EDNA Tutorial

Layout

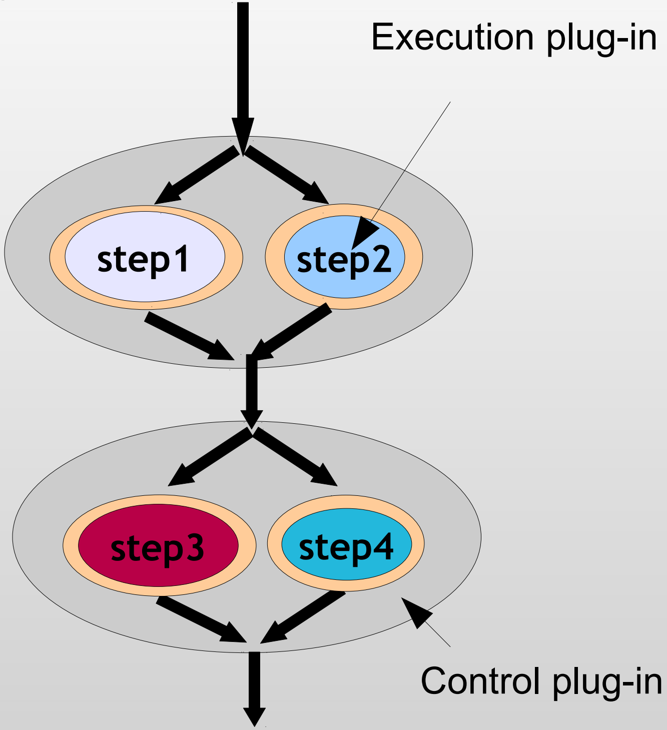
- Short presentation of EDNA
- How to run an EDNA plugin / workflow / application
  - Execution plugins :
    - How to write a wrapper for an external program
    - How to write a pure python plugin
  - Workflow (control) plugins:
- How to implement a new workflow (e.g. Online Data Analysis)
- How to modify an existing workflow

Introduction to EDNA

- See Olof's presentation.

Strength of EDNA

- EDNA is a robust pipe-lining tool for on-line data analysis
  - It has been tested with thousands of tasks at once
- EDNA allows hi-performances
  - Multi-threaded implementation
- EDNA relies on data-models
  - Visual communication with scientists
  - Automatic bindings with the code
- EDNA has a strong testing framework
  - Unit & Execution tests
  - Non regression test before nightly builds
- EDNA is efficient to program
  - Plugin generator for execution plug-ins based on the data-model
  - Re-use of plug-ins already written by others: EDNA-Toolbox
- EDNA is an international collaboration (with DLS, CCP4, ...)



Weaknesses of EDNA

- Technical weaknesses:
  - Learning curve is (too ?) steep, hard to set-up
    - After one year, most of it is justified
  - Multi-threading limited by the GIL in C-Python
    - Issue only in pure python plugins
    - No problem if the GIL is released (Numpy, external programs …)
  - Datamodeling tool is not free (Enterprise Architect)
    - Move to Eclipe-EMF soon
- Collaboration weaknesses:
  - Few scientists know EDNA, mainly IT staff.
  - Too much MX-related for most people not involved in the project

Overview of the EDNA Framework

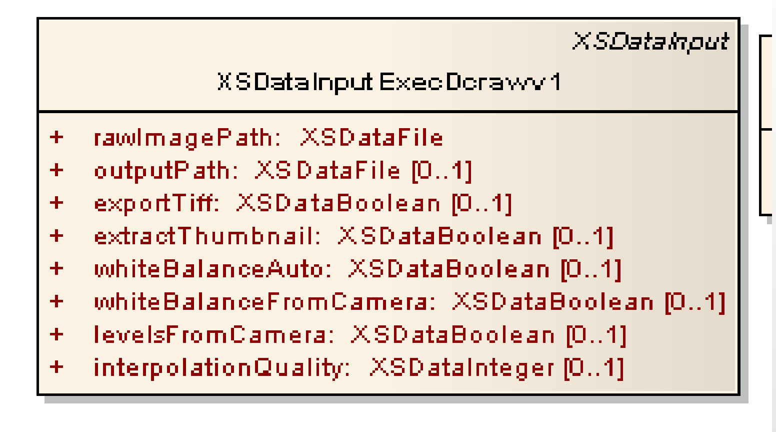
Data modelling tools

- Each plugin has its input and output defined in UML
- UML editing is (still) done in Enterprise Architect
- I/O classes are named XSDataInput / XSDataOutput
    - Exported as PNG
    - Exported as XSD
    - Exported as XMI (for the migration to EMF and exchange with other tools ...)

