OPEN SOURCE CONTRIBUTIONS AND USING OSGI BUNDLES AT DIAMOND LIGHT SOURCE

M. Gerring, A. Ashton, R. Walton, Diamond Light Source, Oxfordshire, UK

Abstract

This paper presents the involvement of Diamond Light Source (DLS) with the open source community, the Eclipse Science Working Group and how DLS is changing to share software development effort better between groups. The paper explains moving from product-based to bundle-based software development process which lowers reinvention, increases reuse and reduces software development and support costs. This paper details specific ways in which DLS are engaging with the open source community and changing the way that research institutions deliver open source code.

INTRODUCTION

Diamond Light Source [1] is a third-generation 3 GeV synchrotron light source based on a 24-cell double-bend achromatic lattice of 561m circumference. The photon output is optimised for high brightness from undulators and high flux from multi-pole wigglers. The accelerators and first phase of seven photon beamline were constructed from 2002 to 2007; a second phase of fifteen photon beamlines from 2006 to 2012; and a third phase of ten photon beamlines was approved in 2011 with construction due to finish in 2017-8.

As well as the construction of the synchrotron, the early phases of the project saw choices about the software which would be deployed on site. For hardware control such as motors, the EPICS framework was chosen which included a data driven user interface called EDM [2] for configuring devices. The acquisition and online data analysis system was developed from a product in operation at the SRS [3] called GDA [4] previously presented at ICALEPCS.

Diamond Light Source (DLS) have switched GDA (client) and a standalone analysis product called DAWN [5] to load using a system called OSGi (Open Service Gateway Initiative) [6]. In addition, the EPICS/EDM screens are planned to be phased out of active support [7] in line with the move to RHEL7. The next generation of software for controls and acquisition is based on Eclipse Rich Client Platform (RCP). This allows software products to be built from OSGi bundles and features to be developed which are interoperable between controls, acquisition and data analysis software. This paper details how DLS interacts with open source technology to deliver feature rich, interoperable and reusable software systems across groups and in the wider community.

OPEN SOURCE 'ECOSYSTEM'

Eclipse RCP [8] is a software technology which has been available for more than a decade. It is used to build user interface applications and OSGi servers. DLS are utilizing it as a platform to deliver native user interface clients. One of the first RCP clients in production for acquisition was on the B18 beamline, presented at ICALEPCS 2011 [9].

The Eclipse Foundation is also an open source publisher which verifies open source code for intellectual property (IP) and software license. Its rigorous IP process [10] renders code safe for commercial companies and institutions to reuse at reduced risk of litigation. It provides a rich open source feature set, similar to that on offer by the Apache Foundation. This 'ecosystem' of IP checked bundles has provided many useful features which DLS have been able to reuse within software products.

In 2014 DLS proposed an Eclipse project [11] to make aspects of its DAWN product open source and IP checked by the Eclipse Foundation. This project was granted and IP checking is active, nearing completion at the time of this conference.

How Open Source Works

The procedure in scientific institutions active in open source software release has often been to provide source using GNU Public License (GPL). Various mechanisms have been used to do this for example: zip file on an ftp site, by implementing a web site using a technology such as Redmine or by using an open repository site like github. These approaches are now considered unsafe however because the GPL can force institutions to release previously unready or non-public code. More importantly, the source code has often not been IP checked before it is released. Foundations such as Apache and Eclipse provide an IP checking service. The service gives confidence to the copyright holder that they have provided something for which they have a lower risk of litigation. The source code of the software is also more likely safe to be re-used. This promotes wider use and contributions from outside collaborators.

ECLIPSE SCIENCE WORKING GROUP

DLS, Oak Ridge National Laboratory [12] and IBM have formed an Eclipse Science Working Group (SWG) in conjunction with ten other members ranging from small contractors to large commercial companies. The SWG members have started projects such as Triquetrum, Chemclipse and Integrated Computational Environment (ICE), links to which can be found on the web site [13].

