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
<table>
<thead>
<tr>
<th>Steering Committee</th>
</tr>
</thead>
<tbody>
<tr>
<td>diamond</td>
</tr>
<tr>
<td>Oak Ridge National Laboratory</td>
</tr>
<tr>
<td>IBM</td>
</tr>
<tr>
<td>Otoma</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Participating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lablicate</td>
</tr>
<tr>
<td>MARINTEK</td>
</tr>
<tr>
<td>Clemson</td>
</tr>
<tr>
<td>Facility for Rare Isotope Beams</td>
</tr>
<tr>
<td>kichwa coders</td>
</tr>
<tr>
<td>Uppsala Universitet</td>
</tr>
<tr>
<td>ifp Energies nouvelles</td>
</tr>
<tr>
<td>iSencia</td>
</tr>
<tr>
<td>Airbus</td>
</tr>
</tbody>
</table>
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
Case Studies
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 check)

```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

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

// Other **config** like name, colour map etc. then
```python
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
- Science Working Group!
- Services = modularity
- Services = reuse – do things once!
- Services = cheaper

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