OPEN SOURCE CONTRIBUTIONS AND USING OSGI BUNDLES AT DIAMOND LIGHT SOURCE

Matthew Gerring
Real Open Source

- Permissive license Apache / EPL
  - DLS still have many GPL codes → we are moving

- IP Checking by a Foundation like Apache or Eclipse
  - Community proposal
  - Consultation Period
  - Initial IP check
  - Public Repository Allocation
  - CLA for all committers and/or organization
Open Source Ecosystem

- Rich set of features
- Permissive licenses
- IP checked
- Wide user base
Eclipse Foundation – Science Working Group

Steering Committee

- diamond
- Oak Ridge National Laboratory
- IBM
- Otema

Participating

- Lab icate
- MARINTEK
- Clemson University
- Facility for Rare Isotope Beams
- Kichwa Coders
- Uppsala Universitet
- IFP Energies nouvelles
- iSenda
- Airbus
Software in Bundles

- OSGi manages dependencies
- Static types but *Dynamic* executable
- Declarative Services
- Code *more* modular
- Cheaper to own/support *large* projects
- Developing bundles / features *not* products
1. Data Format / Math

- IDataset ~numpy / ~MATLAB for Java
  - See NumpyExamples in DAWNSCI eclipse project
- Service that loads any data
  - Extension point available in the binary
  - NeXus / HDF5, CBF, Ascii, scores of others
- Part of DAWNSci Eclipse Project! (IP checking)

```java
ILoaderService service = ... // OSGi
File file = new File(...);
IDataset d = service.getDataset(file, ...)
```
2. Plotting with Tools

- IPlottingSystem, IPlottingService
  - 1D, 2D, 3D plotting
  - Many visual tools
  - Regions of Interest
  - Python connectivity

```java
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)
```
3. Malcolm

- **IMalcolmService**
  - Connect and drive *Zebra* devices
  - Connect any device and use standardized UI
  - State Machine Design

```java
IMalcolmService ms = ... // OSGi
IMalcolmConnection c = ms.createConnection(...)
IMalcolmDevice device = c.getDevice(...)
device.configure(...)
device.run()
```
Lots of Services

- Conversion
- Operation
- Persistence
- Macro
- Expression
- ... Around 40 others
Conclusion

- Permissive licenses
- (Joint the) Science Working Group!
- Services = modularity
- Services = reuse – do things once!
- Services = cheaper to own

Thanks to: Eclipse Foundation, SWG, DLS colleagues(!), Java Community, Apache Foundation, Kichwa Coders, Itema