ISBN 978-3-95450-148-9

The SWG also propose projects such as Eclipse Rich Beans and Eclipse Advanced Visualization. The SWG are working together on interoperable features and to deliver novel software solutions. A key area identified to allow interoperability has been the description and mathematics of n-dimensional data. In order to address this, DLS have produced an API of two OSGi bundles. These bundles are similar in scope and functionality to numpy [14] for python. The SWG have identified data description important to making future software features interoperable/reusable and propose projects planned to reuse this work in other fields.

SOFTWARE IN BUNDLES

Traditionally software has been created using designs which are non-modular at runtime – all of the program has to be in memory for it to run. In the Java world, this resulted in long start up times because a large "classpath" of jar files (compiled code) had to be resolved while software started.

OSGi bundles make code modular and low dependency. They have a feature known as declarative services which export functionality to other bundles without making hard dependencies. This means that software does not have to fully be in memory when it is run and so has fast start up times. Without hard dependencies, the architecture can load features incrementally as they are required by the user. It also means that software is made formally modular making it easy to reuse within different products because its requirements for compilation are low and well defined. The declarative services approach results in lower software development and support costs.

DLS have divided features into OSGi bundles for many features, for example: data including nD mathematics, file loading including HDF5, plotting and visualization, user interface widgets and auto generation, communication and devices, interacting with remote data, hardware configuration and mathematical processing pipelines, see also Table 1. This approach is leading to a swap in orientation from developing products to developing bundles. Management and support engineers are then free to define products based on the needs of the synchrotron users, software developers can more easily work together on multiple products and features can be reused in multiple areas.

CASE STUDIES

Data Format

A first common problem encountered by institutions running large experimental physics control, is the format in which data is written. Then once data is read into memory from a given format, a second problem is that data needs to be described in a standard way. Various frameworks have existed which deal with these problems at different institutions and in different programming languages.

DLS software developers took the approach that the details of the file format and the way it is loaded should be hidden from the part of software where the data is used. An OSGi service was created for loading data which supports a wide range of data formats such as HDF5 incl. NeXus, CBF, ASCII formats and most standard image formats. However it is also extendible in the compiled product by users. This is available because an extension point in the RCP framework has been created whereby the application may be added to after compilation and deployment. This approach means that any data format can be supported, providing a reader for it may be written.

```
ILoaderService service = …  // OSGi
File file = new File(…);
IDataset d = service.getDataset(file, …)
```

Figure 1: Loading data by service.

These lines of code read one dataset from a file and return an object similar to a numpy array, called IDataset. This object is the key to the second problem. (A comparison of MATLAB, numpy and IDataset is available in the DAWNSci Project examples.) It means that any application, written in Java, can load and share the data of multiple formats and create tools which use the data. The mathematics of the tools can be reused because of the common data format. The user interface can also be reused if the chosen platform is SWT. In addition a non-RCP application is not precluded from using the data layer, including non-OSGi based frameworks. The feature is a standard Java 'jar' file.

Plotting

The ability to plot data and interact with visual tools has been designed as an OSGi service. For instance all the plot operations, generic tools like peak fitting or integration of different regions and science specific tools like diffraction experiment ring fitting. This plotting service allows a plotting system to be created and data to be plotted with the proviso it is described as an IDataset.

```
IPlottingService ser = …  // OSGi
IPlottingSystem ps = ser.createPlottingSystem()
ps.createPlotPart(…)
ITrace trace = ps.createImageTrace(…)
trace.setData(d)

// Other config like name, colour map etc. then
ps.addTrace(trace)
```

Figure 2: Plotting an image.

Visual Tools are available to the user once the plotting system has been created, automatically. The plotting system comes with a wide range of tools which are extendible, again via extension points. The data rank plotted is passed to an underlying tool system which determines the available tools for that rank, see Figure 3 and 4. Tools may be chained together.

