Journal of
Synchrotron
Radiation

ISSN 0909-0495

Received 26 June 2009
Accepted 10 September 2009

EDNA: a framework for plugin-based applications applied to X-ray experiment online data analysis

Marie-Françoise Incardona,[a] Gleb P. Bourenkov,[b] Karl Levik,[c] Romeu A. Pieritz,[a] Alexander N. Popov[a] and Olof Svensson[a]*

[a]ESRF, 6 Rue Jules Horowitz, 38043 Grenoble, France, [b]EMBL Hamburg Outstation, c/o DESY, Notkestrasse 85b, 22607 Hamburg, Germany, and [c]Diamond Light Source Ltd, Didcot, Oxfordshire OX11 0DE, UK. E-mail: svensson@esrf.fr

EDNA is a framework for developing plugin-based applications especially for online data analysis in the X-ray experiments field. This article describes the features provided by the *EDNA* framework to ease the development of extensible scientific applications. This framework includes a plugins class hierarchy, configuration and application facilities, a mechanism to generate data classes and a testing framework. These utilities allow rapid development and integration in which robustness and quality play a fundamental role. A first prototype, designed for macromolecular crystallography experiments and tested at several synchrotrons, is presented.

Keywords: online data analysis; beamline automation; data collection.

1. Introduction

Recent advances of X-ray beamlines technologies, including the advent of very high brilliance beamlines at third-generation synchrotron sources and advanced detector instrumentation, have led to an exponential increase in the speed of data collection. As a consequence, there is an increasing need for a data analysis platform that can refine and optimize data collection strategies online. The requirements for online data analysis include the automation or semi-automation of the experimental steps (involving either computer or operator decisions), feedback of results and indicators of data quality in (quasi) real time, the capability of high data throughput rates, and graphical user interfaces (GUIs) that provide remote access potentially with the possibility to perform experiments on a distributed synchrotron environment. In recent years, software solutions have been developed to meet these requirements in different X-ray experimental fields such as macromolecular crystallography (Lamzin & Perrakis, 2000; Leslie *et al.*, 2002; Beteva *et al.*, 2006; Leonard *et al.*, 2007, Soltis *et al.*, 2008; Minor *et al.*, 2006; Yamada *et al.*, 2008), small-angle X-ray scattering (Konarev *et al.*, 2006) and X-ray absorption spectroscopies (Sanchez del Rio, 1997; Korbas *et al.*, 2006; Leonard *et al.*, 2009).

One of the challenges for online data analysis software resides in the capability of smoothly integrating extensions to existing software. Adding new features in a flexible and configurable way or invoking different external scientific programs for a particular step to be processed (either sequentially or in parallel) is indeed the key point of modern object-oriented software. The trend in many synchrotron facilities is to adopt modern software technologies based on components architecture including CORBA technology applied for device control systems (Chaize *et al.*, 1999; Gibbons & Pulford, 2006) and/or graphical brick components to build Python, Java or Eclipse RCP GUIs of higher level (Guijarro, 2004; Solé *et al.*, 2007; Gotz, 2007).

Another important consideration is the integration of a data management system that allows a complete definition of the experimental data, which can be quite complex in some cases. Moreover, an efficient error-tracking mechanism is crucial for the propagation of potential warning and error messages in a user-friendly way. Finally, a strong testing framework is essential to guarantee code robustness, quality confidence and, thus, low maintenance costs.

The *EDNA* framework described here has been developed to address these challenging issues. Based on an object-oriented plugin architecture, *EDNA* is a technical platform (or framework) that incorporates scientific components (or plugins) development facilities, data model definition utilities, a mechanism for automated data classes generation and a testing framework. Having such a framework is particularly relevant for collaborative scientific software development. It allows a rapid development process of extensible and customizable X-ray online data analysis applications with low regression risks and high-quality confidence.

We developed a first application prototype with this framework, edna-prototype, which aims to fully automate the calculation of optimized data collection strategies for macromolecular crystals at synchrotron beamlines, taking radiation damage into account. The *EDNA* framework and this first prototype are described in this article.

2. Definitions

A *plugin* is a re-usable and configurable software component that can be dynamically loaded by an application in order to extend its existing functionalities.

In computing, a *framework* is a modular workbench that provides a set of libraries, tools and conventions that are used in the development of applications. It provides the necessary software bricks and imposes sufficient rules to allow the construction of accomplished applications that are both robust and easy to maintain.

A *kernel* is the core of a system. It provides all the necessary features on top of which upper layers can be developed.

The goal of a *unit test* is to isolate each part of a program and show that the individual parts are correct.

In object-oriented programming, the data structures consisting of data fields and methods are also called *data classes*. A *data model* is the abstract description of the data classes, showing how data are represented and accessed.

3. The *EDNA* framework

The main goal here is to provide answers to typical issues of online data analysis. For example,

(i) Complex and heterogeneous input data.

(ii) Technical configuration: technical parameters needed for individual workflow steps to run properly, *e.g.* identification of third-party programs for intermediate steps.

(iii) Transfer of complex heterogeneous data between the different steps.

(iv) Performance (set by the high speed of experiments) and a necessity of parallel processing whenever possible.

(v) Feedback on the results and data along with quality indicators.

(vi) Robustness.

