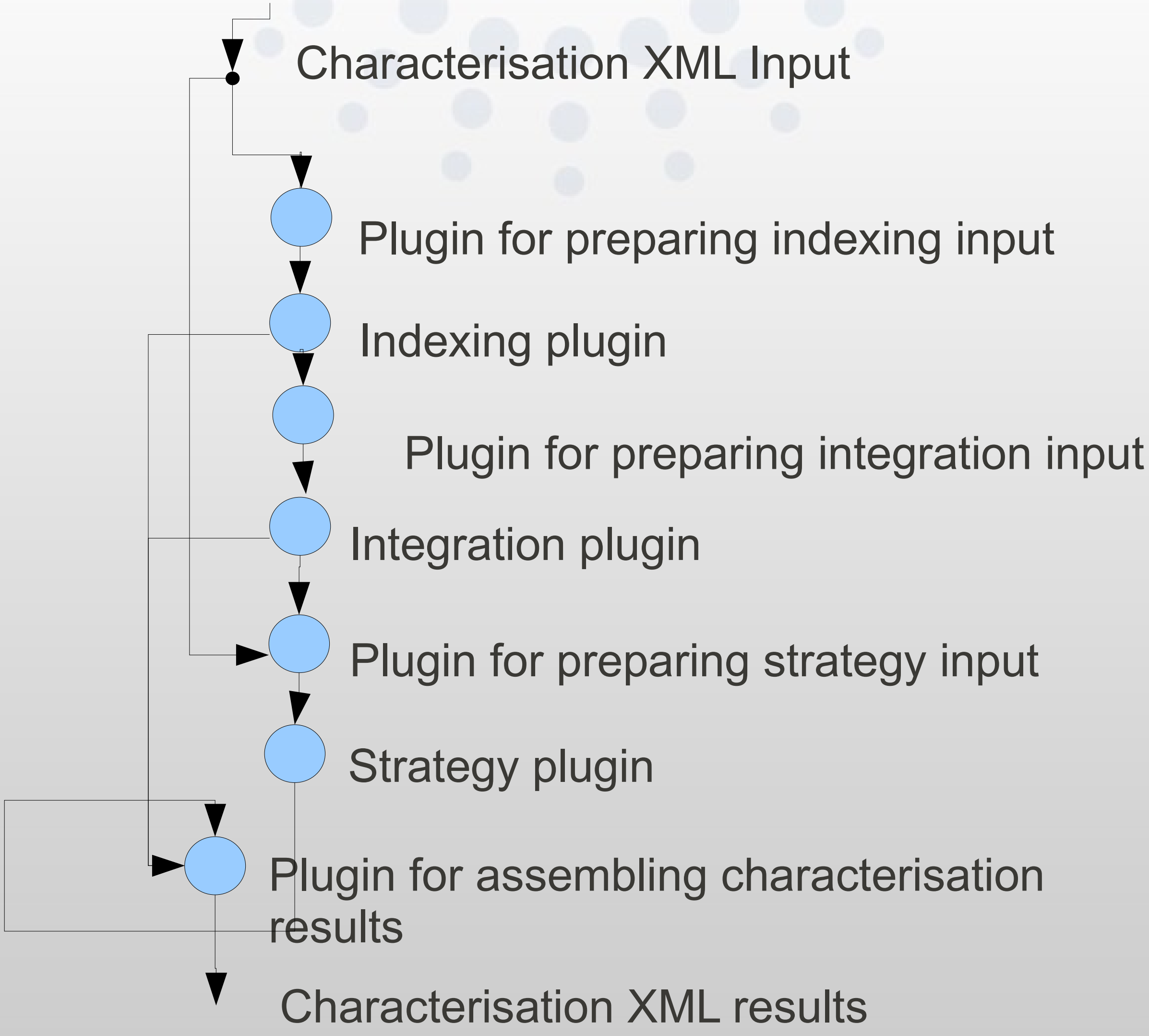
Project Management

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

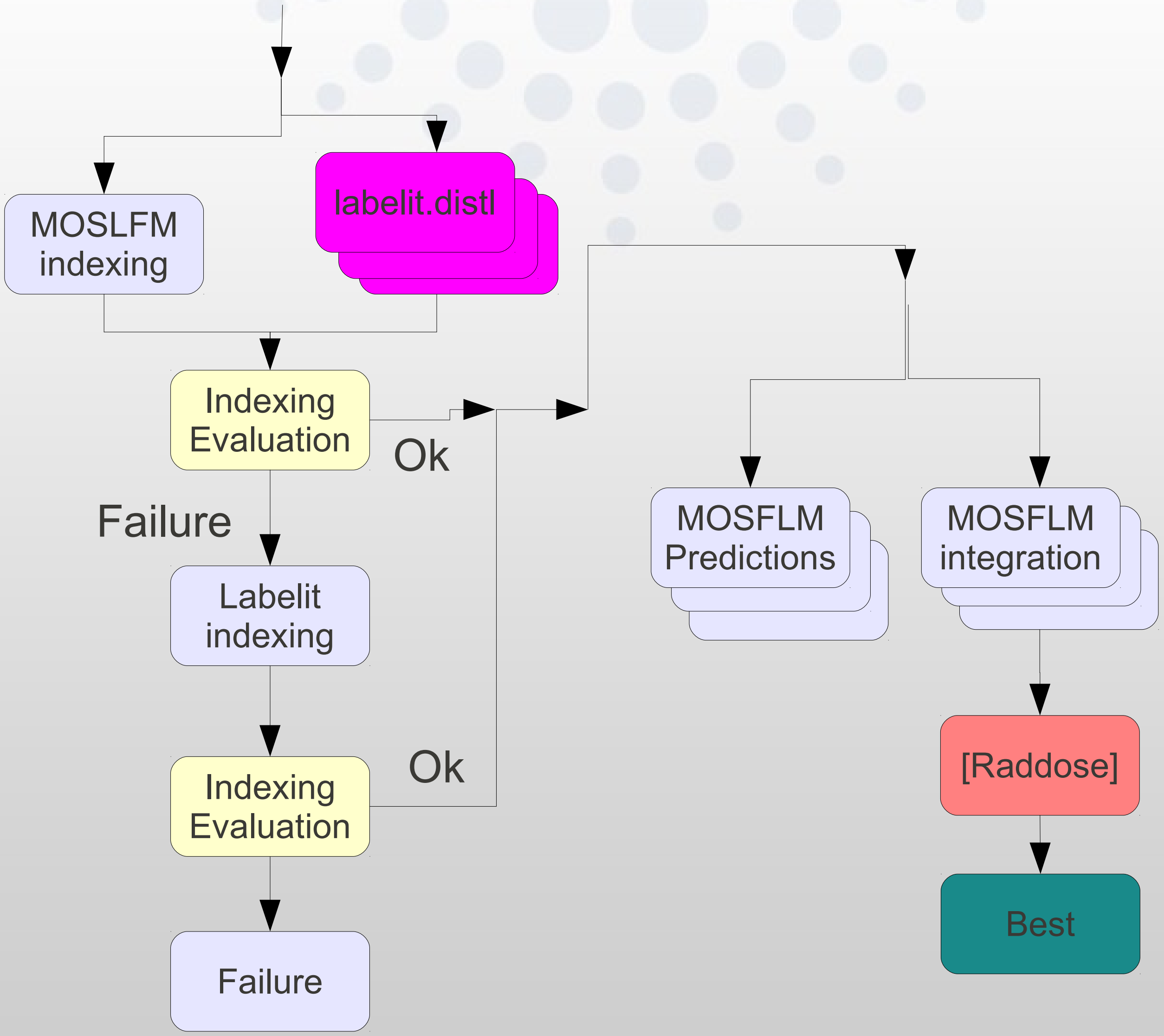