ISBN 978-3-95450-148-9

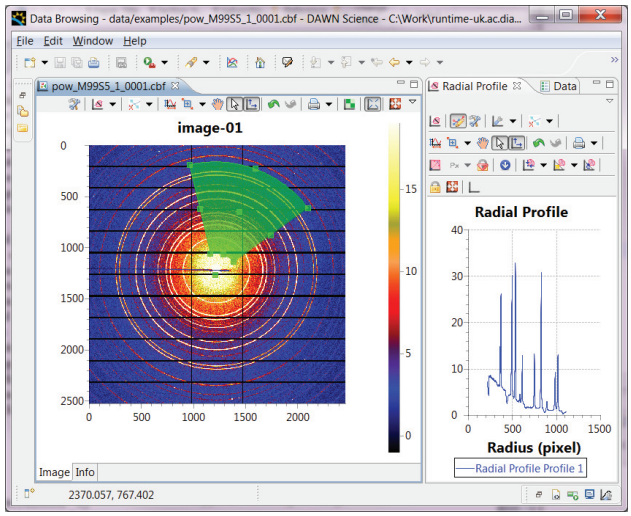

Figure 3: Using a single tool for radial profile

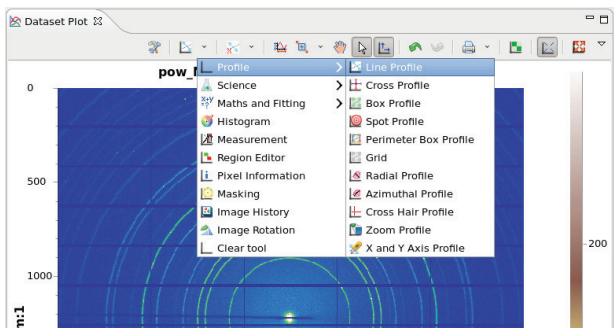

Figure 4 Some of the available tools for 2D data.

Regions of Interest are supported by the plotting system enabling a variety of shapes useful for data analysis to be drawn. The region system is integrated with the plotting system and uses a standard factory pattern (rather than extension points) to define available regions.

Multiple Viewers plug into the plotting system, contributable by extension point. For instance the system abstracts the plotting of lines from that of isosurfaces and uses different underlying user interface toolkits, Draw2D and JavaFX respectively, to render them.

Python Extensibility is supported by the plotting system. It is extensible for those that can write Java because of the hooks into it by extension point. However many users of the applications produced at DLS are not familiar with Java. Therefore the plotting system API is available to Python programmers via a plotting to python link. It is possible to either use simple factory methods for plotting numpy ndarrys or to get a reference to the actual plotting system Java object and make calls on to it, using a technology called py4j [14].

Streaming Data is supported by the plotting system. It can be connected to streams and data from remote file systems using a feature called Remote Dataset developed at DLS and deployed over several bundles. It is based on a servlets running in a Jetty server. In this case the data object on the client must implement an interface to mark it as dynamic and optionally remote. This dataset can be an MJPEG stream or a NeXus file on another file system, for instance. The plotting system will then update live feeds and the tool system is designed to work with live data. This works by using the Eclipse Job system to complete the tool mathematical operations in a separate thread and using a queue.

Reuse of the plotting system bundles are in the GDA client and DAWN products at DLS. It is also available in a product running at ISIS. It is being investigated for use with plotting of data from Control System Studio at DLS because of features like the tool system, python connectivity, streaming and ability to deal with many plot viewers.

Malcolm

DLS have a prototype project designed to abstract configuration and running of devices. The focus of the project, called 'Malcolm', is to deliver a way to configure and run Zebra devices [16]. It exposes a configurable device via a port to any client application via objects encoded into JSON strings. DLS have created a set of OSGi bundles for interacting with Malcolm devices which expose them as an OSGi service, see Fig 5.

```
IMalcolmService ms = …  // OSGi
IMalcolmConnection c = ms.createConnection(…)
IMalcolmDevice device= c.getDevice(…)
device.configure(…)
device.run()
```

Figure 5: A service to configure devices.

A **State Machine** has been designed to control Zebra but it also allows any hardware to be integrated. The well-defined states ensure that a programmer using the device has a clear idea of how to drive it, regardless of the details of the underlying hardware, see Fig 6.