- XSD-files are automatically converted into python code
    - Never edit those modules
    - Don't even look at them (the code is ugly)

Example of datamodel

- UML datamodel

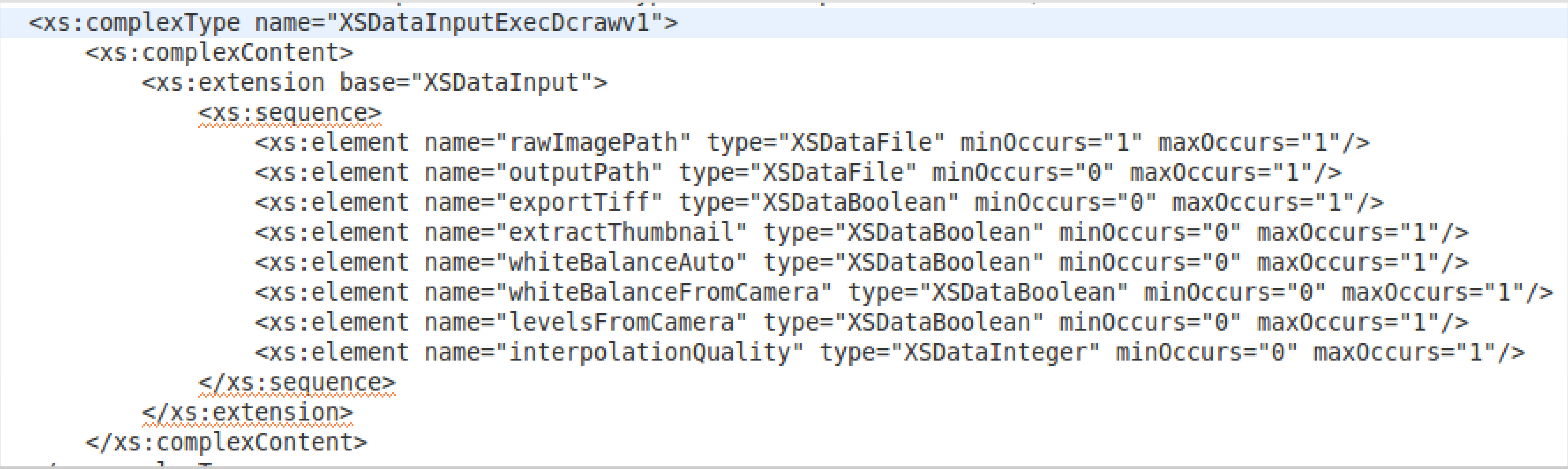


XSDataInput

XSDataInputExecDcrawv1

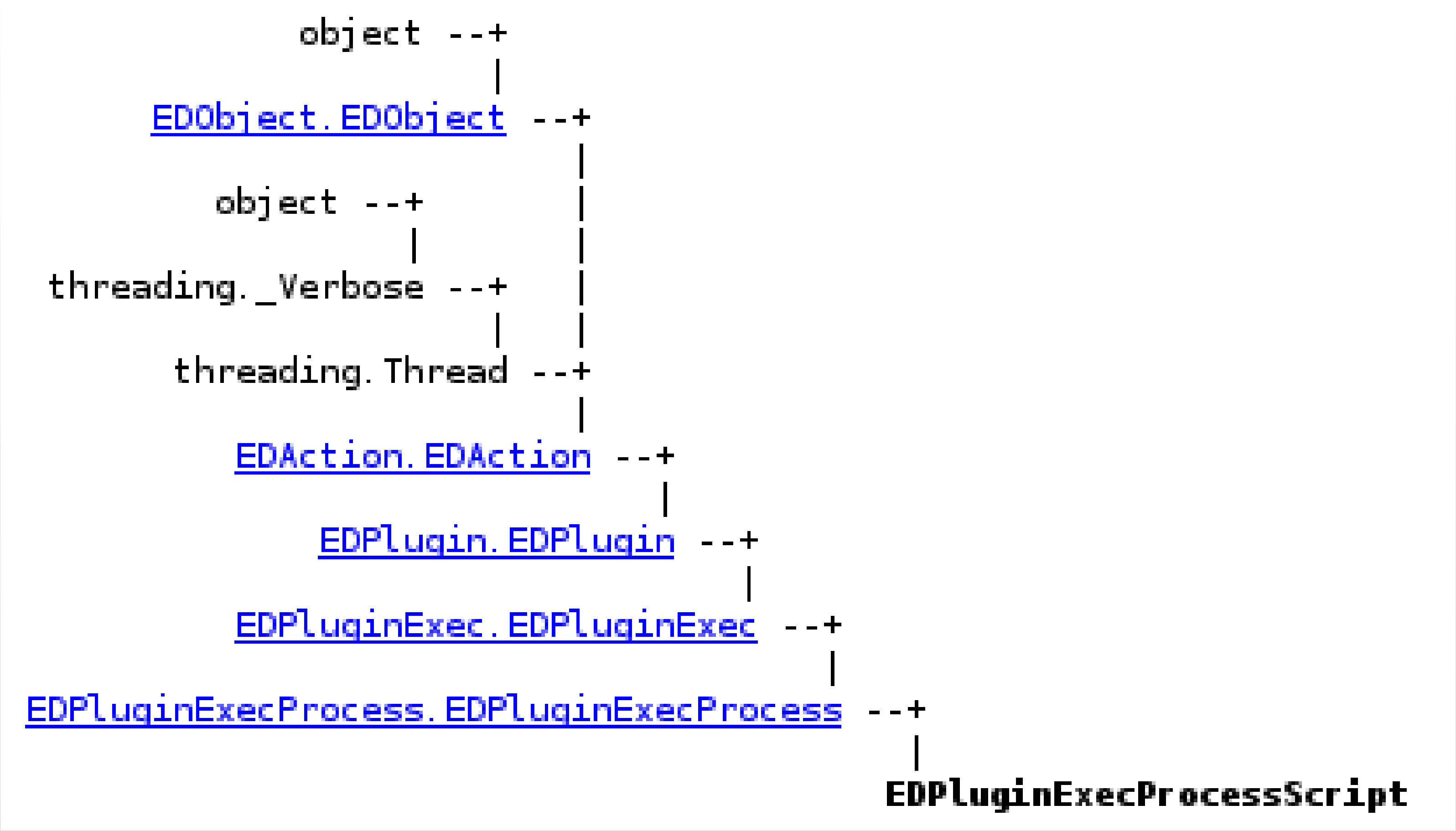
- + rawImagePath: XSDataFile
- + outputPath: XSDataFile [0..1]
- + exportTiff: XSDataBoolean [0..1]
- + extractThumbnail: XSDataBoolean [0..1]
- + whiteBalanceAuto: XSDataBoolean [0..1]
- + whiteBalanceFromCamera: XSDataBoolean [0..1]
- + levelsFromCamera: XSDataBoolean [0..1]
- + interpolationQuality: XSDataInteger [0..1]

- XSD file:



```
<xs:complexType name="XSDataInputExecDcrawv1">
    <xs:complexContent>
        <xs:extension base="XSDataInput">
            <xs:sequence>
                <xs:element name="rawImagePath" type="XSDataFile" minOccurs="1" maxOccurs="1"/>
                <xs:element name="outputPath" type="XSDataFile" minOccurs="0" maxOccurs="1"/>
                <xs:element name="exportTiff" type="XSDataBoolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="extractThumbnail" type="XSDataBoolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="whiteBalanceAuto" type="XSDataBoolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="whiteBalanceFromCamera" type="XSDataBoolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="levelsFromCamera" type="XSDataBoolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="interpolationQuality" type="XSDataInteger" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
```

Base classes



```
             object --+
                      |
      EDObject.EDObject --+
                          |
         object --+       |
                  |       |
 threading._Verbose --+   |
                      |   |
       threading.Thread --+
                          |
          EDAction.EDAction --+
                              |
              EDPlugin.EDPlugin --+
                                  |
          EDPluginExec.EDPluginExec --+
                                      |
EDPluginExecProcess.EDPluginExecProcess --+
                                          |
                                         EDPluginExecProcessScript
```

Utilities: Static classes

- EDVerbose: everything related to logging (will be changed)
- EDUtilsArray: for handling numpy-like arrays
- EDUtilsFile:
- EDUtilsImage:
- EDUtilsLibraryInstaller: used to install libraries on the fly
- EDUtilsParallel: detection of the number of processors
- EDUtilsPath:
- EDUtilsSymmetry: static methods useful for handling symmetries
- EDUtilsTable: Related to old DNA tables
- EDUtilsUnit: For handling units and sub-units
- EDUtilsXML: to encapsulate XML in XSD