Plugin for preparing strategy input

Strategy plugin

Plugin for assembling characterisation results

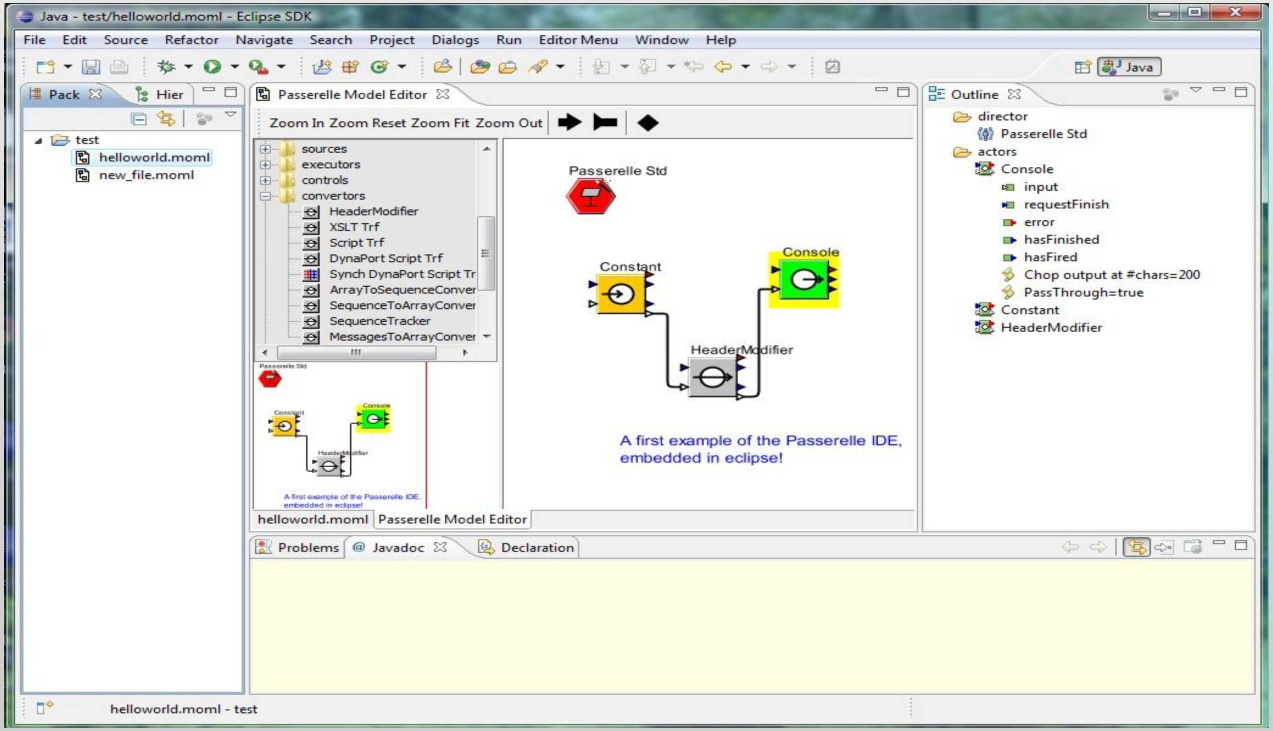