The *EDNA* framework is composed of a kernel, corresponding to the core of the system, and a set of functional components. The *EDNA* kernel provides the necessary utilities for building configurable components (plugins), data models and applications, taking all the above-mentioned technical issues into consideration. It also provides a testing framework to facilitate the development of unit tests and execution tests. The main components of the *EDNA* kernel are described in the following sections.

3.1. Plugins facility

The components are organized in a logical class hierarchy (Fig. 1). This hierarchy makes it straightforward to develop new functional plugins by deriving them from the appropriate parent. Two families of plugins have been designed: the first branch contains the execution plugins (EDPluginExec classes) that are responsible for the execution of a particular action (*e.g.* execution of third-party software); the second branch contains the control plugins (EDPluginControl classes) that are responsible for the data flow (propagation of the data), the workflow (sequential or parallel execution of appropriate execution plugins) and the error-tracking mechanism (propagation of the errors).

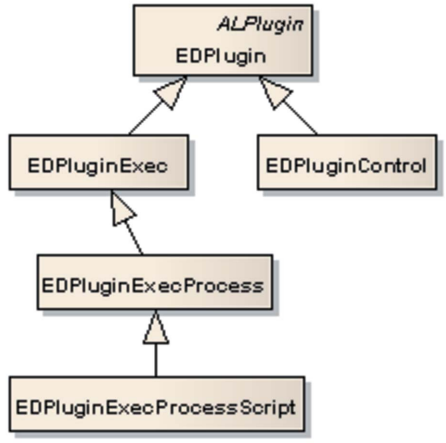

Figure 1
EDNA plugin class hierarchy. In addition to the features inherited from the basic AALib class ALPlugin (Pieritz, 2007), the *EDNA* common parent class EDPlugin provides configurable properties and can have input and output data. The EDPluginExec group is related to the execution of specific actions, whereas the EDPluginControl group is responsible for the data flow and the workflow. EDPluginExecProcess is a particular EDPluginExec that is able to execute an external process. EDPluginExecProcessScript is a specialized EDPluginExecProcess able to launch an external process *via* a script.

The common parent of both families is the EDPlugin class. This class allows all derived plugins to have input and output data and to be configurable *via* the application configuration file (see §3.3) that should contain all the technical parameters needed for the plugin to run properly (working directory, executable to invoke *etc.*). The plugin input and output data include all the scientific parameters to be given to and returned by the plugin. The plugin input and output data, formatted in XML (extensible markup language), can be defined and described in a specific data model (see following section).

Each plugin is stored in a specific repository directory that contains the related code sources, data model and tests suites.

3.2. Data model facility

Describing the data is particularly relevant for applications dealing with X-ray experiments where the data can be quite complex. Organizing them in a data model is beneficial for both documentation and communication purposes and also makes implementation more straightforward. It allows automatic data classes generation, which is a valuable feature for productivity and reproducibility.

The *EDNA* kernel provides a data model tool kit that allows the construction of elaborated data models needed by advanced applications. In addition, it allows the design of unitary data models that can be unitary tested, so that a plugin can be launched and tested independently of any application context. This tool kit consists of generic low-level class definitions including general types (XSDataString, XSDataFloat *etc.*) and X-ray experiment classes which can be re-used when designing specific components data models. It is available in

```
<?xml version="1.0" ?>
<XSConfiguration>
        <XSPluginList>
        <XSPluginItem>
                <name>EDPluginMOSFLMv01Indexing</name>
                <XSParamList>
                        <XSParamItem>
                                <name>execProcessScriptShell</name>
                                <value>/bin/bash</value>
                        </XSParamItem>
                        <XSParamItem>
                                <name>execProcessScriptExecutable</name>
                                <value>/path/to/ipmosflm-7.0.1-20aug07</value>
                        </XSParamItem>
                        <XSParamItem>
                                <name>execProcessScriptSetupCCP4</name>
                                <value>/path/to/ccp4.setup-bash.orig</value>
                        </XSParamItem>
                        <XSParamItem>
                                <name>execProcessScriptVersionString</name>
                                <value>Version 7.0.1  for Image plate and CCD data 20th August 2007</value>
                        </XSParamItem>
                        <XSParamItem>
                                <name>execProcessTimeOut</name>
                                <value>600</value>
                        </XSParamItem>
                </XSParamList>
        </XSPluginItem>
        <XSPluginList>
</XSConfiguration>
```

Figure 2
XML application configuration file. Each plugin of the application can be configured *via* an XSPluginItem section. A particular plugin parameter can be configured by setting up an XSParamItem section.

several standard data formats including XMI (XML metadata interchange) and XSD (XML schema definition) in order to facilitate importing into UML (unified modeling language) data modelling tools and/or XSD files. The framework also includes code generation machinery (Kuhlman, 2004) that allows automatic code generation from UML diagrams to Python code *via* XSD format.

3.3. Application facility

The *EDNA* kernel also includes the EDApplication class, that can be used as the parent class of more specialized applications. Owing to its capability to launch any given plugin (either single or several encapsulated in an EDPluginControl) and to its configuration properties, EDApplication makes the application system highly generic and flexible. Two means of launching a particular plugin are provided. The first one is *via* the edna-prototype-plugin-launcher command by giving the plugin to be executed (–execute <pluginName>), the XML application configuration file (–conf <configuration FileName>; Fig. 2) and the XML input data file of the related plugin (–inputFile <dataFileName>). The second one is *via* the EDApplication constructor. All the available options can be consulted *via* the –help option.