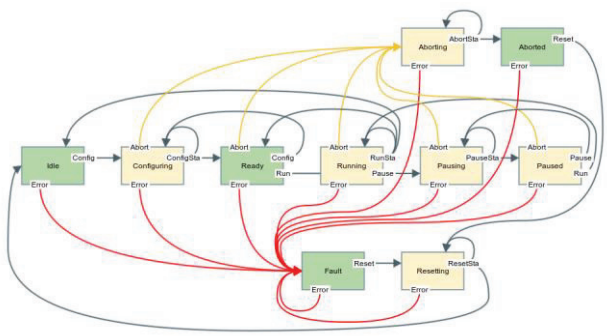

Figure 6: Design of Malcolm state machine.

The service is implemented using a connection to a python program which manages sending commands to the Zebra device and status to the user of the Malcolm state machine. This approach has allowed a software layer to sit close to the device and expose a well-defined way of using that device so reducing redevelopment of user interface and other features for similar devices.

ISBN 978-3-95450-148-9

Other Services

DLS would like to reuse more services between applications and have created more, which it hopes to reuse in the future between applications. Some of the more important are in Table 1.

Table 1: Some Useful Services Available from the Framework.

Service	Description
Conversion	Convert files from any format in which can be loaded to any which may be written.
Operation	Allows the programmer to create a processing pipeline from a large library of mathematical operations and run the pipeline over large image stacks on clusters.
Persistence	Save data, meta-data, masks and regions to a persisted NeXus file which can be loaded later and reimported.
Macro	A service which maps user interface actions with their python equivalent and allows the user interface to print macro commands into a running terminal while using the user interface.
Expression	A service to evaluate expressions. For instance the programmer may enter string expressions of datasets in expression language similar to python and evaluate them.

CONCLUSION

Diamond Light Source software developers have and continue to engage with the open source community. Best practice for IP checking code and releasing code is being adopted to ensure that it can be safely contributed for reuse and safely extended by external developers. New open source projects have been created which allow reusers of software produced at the synchrotron to take advantage of features developed. A new Eclipse Science Working Group has been formed where new ideas for projects and APIs can be discussed and the more interesting ones tried out. Many of the internal features developed at DLS are being released as bundles which contribute services. This is decoupling the software at DLS - making it more modular, easier to reuse features between applications and cheaper to support.

ACKNOWLEDGEMENT

The members of the Eclipse Science Working Group, large and small whom have helped create a positive environment for discussing and sharing code. Thanks to the many contributors to RCP products at DLS, especially DAWN and its external contributors.

REFERENCES

[1] R. P. Walker, “Commissioning and Status of The Diamond Storage Ring”, APAC 2007, Indore, India.

[2] John Sinclair, “EDM: Extensible Display Manager for EPICS”, USPAS 2003

[3] V.P. Suller, “Performance of the Daresbury SRS with an Increased Brilliance Optic”, EPAC 1988

[4] Generic Data Acquisition, www.opengda.org

[5] M. Basham, “Data Analysis WorkbeNch (DAWN)”, J. Synchrotron Rad. (2015). 22, 853-858

[6] O. Alliance, “Osgi service platform, release 3” IOS Press, Inc. 2003

[7] M. Furseman, “Adopting and Adapting Control System Studio at Diamond Light Source”, ICALEPCS 2015, Melbourne, Australia

[8] Eclipse - www.eclipse.org

[9] R. J. Woolliscroft, “Quick EXAFS Experiments Using a New GDA Eclipse RCP GUI with EPICS Hardware Control”, ICALEPCS 2011, Grenoble, France

[10] www.eclipse.org/projects/dev_process/ip-process-in-cartoons.php

[11] DAWNSci projects.eclipse.org/proposals/dawnsci

[12] ORNL www.ornl.gov/

[13] Eclipse Science Working Group - science.eclipse.org

[14] NUMPY www.numpy.org

[15] PY4J www.py4j.org

[16] T. Cobb, “Zebra: A Flexible Solution for Controlling Scanning Experiments”, ICALEPCS 2013, San Francisco, USA

ISBN 978-3-95450-148-9