Characterisation XML results

MXv1 Characterisation (2)



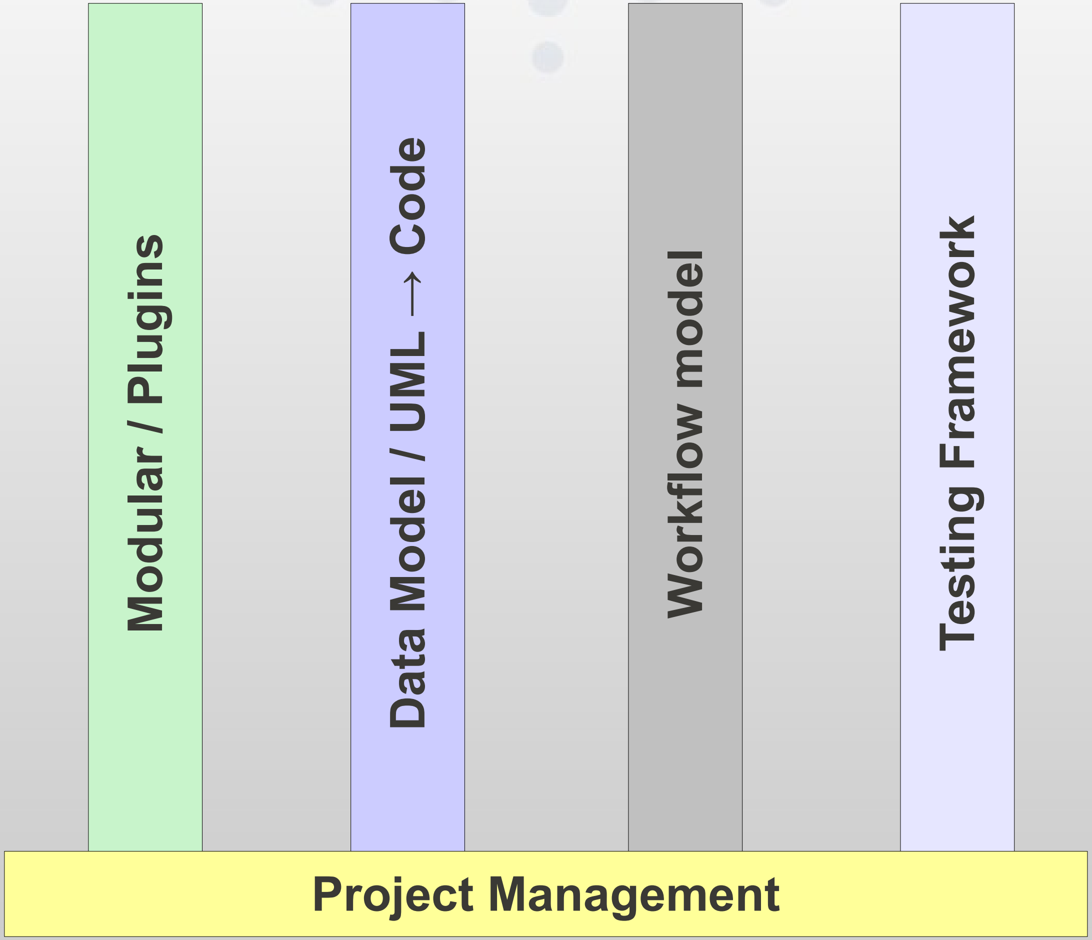
Why use a workflow tool in EDNA?