3.4. Testing facility

A testing framework has been developed and integrated with the kernel in order to test the applications and the components easily and efficiently. To ensure the reliability and robustness of the components, the testing framework provides all the necessary utilities to check a class (EDTestCase) and a plugin (EDTestCasePlugin) either in a unitary manner (EDTestCasePluginUnit) or by testing its execution in an application context (EDTestCasePluginExecute). These test classes are hierarchically organized as described in Fig. 3. These families of tests can be automatically launched *via* test suites. An automatic analysis of the test results is performed by comparing the obtained result with the expected one (assert mechanism), so that a successful test proves that a result conforms to the expectation (Fig. 4). This allows a high degree of confidence when implementing new components or re-implementing features of existing components.

4. MX data collection application prototype

4.1. Scientific use case

Typical data collection in macromolecular crystallography (MX) using the rotation method (Arndt & Wonnacott, 1977) involves preliminary steps: measuring a few diffraction patterns (reference images) of a crystal under investigation, and their evaluation. The procedure aims to determine the basic crystallographic parameters (unit-cell dimensions and putative Laue class, crystal orientation, mosaicity, diffraction spot shapes) and to evaluate the signal-to-noise ratio in the data. On the basis of these processing results, the data collection parameters (data collection resolution, rotation range, oscillation width and exposure time, jointly called a 'data collection strategy') are determined, along with the decision on suitability of a sample for a particular crystallographic problem. An additional consideration that must be taken into account in a modern data collection procedure at third-generation beamlines is an anticipated sustainable radiation dose for the crystal (Owen *et al.*, 2006; Ravelli & Garman, 2006). At weaker sources, the total time

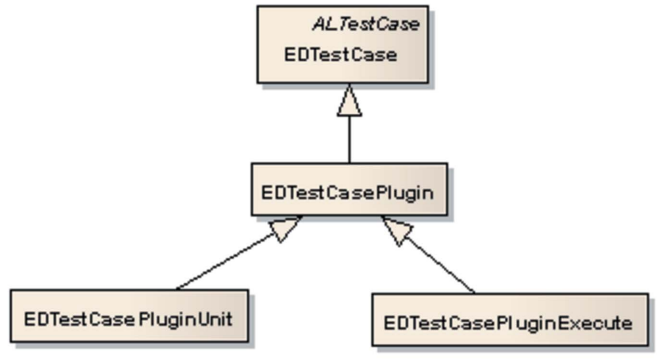

Figure 3
EDNA test classes hierarchy. This organization facilitates the development of class tests as well as the plugin unit tests and execution tests.

```
================================================================================

  [SUCCESS] [ 1 ][ EDTestCaseEDConfiguration.execute ][0.09827995300029]
          [SUCCESS] [ 1 ][ EDTestCaseEDConfiguration.testGetPluginList ][0.00691819190979]
          [SUCCESS] [ 2 ][ EDTestCaseEDConfiguration.testGetPluginItem ][0.01253199557733]
          [SUCCESS] [ 3 ][ EDTestCaseEDConfiguration.testGetPluginItemError ][0.00690197944641]
          [SUCCESS] [ 4 ][ EDTestCaseEDConfiguration.testGetParamItem ][0.0071759223938]
          [SUCCESS] [ 5 ][ EDTestCaseEDConfiguration.testGetParamValue ][0.0155029296875]
          [SUCCESS] [ 6 ][ EDTestCaseEDConfiguration.testGetOptionItem ][0.013533115387]

  [SUCCESS] [ 2 ][ EDTestCaseEDPluginExecProcessScript.execute ][1.69729089737]
          [SUCCESS] [ 1 ][ EDTestCaseEDPluginExecProcessScript.testGenerateScript ][0.0253608226776]
          [SUCCESS] [ 2 ][ EDTestCaseEDPluginExecProcessScript.testGenerateExecutableScript ][0.0268001556396]
          [SUCCESS] [ 3 ][ EDTestCaseEDPluginExecProcessScript.testExecuteScript ][1.62124013901]

  [SUCCESS] [ 3 ][ EDTestCaseEDUtilsImage.execute ][0.0826640129089]
          [SUCCESS] [ 1 ][ EDTestCaseEDUtilsImage.testGetImageNumber ][0.0139808654785]
          [SUCCESS] [ 2 ][ EDTestCaseEDUtilsImage.testGetTemplateHash ][0.0166668891907]
          [SUCCESS] [ 3 ][ EDTestCaseEDUtilsImage.testGetTemplateQuestionMark ][0.00994682312012]
          [SUCCESS] [ 4 ][ EDTestCaseEDUtilsImage.testReadHeaderADSC ][0.013573884964]

  [SUCCESS] [ 4 ][ EDTestCaseEDUtilsTable.execute ][0.222398042679]
          [SUCCESS] [ 1 ][ EDTestCaseEDUtilsTable.testTableListItem ][0.09814786911101]
          [SUCCESS] [ 2 ][ EDTestCaseEDUtilsTable.testListOfTables ][0.105435848236]

  [SUCCESS] [ 5 ][ EDTestCaseEDUtilsSymmetry.execute ][0.0567150115967]
          [SUCCESS] [ 1 ][ EDTestCaseEDUtilsSymmetry.testGetMinimumSymmetrySpaceGroupFromBravaisLattice ]
[0.0415990352631]

  ================================================================================


  [UnitTest]: ######################################################################
  [UnitTest]: Summary Report:
  [UnitTest]:                 Total TestCases: 5
  [UnitTest]:     Total TestCases [SUCCESS]: 5
  [UnitTest]:           Total TestCases [FAIL]: 0
  [UnitTest]:             [Total TestMethods]: 16
  [UnitTest]:                         Runtime: 12.5 [s]
  [UnitTest]:                              Run: 00d:00h:00m:12s:506ms
```