EDNA conventions

- Most the EDNA initial developers came from Java
  - Inheritance model: (too?) many level of inheritance
  - CamelCase notation for class and variable names
  - Hugarian notation: strInputFilename

Edna toolbox

- 70 Execution plugins :
  - Generic command line execution, Image conversion, movied ...
  - HDF5 writers for stack of images, map of spectra
  - Conditional branching, accumulator of information
- 5 real applications available from repository (2 demo)
  - BioSaxs
  - Ccp4 (DEMO)
  - DiffractionCT
  - Dimple
  - MX v1 & v2
  - Raw photography development (DEMO)
- 3 kind of launchers: Command line, Parallel, Tango

EDNA Launchers

- Command line launcher:

```
$EDNA_HOME/kernel/bin/edna-plugin-launcher.sh \
--exec EDPluginName \
--inputFile pathToConfig.xml \
```

- Tango device server to which you will provide:
  - Name of the plugin
  - XML configuration, as string, following to the datamodel.

  Available in $EDNA_HOME/tango/bin/tango-EdnaDS.py
- Parallele Execute scripts in the bin directories of your projects; to be copied and hacked.
  - $EDNA_HOME/execPlugins/plugins/EDPluginExecThumbnail/bin/edna-png.py

  options are: -ncpu=8, --debug, ...
-

Hand's on [1]

- Install EDNA
- Configure it
- Have it running

http://www.edna-site.org/images/tutorial1.flv

Get EDNA from web ...

- Tested nightly build are available on the net:

  http://www.edna-site.org/pub/nightly/

- Download the latest version (update the command)

  ```
  wget http://www.edna-site.org/pub/nightly/EDNA-20101028-ExecPlugins-rev2271.tar.gz
  ```

- Don't forget to setup your proxy (if you are at ESRF)
  - export http_proxy=http://proxy.esrf.fr:3128
  - Setup the proxy even under Windows, unless tests will not work !

Unzip the archive

- Unzip archive:

  ```
  tar -xvzf EDNA-20101028-ExecPlugins-rev2271.tar.gz
  ```

- Define a couple of environment variables:
  - export EDNA_HOME=$PWD/edna
  - export EDNA_SITE=ESRF
- EDNA_HOME refers to the location of the EDNA install
  - We are working to auto-guess it but not yet everywere
- EDNA_SITE refers to the locale configuration.

  All configurations are in XML files in:

  $EDNA_HOME/project/conf folder.
- PYTHON if you want to specify the path of you python

Run the tests.

- Check that everything is working
  - Python version (2.5 <= version < 3.0
  - Dependencies (EDNA will compile some of the missing dep.)
  - Configuration files and external executables are available
- Tests of the EDNA Kernel:

```
$EDNA_HOME/kernel/bin/edna-test-launcher.sh –test EDTestSuiteKernel
```

- Useful options available: --debug
- Tests of the EDNA execPlugins tool box:

```
$EDNA_HOME/kernel/bin/edna-test-launcher.sh –test EDTestSuitePluginExecPlugins
```