- Implicit documentation of workflow
- Implicit parallel workflows
- Possibility to “easily” modify / construct new workflows
- Possibility to debug workflows
- Possibility to restart a stopped workflow



A first example of the Passerelle IDE, embedded in eclipse!

The fourth pillar – the testing framework



Modular / Plugins

Data Model / UML → Code

Workflow model

Testing Framework

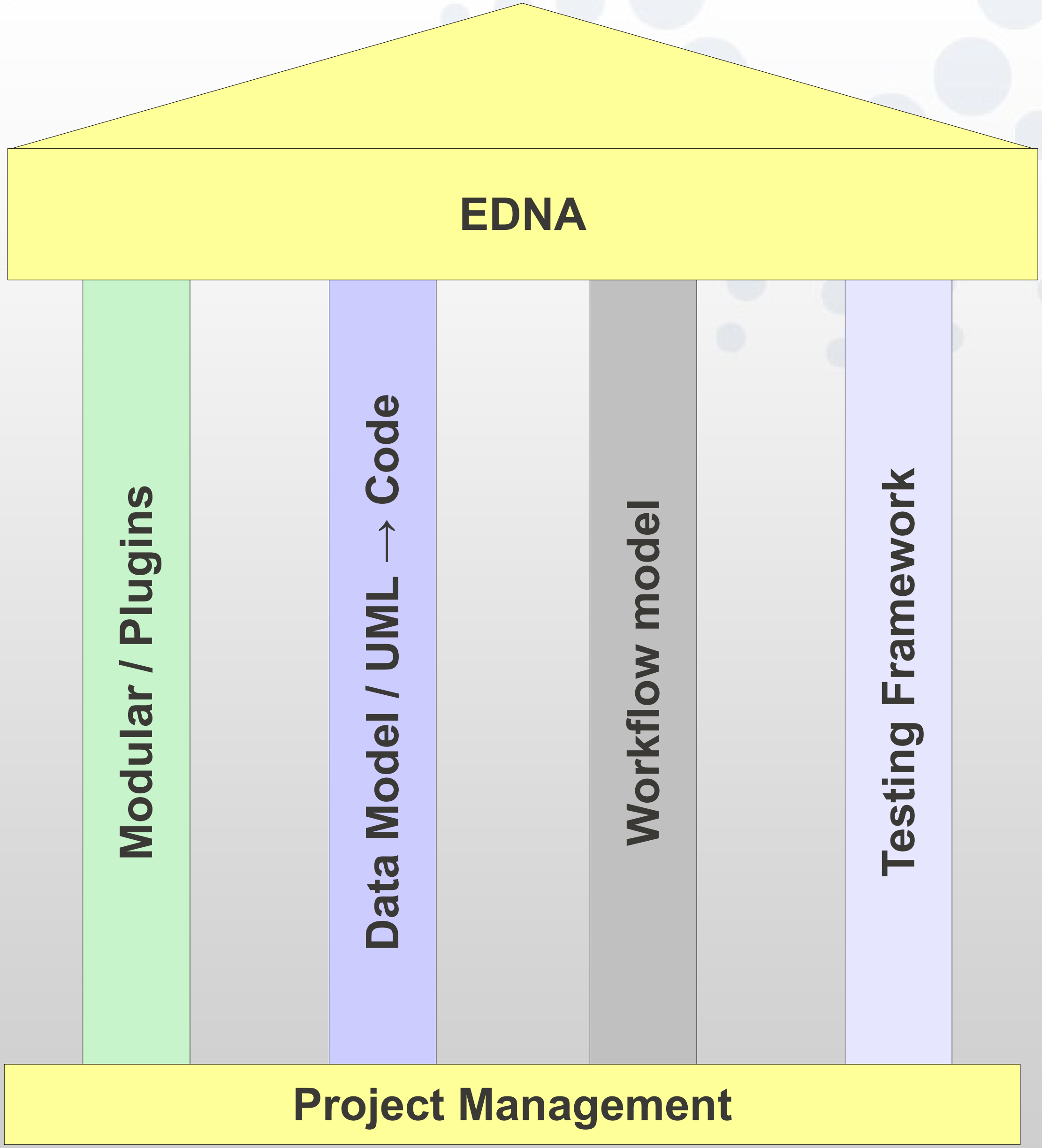
Project Management

EDNA Testing Framework

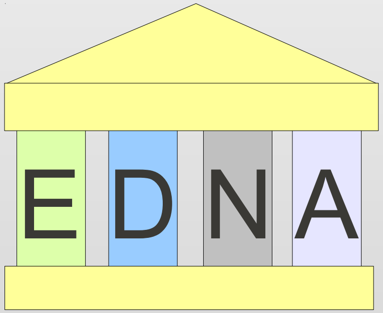
- The EDNA testing framework consist of three layers :