Figure 4
Test Report of the *EDNA* Kernel Test suite.

available for the experiment may form a significant restriction. Such procedures (manual, semi- or fully automated) are essential for obtaining acceptable-quality diffraction data and may not be replaced by, for example, a set of 'standard' data collection conditions that are satisfactory for most samples (Dauter, 1999; Evans, 1999).

The individual steps of reference image interpretation and/or determination of a subset of data collection strategy parameters (rotation range and oscillation width) are readily performed by available data processing packages [*MOSFLM* (Leslie, 1992), *LABELIT* (Sauter *et al.*, 2004), *XDS* (Kabsch, 1993), *HKL* (Otwinowski & Minor, 1997), *D*TRACK* (Pflugrath, 1997)]. Given the results of reference image processing and the signal-to-noise requirements of the proposed experiment, the program *BEST* (Popov & Bourenkov, 2003) performs joint optimization of a complete set of data collection parameters, which notably includes the exposure time and data collection resolution. Thereby, the anticipated effects of radiation damage on the signal-to-noise ratio in the data is taken into account quantitatively (Bourenkov & Popov, 2006; Bourenkov *et al.*, 2006). An essential parameter for the strategy calculation is the X-ray dose deposition rate, which is in turn defined by the chemical composition of the sample, the incident beam energy and the flux density, and can be estimated using the *RADDOSE* software (Murray *et al.*, 2004, 2005). The key feature of *BEST* strategies is that the data are collected in a few wedges with the exposure time and oscillation width varying over the rotation range in order to compensate for the signal-to-noise degradation owing to the radiation-induced decay of the scattering power, as well as to adapt the measurement conditions according to the intrinsic anisotropy of diffraction and crystal orientation with respect to the beam.

The main goal of this *EDNA* MX application prototype is to provide the software which implements the procedure described above in a fully automated way. The main characteristics of online data analysis are addressed (as described in §3). In particular, the complex and heterogeneous input data, being taken into account here, are the reference images with their experimental conditions, optional prior knowledge (space group), sample description and diffraction plan (experiment requirements such as required resolution, completeness, multiplicity *etc.*).

In the following sections we present a brief description of the *EDNA* MX application prototype and initial test results. Full details of this implementation and experimental results will be published elsewhere (Svensson *et al.*, in preparation).

4.2. Workflow

The workflow of the *EDNA* MX application prototype is presented in Fig. 5. The indexing step may be implemented using either *MOSFLM* or *LABELIT*. At this step, a set of putative indexing solutions (Bravais lattice and unit cell) are reported to the user. One solution is selected for further processing by *MOSFLM* (or *LABELIT* depending on the workflow) either according to the *a priori* known space group of the crystal supplied by the user or fully automatically. In the latter case, the lowest symmetry space group matching the lattice symmetry is chosen. The indexing solutions are refined with the lattice symmetry constraints applied, and the crystal mosaicity is estimated. The results of refinement, *i.e.* the number of indexed reflection spots, r.m.s. deviations of predicted-to-observed spot positions, and the shifts of the refined direct beam coordinates, are used as quantitative criteria to judge the success and accuracy of the indexing solution. For a further control, a grayscale representation of the diffraction pattern (in JPEG format) with the predicted spot positions overlaid is generated for each of the reference images.

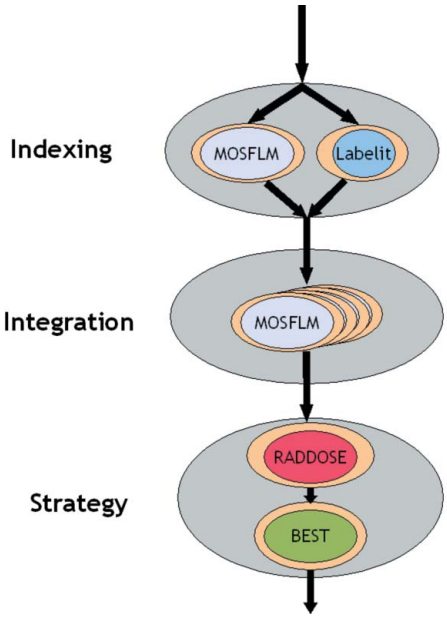

Figure 5
EDNA MX application prototype workflow.

The selected indexing solution and the refined beam coordinates are then used in an integration step. At the same time, the radial X-ray background is measured as required for strategy calculations by *BEST*. Integrations (with *MOSFLM*) are carried out for each of the reference images in parallel. The sample characterization output of the integration step comprises the data statistics (average ratios of intensities to their estimated standard uncertainties, I/SigI) displayed in the appropriate resolution shells for each reference frame separately.