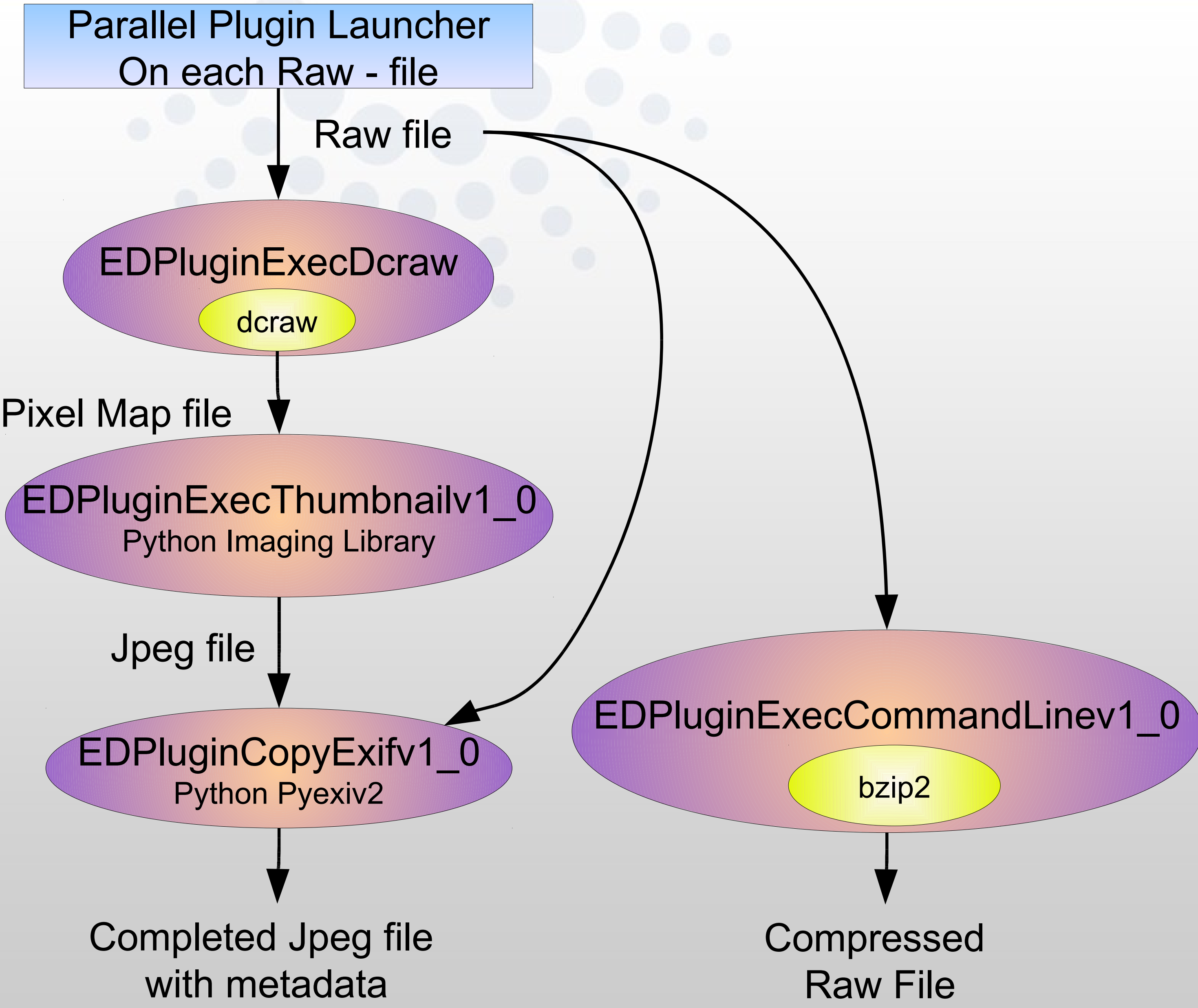
- This will download, compile and install all libraries like: numpy, scipy, PIL, h5py, … automatically (can be long !)

Project:

Create a pipeline for Raw Images development.

Subdivision of tasks

- Develop a raw image into a 8-bits RGB array
  - Use the ***dcraw*** utility to develop the raw
  - Output can be Tiff or pixel maps format (PGM/PPM)
- Compress the image in Jpeg
- Transfer Exif metadata from Raw to Jpeg
- Optional things:
  - Crop the image to remove the black borders
  - Compress the input raw image for archiving
  - Archive the raw data

