3MF:
EMF TO THE INFINITY... AND BEYOND

MARCH 27TH, 2012
MIKAEL BARBERO – OBE0
Multiple Ecore model versions in the same OSGi container

Wipe out dependency from Java interfaces to implementation

Make EMF a better OSGi citizen

Work inspired from Alex Blewit and Neil Bartlett following posts [1, 2] and bugs [3, 4]

(1) http://njbartlett.name/2011/02/07/emf-in-osgi.html
(3) https://bugs.eclipse.org/bugs/show_bug.cgi?id=329209
(4) https://bugs.eclipse.org/bugs/show_bug.cgi?id=328227
Possibility of backward incompatibility
USE CASE 1

ECORE MODEL V1

Tool Vendor 1

ECORE MODEL V2

Tool Vendor 2
WAIT, I CAN’T USE TOOLS FROM THESE TWO VENDORS AT THE SAME TIME?

I don’t care if they are based on some data model whose different versions can not live together.
FIRST GUILTY:
ECLIPSE EXTENSIONS
FIRST GUILTY: EXTENSIONS

- OSGi is modular and dynamic by nature.
- Extensions are reducing some of these advantages by forcing plug-in to be singleton.
- EMF uses extensions to contribute generated and dynamic EPackages.
FIRST SOLUTION

- Mimic `GeneratedPackageRegistryReader` and `DynamicPackageRegistryReader` during activation of model plug-ins.

- Advantages: simple and mostly backward compatible.

- Drawbacks: plug-in must be activated to contribute its model and some boilerplate code needs to be written.
SECOND SOLUTION

- Use OSGi (declarative) services.

- Advantages: simple and mostly backward compatible. Once defined as a service, any requiring class can be made a component requiring this service (DI).

- Drawbacks: needs a ServiceTracker to track services that can come and go anytime.

**EVEN BETTER OSGI R4.3: CREATE A CAPABILITY FOR EPACKAGES**
SECOND GUILTY

EPACKAGE.REGISTRY KEY

HTTP://WWW.FLICKR.COM/PHOTOS/FIDDEYDEE/4349596304/
EPackage instances are identified by namespace URI in the Registry.

Best practice (workaround?) today is to include version into namespace URI.

It should be a tuple (URI, Version)
SOLUTION

- Need a specific EPackage.Registry implementation that handle versions

```java
public interface VersionedRegistry extends EPackage.Registry {
    EPackage getEPackage(String nsURI, String version);
    EFactory getEFactory(String nsURI, String version);
}
```

- Contributed through `package_registry_implementation` extension point.

- Version can be valued from a property of service or from version of contributing bundle (or from capability?)
USE CASE 2

« If the **interfaces** can be kept "clean", i.e. **without any references to** EMF-generated **implementation** classes or to EMF itself, then the APIs can be used standalone, for example in a memory constrained environment. In this scenario, normal hand-coded implementation classes could be used. »

Neil Bartlett
https://bugs.eclipse.org/bugs/show_bug.cgi?id=329209
GUILTY:
EPACKAGE.EINSTANCE

HTTP://WWW.FLICKR.COM/PHOTOS/DCJOHN/12852539/
EMF

Interfaces

EOBJECT

EPACKAGE

EFACTORY

YOUR MODEL
 YOUR MODEL

Interfaces

EMF

EObject

EPackage

EFactory

YOUR MODEL

Foo

BarPackage

BarFactory

Impl

EMF

EObjectImpl

EPackageImpl

EFactoryImpl

YOUR MODEL

FooImpl

BarPackageImpl

BarFactoryImpl

Interfaces

Impl
BENEFITS

- Make generation gap pattern very natural and easy.

- Change from plain EMF objects to CDO native objects.
SOLUTION

- Add a method to (Internal)EObject

  ```
  abstract EPackage ePackage();
  ```

- Can be done in an extending interface implemented by only newly generated models
In all cases, it requires to give its defining EPackage to the EObject’s constructor, and then to modify XXXFactoryImpl.
SOLUTION

- Some adapter factories need to be modified too (AdapterFactory for models)
- Need to introduce a new interface extending AdapterFactory

```java
public interface ModelAdapterFactory<P extends EPackage, F extends EFactory>
    extends AdapterFactory {
    P getEPackage();
    F getEFactory();
}
```
SOLUTION

```java
public interface ModelAdapterFactory<P extends EPackage, F extends EFactory>
    extends AdapterFactory {
    P getEPackage();
    F getEFactory();
}
```

XXXItemProviderAdapterFactory is also implementing it

XXXItemProviders are taking this interface as argument to their constructor instead of simple AdapterFactory.
BIND IT ALL

This solution perfectly fits with registration of EPackages as OSGi (declarative) services
USE CASE 3

« I want to use EMF models on another OSGi runtime than Equinox »
<table>
<thead>
<tr>
<th>SUPPORTED PLATFORMS</th>
<th>RAP AND GWT SINCE EMF 2.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECLIPSE WORKBENCH</td>
<td>IDE</td>
</tr>
<tr>
<td>RICH CLIENT PLATFORM</td>
<td>RCP</td>
</tr>
<tr>
<td>ECLIPSE HEADLESS</td>
<td>EQUINOX</td>
</tr>
<tr>
<td>STANDALONE</td>
<td>JAVA RUNTIME</td>
</tr>
<tr>
<td>ECLIPSE RICH AJAX PLATFORM</td>
<td>RAP</td>
</tr>
<tr>
<td>GOOGLE WEB TOOLKIT</td>
<td>GWT</td>
</tr>
</tbody>
</table>
What about other OSGi runtimes?

When running in an OSGi container, EMF explicitly requires the following 8 bundles:

- org.eclipse.core.runtime
- org.eclipse.equinox.common
- org.eclipse.core.jobs
- org.eclipse.equinox.registry
- org.eclipse.equinox.preferences
- org.eclipse.core.contenttype
- org.eclipse.equinox.app
- org.eclipse.osgi
SOLUTION 1 & 2

- (Repackage EMF bundles)
- Big refactoring to separate Eclipse/Equinox related bits from OSGi related bits
- Use import-package instead or require-bundle
SOLUTION 3

- There are some less radical changes that can be done in a backward compatible fashion, but
- Introduces even more graceful handling of target running platform into the runtime
- Ex: rewriting EMFPlugin.IS_ECLIPSE_RUNNING in a similar way to IS_RESOURCE_BUNDLE_AVAILABLE in the term of a new EMFPlugin.IS_OSGI_RUNNING.
CONCLUSION

- EMF is a great piece of engineering
- OSGi is a great piece of engineering too
- 3MF is an experimentation to make EMF mimic / use some of the best qualities of OSGi (modularity, dynamicity)
- With these, EMF would reach the infinity... and beyond!
CONCLUSION

- EMF is a great piece of engineering
- OSGi is a great piece of engineering too
- 3MF is an experimentation to make EMF mimic / use some of the best qualities of OSGi (modularity, dynamicity)
- With these, EMF would reach the infinity... and beyond!
HTTPS://GITHUB.COM/MBARBERO/EMF/TREE/3MF
HTTP://WWW.SLIDESHARE.NET/MIKAELBARBERO/