The strategy plugin combines the dose rate calculation from *RADDOSE* and the strategy calculation by *BEST*. For the dose rate calculation, the content of the crystallographic asymmetric unit, the chemical composition of crystallized biomolecule(s), ligand(s) and the crystallization solution need to be provided by the user. The *EDNA* application prototype has an extensive data model for the crystal contents which permits any crystal structure to be accurately described (Fig. 6). The consistency of the indexing solution with the specified crystal contents is checked *via* the solvent content calculations (Matthews, 1968); a warning is issued if the predicted solvent content is outside the range 25–75%. It is possible to optionally use a default protein chemical composition.

At a final step of the procedure, the program *BEST* is invoked. The input to *BEST* comprises the results of previous processing of the reference images and user requirements for the experimental data formulated in a 'diffraction plan' (Beteva *et al.*, 2006). The minimum requirements are the target I/SigI in the outer resolution shell, and the maximum total exposure time for the data set. Optionally, the diffraction plan may include the desired values for the resolution limit and data completeness, the requirement to collect data with a given (high) multiplicity, the limitation on the oscillation width

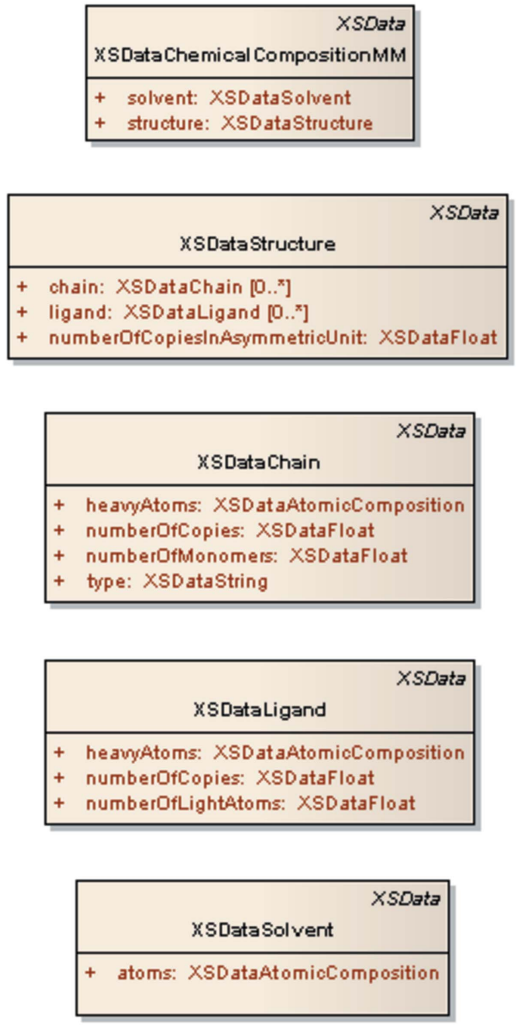

Figure 6
Data model and definition of the chemical composition of a macromolecular crystal. A crystal is composed of solvent (XSDataSolvent) and biopolymer molecules (XSDataStructure). Basically, a biopolymer is composed of protein/nucleic acid chains (referred to as 'type' in the above data model) and ligands. For each chain, information about the number of monomers (amino acids or nucleotides), number of identical chains in the polymer and heavy atoms identification should be provided. For each ligand, information about heavy and light atoms and number of copies of ligand in the molecule should be provided as well.

per frame, as well as the possibility to disable the variation in the oscillation width and exposure time *versus* rotation angle (multiple sub-wedge strategy feature). The strategy output includes an optimized plan for data collection (Table 1) and predicted statistics for the data that will be collected according to the plan (Table 2). The standard completeness, multiplicity, I/SigI and $R_{\rm merge}$ statistics in the resolution shells are defined in such a way that they are directly comparable with the results of data processing using standard data processing packages.

4.3. Implementation

This prototype has been designed and developed with the aim of being easily configurable, extensible and smoothly maintainable. This has been made possible thanks to the

Table 1
Plan of data collection for a SurE crystal.

N	φ start (°)	Rotation width (°)	Number of images	Exposure per image (s)
1	54.0	0.3	267	14.1
2	134.1	0.3	83	40.5
3	159.0	0.3	17	85.1

Table 2
Data processing statistics.

Corresponding predicted statistics (from *BEST*) are given in brackets: $R_{\rm merge} = \Sigma_{hkl}\Sigma_i|I_{i(hkl)} - \langle I_{(hkl)}\rangle|/\Sigma_{hkl}\Sigma_i I_{i(hkl)}$. $\langle I_{hkl}\rangle$ denotes averaging over symmetry-equivalent and redundant measurements.

	Overall	Inner shell	Outer shell
Low-resolution limit (Å)	20.00	20.00	3.06
High-resolution limit (Å)	2.90	9.17	2.90
$R_{\rm merge}$ (%)	8.1 (6.5)	4.8 (2.6)	29.4 (42.1)
Mean [(I)/sd(I)]	11.7 (14.2)	30.7 (35.2)	4.0 (3.0)
Completeness (%)	97.4 (98.9)	93.2 (95.3)	99.6 (99.7)

technical facilities that the *EDNA* framework provides, including configuration facilities, a library of re-usable components, data-driven code generation machinery and a testing framework.