  - Kernel Unit tests
  - Plugin Unit tests
  - Plugin Execution tests

- 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]: ###################################################################
```

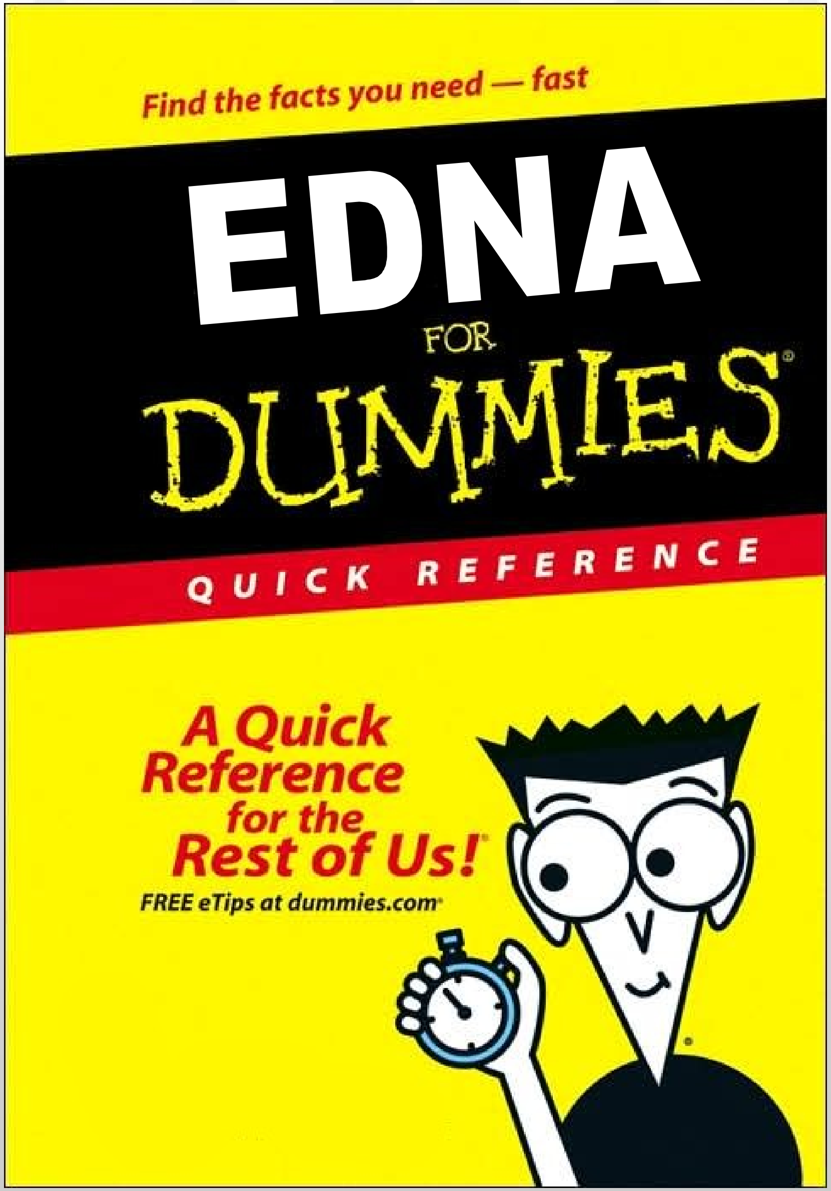

EDNA
Modular / Plugins
Data Model / UML → Code
Workflow model
Testing Framework
Project Management

New logo :


E D N A

To be avoided...

Documentation!


Find the facts you need — fast
EDNA
FOR
DUMMIES®
QUICK REFERENCE
A Quick Reference for the Rest of Us!®
FREE eTips at dummies.com®

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)