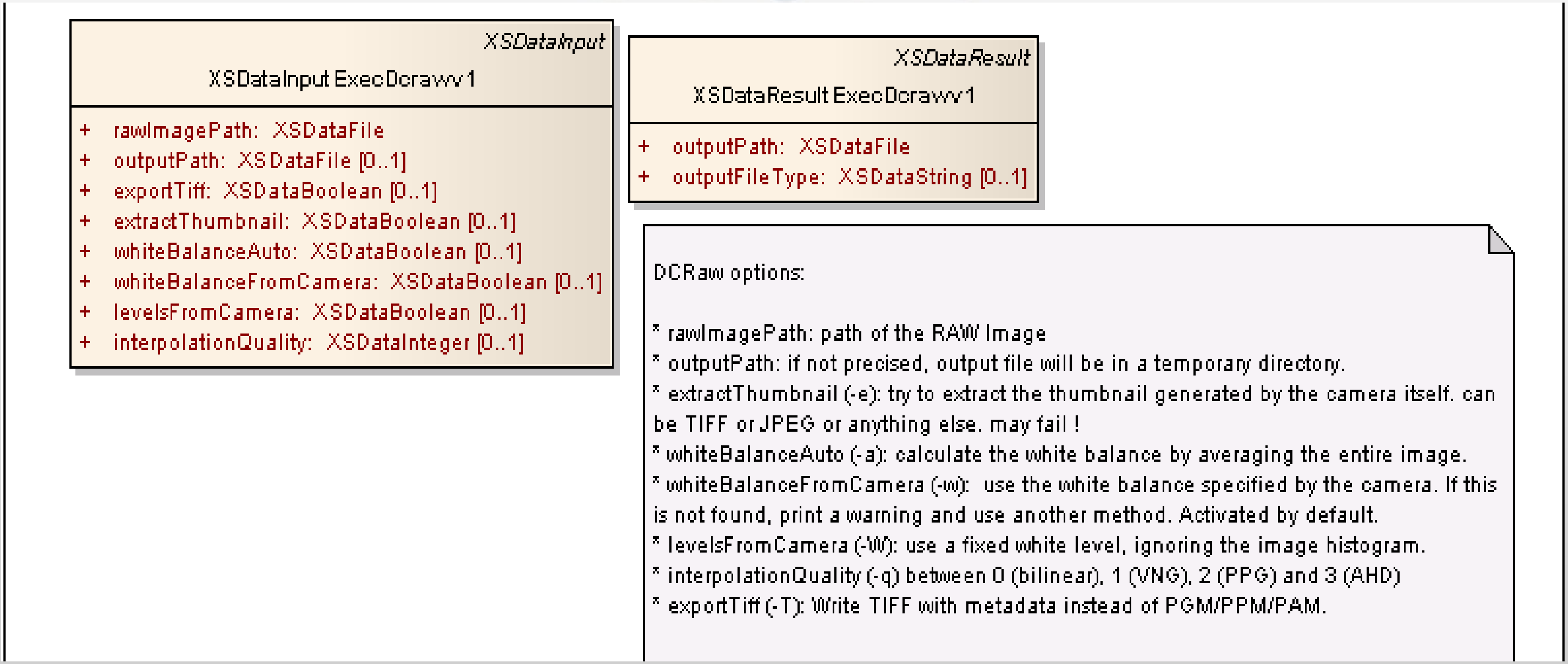

Parallel Plugin Launcher
On each Raw - file
Raw file
EDPluginExecDcraw
dcraw
Pixel Map file
EDPluginExecThumbnailv1_0
Python Imaging Library
Jpeg file
EDPluginCopyExifv1_0
Python Pyexiv2
EDPluginExecCommandLinev1_0
bzip2
Completed Jpeg file
with metadata
Compressed
Raw File

Install UML Modeling tool

- Download Enterprise Architect from:
  - http://www.sparxsystems.com.au/bin/easetup.exe
  - Demo version for 30 days :(
- Usable under windows, Wine or with virtualisation

- Or Use TopCased:
  - Get EMF SDF from Galileo (or Helios) update site
  - Install TopCased UML modeling tool
  - Install XSD2UML from edna-site

Datamodel for EDPluginExecDcraw



XSDataInput
XSDataInputExecDcrawv1

+ rawImagePath: XSDataFile
+ outputPath: XSDataFile [0..1]
+ exportTiff: XSDataBoolean [0..1]
+ extractThumbnail: XSDataBoolean [0..1]
+ whiteBalanceAuto: XSDataBoolean [0..1]
+ whiteBalanceFromCamera: XSDataBoolean [0..1]
+ levelsFromCamera: XSDataBoolean [0..1]
+ interpolationQuality: XSDataInteger [0..1]

XSDataResult
XSDataResultExecDcrawv1

+ outputPath: XSDataFile
+ outputFileType: XSDataString [0..1]

DCRaw options:

* rawImagePath: path of the RAW Image
* outputPath: if not precised, output file will be in a temporary directory.
* extractThumbnail (-e): try to extract the thumbnail generated by the camera itself. can be TIFF or JPEG or anything else. may fail !
* whiteBalanceAuto (-a): calculate the white balance by averaging the entire image.
* whiteBalanceFromCamera (-w): use the white balance specified by the camera. If this is not found, print a warning and use another method. Activated by default.
* levelsFromCamera (-W): use a fixed white level, ignoring the image histogram.
* interpolationQuality (-q) between 0 (bilinear), 1 (VNG), 2 (PPG) and 3 (AHD)
* exportTiff (-T): Write TIFF with metadata instead of PGM/PPM/PAM.

A Light for Science

ESRF

European Synchrotron Radiation Facility