A GUI (Fig. 7) and a command-line mode are available for launching the application prototype. In the command line mode, the input is either an XML file comprising a complete description of the reference images, the beam parameters, diffraction plan and (optionally) chemical composition of the sample, or the reference image files themselves. In the latter case, the experimental parameters that are not defined in the image headers, as well as the diffraction plan parameters, can be defined *via* specific command line arguments. A basic Tcl/Tk GUI integrated in the CCP4 (Collaborative Computational Project, Number 4, 1994) suite has been developed with the aim of facilitating the input of all required parameters that could be difficult to do with the command-line mode. The output (executive textual output and prediction images) are displayed to the user *via* the CCP4i GUI. Furthermore, complete results of all steps of the processing are available in an XML file, and can be easily linked to data collection GUIs. This has been implemented within the *CBASS* software at the National Synchrotron Light Source (Brookhaven) beamlines (J. Skinner, private communication).

4.4. Testing

Several test data collections using this *EDNA* application prototype have been carried out at the European Synchrotron Radiation Facility beamlines. Here we present the example of data collections for the orthorhombic crystal form of SurE protein from *Campylobacter jejuni* (Gonçalves *et al.*, 2008). The SurE crystals belong to the space group $P2_12_12_1$ with four molecules in the asymmetric unit. Each chain consists of 267 amino acid residues. The experiment was carried out at the beamline ID14-2 using the wavelength 0.93 Å and an ADSC Q4 detector. The X-ray beam cross section was 0.15 × 0.15 mm (FWHM), which matched (slightly exceeded) the crystal size. The photon flux was 1.2×10^{11} photon s^{-1}, measured using an intensity monitor. Two reference images at rotation angles 0° and 90° were measured (Fig. 8) and used for sample characterization by the prototype. Complete characterization output has been deposited.[1]

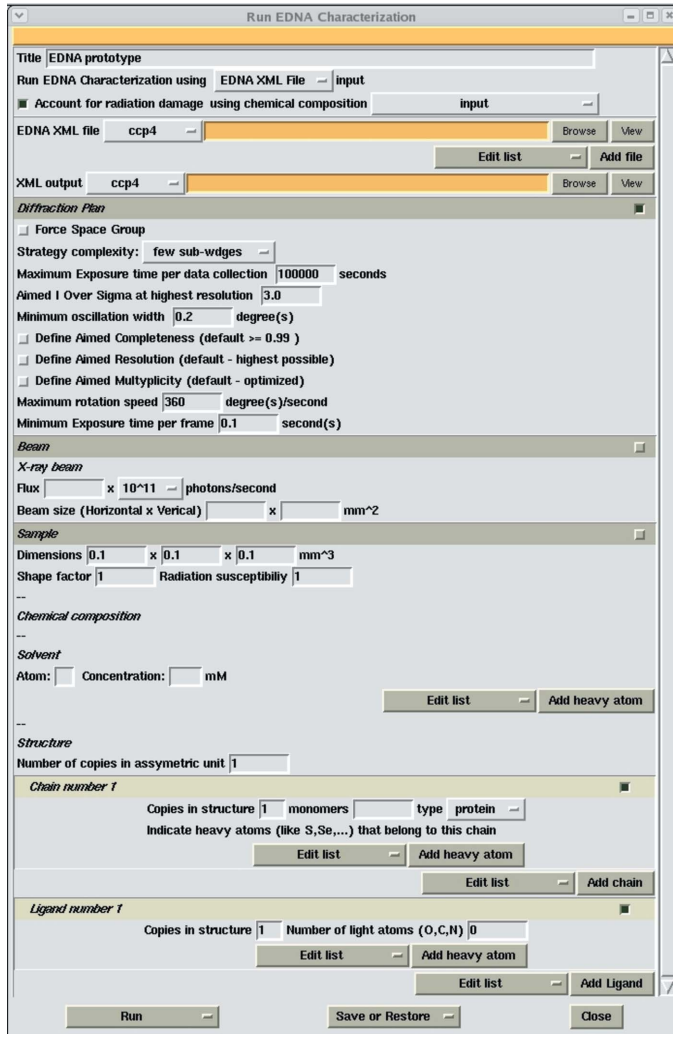

Figure 7
EDNA MX application prototype GUI screen shot. The GUI is composed of four panels. The first panel defines the input data (images or an XML experiment descriptor) and type of the radiation damage model to be used based on an average protein crystal composition or explicitly defined crystal composition, if the account for radiation damage mode is selected. The second panel defines the diffraction plan. The third (beam parameters definitions) panel is enabled when the radiation-damage-based strategy option is active; and an optional fourth panel is activated when explicit chemical composition definition is requested.

The crystal exhibited rather poor diffraction quality, with mosaicity 0.6° and an apparent diffraction limit of ∼3.3 Å for an exposure time of 5 s. For strategy calculations, an I/SigI value of 3 for the outer-resolution shell was requested. The *EDNA* prototype estimated that the maximal resolution that

[1] Supplementary data for this paper are available from the IUCr electronic archives (Reference: WA5014). Services for accessing these data are described at the back of the journal.

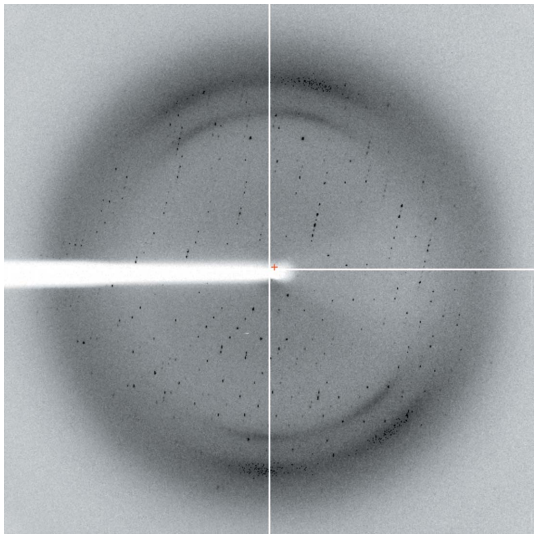

Figure 8
Reference diffraction image of a SurE crystal. The resolution limit is 2.8 Å at the edge of the detector. Exposure time 5 s.

could be achieved with the given statistical target, 2.9 Å, was limited by radiation damage. The resulting plan for data collection consisted of three entries with a stepwise increase in exposure time per image (Table 1) to compensate for the diffraction intensity decrease owing to the radiation damage. Incidentally, the optimal oscillation width (0.3°) did not vary across the rotation range.

The data collected following this plan were processed using *MOSFLM* and scaled using *SCALA* (Evans, 2006). The processing results are shown in Table 2. The data statistics are in good agreement with target values for this data collection. The predicted I/SigI and $R_{\rm merge}$ statistics are marginally worse than the experimental results. We attribute this discrepancy to a slight overestimation of the photon flux.

5. External libraries, language, platform, tools and availability

The *EDNA* framework is open-source software distributed under the GPL v3 license. It can be downloaded from the web site http://www.edna-site.org/.

The source code is written in Python (version 2.4). Two open-source libraries have been incorporated in the framework to meet the requirements of having a system that is able to launch parallel steps and can also generate Python code from XSD definition files. The two libraries are, respectively, the Asynchronous Action Library (AALib), which is a library based on object-oriented concepts to mainly facilitate asynchronous communication between threads and external processes (Pieritz, 2007), and the generateDS library, which allows the generation of Python data structures from an XML Schema Document (Kuhlman, 2004).

The *EDNA* kernel runs on all platforms which can provide a Python interpreter; the MX application prototype runs on all Unix/Linux platforms that have bash, Python (version 2.4 and later) and which can run the programs *MOSFLM*, *LABELIT*, *RADDOSE* and *BEST*. The *EDNA* framework and prototype are installed and regularly used at the Diamond Light Source (Didcot), EMBL Hamburg, ESRF Grenoble and NSLS Brookhaven National Laboratory (New York).

The Subversion system (Subversion Corporation, http://subversion.tigris.org/) is used for version control and software distribution. Bugzilla (Mozilla Organization, http://www.mozilla.org/) is used for managing feature requests and bug reports. Eclipse (Eclipse Foundation, http://www.eclipse.org) is used as the development environment. Enterprise Architect is used for the data model design (Sparx Systems, http://www.sparxsystems.com/). Additional information about the software may be found at the *EDNA* web site (http://www.edna-site.org/).

6. Conclusions and future plans

Setting up a framework presents many advantages. Sharing the same technical platform facilitates collaboration, especially for geographically remote sites. Furthermore, it allows the automation of the major part of the development process, resulting in potential gains in both productivity and quality.

The *EDNA* framework provides the possibility to build extensible scientific applications. Thanks to this feature, it has been possible to create a first prototype for macromolecular crystallography experiments to calculate strategies for data collection taking radiation damage into account. In order to allow the execution of one or several third-party programs in parallel for a given step, a generic data model is required and has been included, which in turn relies on the adoption of a standard data format by the different synchrotron facilities that are involved in the project. The first tests successfully carried out and the generic approach of the *EDNA* framework allowed a growing community of scientific institutes to show a great interest. A dedicated application (MXv1), based on the prototype developments, aiming to characterize MX crystals, has been released very recently. A short-term plan is to replace the data collection strategy application DNA (Leslie *et al.*, 2002) with an *EDNA* characterization application. Plans in the short and medium term will be focused on the development of new features including the processing of the diffraction data as soon as they are collected and single-axis kappa experiments. Longer term plans are to extend the generic data model to the description of the sample geometry (*e.g.* to allow more complex kappa experiments) and detector geometry. Integration of the new features provided by the *EDNA* MX application prototype within the beamline GUIs, MXCube (Leonard *et al.*, 2007) and GDA (Gibbons & Pulford, 2006), are also planned in order to take full advantage of the automated pipeline and the online graphical presentation of the data. Finally, the implementation of applications for fields other than macromolecular crystallography, in particular X-ray tomography, are also planned.

The authors would like to thank the *EDNA* executive committee and management team (Alun Ashton, Gérard Bricogne, Andrew Leslie, Andrew McCarthy, Sean

McSweeney, Thomas Schneider, Andrew Thompson) and all the *EDNA* project members for their support and efforts with this project. In addition, we would like to thank all the partners and users at Diamond Light Source (Didcot), ESRF (Grenoble), European Molecular Biology Laboratory (Grenoble, Hamburg), Global Phasing (Cambridge), MaxLab (Lund), MRC LMB (Cambridge), NSLS Brookhaven National Laboratory (New York), SLS (Villigen), Synchrotron Soleil (Paris) who gave their feedback on the functionality of the software. We especially acknowledge Ana-Maria Goncalves for providing the SurE crystals, and Elspeth Gordon for her help during experiments and fruitful discussions. We also wish to acknowledge the BIOXHIT foundation for the financial support of this initiative (grant number LSHG-CT-2003-503420).

References

Arndt, U. W. & Wonnacott, A. J. (1977). *The Rotation Method in Crystallography*. Amsterdam: North Holland.
Beteva, A. *et al.* (2006). *Acta Cryst.* D**62**, 1162–1169,
Bourenkov, G. P., Bogomolov, B. A. & Popov, A. N. (2006). *Fourth International Workshop on X-ray Damage to Biological Crystalline Samples*, Spring-8, Japan.
Bourenkov, G. P. & Popov, A. N. (2006). *Acta Cryst.* D**62**, 58–64.
Chaize, J. M., Gotz, A., Klotz, W. D., Meyer, J., Perez, M. & Taurel, E. (1999). *Internal Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, Italy.
Collaborative Computational Project, Number 4 (1994). *Acta Cryst.* D**50**, 760–763.
Dauter, Z. (1999). *Acta Cryst.* D**55**, 1703–1717.
Evans, P. R. (1999). *Acta Cryst.* D**55**, 1771–1772.
Evans, P. (2006). *Acta Cryst.* D**62**, 72–82.
Gibbons, P. & Pulford, W. C. (2006). *Diamond Light Source*, http://www.diamond.ac.uk/.
Gonçalves, A. M. D., Rêgo, A. T., Thomaz, M., Enguita, F. J. & Carrondo, M. A. (2008). *Acta Cryst.* F**64**, 213–216.
Gotz, A. (2007). *FABLE*, http://fable.wiki.sourceforge.net/.
Guijarro, M. (2004). *The BLISS Project*, http://lns00.psi.ch/nobugs2004/papers/paper00065.pdf.
Kabsch, W. (1993). *J. Appl. Cryst.* **26**, 795–800.
Konarev, P. V., Petoukhov, M. V., Volkov, V. V. & Svergun, D. I. (2006). *J. Appl. Cryst.* **39**, 277–286.
Korbas, M., Fulla Marsa, D. & Meyer-Klaucke, W. (2006). *Rev. Sci. Instrum.* **77**, 063105.
Kuhlman, D. (2004). *generateDS – Generate Data Structures from XML Schema*, http://www.rexx.com/~dkuhlman/generateDS.html.
Lamzin, V. & Perrakis, A. (2000). *Nat. Struct. Biol.* pp. 978–981.
Leonard, G. A., McCarthy, J., Nurizzo, D. & Thibault, X. (2007). *Synchrotron Radiat. News*, **20**, 18–24.
Leonard, G., Solé, V. A., Beteva, A., Gabadinho, J., Guijarro, M., McCarthy, J., Marrocchelli, D., Nurizzo, D., McSweeney, S. & Mueller-Dieckmann, C. (2009). *J. Appl. Cryst.* **42**, 333–335.
Leslie, A. G. W. (1992). *Jnt CCP4/ESF-EACMB Newsl. Protein Crystallogr.* **26**.
Leslie, A. G. W., Powell, H. R., Winter, G., Svensson, O., Spruce, D., McSweeney, S., Love, D., Kinder, S., Duke, E. & Nave, C. (2002). *Acta Cryst.* D**58**, 1924–1928.
Matthews, B. (1968). *J. Mol. Biol.* **33**, 491–497.
Minor, W., Cymborowski, M., Otwinowski, Z. & Chruszcz, M. (2006). *Acta Cryst.* D**62**, 859–866.
Murray, J. W., Garman, E. F. & Ravelli, R. B. G. (2004). *J. Appl. Cryst.* **37**, 513–522.
Murray, J. W., Rudiño-Piñera, E., Owen, R. L., Grininger, M., Ravelli, R. B. G. & Garman, E. F. (2005). *J. Synchrotron Rad.* **12**, 268–275.
Otwinowski, Z. & Minor, W. (1997). *Methods Enzymol.* **276**, 307–326.
Owen, R. A., Rudino-Pinera, E. & Garman, E. F. (2006). *Proc. Natl. Acad. Sci. USA*, **103**, 4912–4917.
Pflugrath, J. W. (1997). *Methods Enzymol.* **276**, 286–306.
Pieritz, R. A. (2007). *AALib – Asynchronous Action Library Project*, http://aalib.sourceforge.net/.
Popov, A. N. & Bourenkov, G. P. (2003). *Acta Cryst.* D**59**, 1145–1153.
Ravelli, R. & Garman, E. F. (2006). *Curr. Opin. Struct. Biol.* **16**, 624–629.
Sanchez del Rio, M. (1997). *J. Phys. IV*, C**2**, 209–210.
Sauter, N. K., Grosse-Kunstleve, R. W. & Adams, P. D. (2004). *J. Appl. Cryst.* **37**, 399–409.
Solé, V. A., Papillon, E., Cotte, M., Walter, Ph. & Susini, J. (2007). *Spectrochim. Acta*, B**62**, 63–68.
Soltis, S. M. *et al.* (2008). *Acta Cryst.* D**64**, 1210–1221.
Svensson O. *et al.* In preparation
Yamada, Y., pHonda, N., Matsugaki, N., Igarashi, N., Hiraki, M. & Wakatsuki, S. (2008). *J. Synchrotron Rad.* **15**, 296–299.