Creating Platforms Using Eclipse Equinox

Jeff McAffer
Equinox, RCP, RT co-lead
CTO, EclipseSource
Modern application development and deployment

<table>
<thead>
<tr>
<th>Current Issues</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component models vary across tiers and platforms</td>
<td>• MS .Net == MS platforms</td>
</tr>
<tr>
<td></td>
<td>• Java ME, SE and Java EE imply different component models on embedded devices, desktops and server</td>
</tr>
<tr>
<td>New type of applications</td>
<td>• SaaS, web 2.0, mashups and social networks require new approaches</td>
</tr>
<tr>
<td>Business Agility Demands IT Agility</td>
<td>• One size does not fit all</td>
</tr>
<tr>
<td>Improve integration of 3rd party software</td>
<td>• Different technology platforms make it difficult to integrate with customers and partners</td>
</tr>
</tbody>
</table>
Platforms
Enabling the Unexpected
The Evolution of Eclipse Platforms

Value for the Eclipse Ecosystem

Tooling Platform

Rich Client Platform

Runtime Platform

Today
NASA Maestro/Ensemble
On the power of adopting a platform

“Since adopting the [platform], our team has been able to retire thousands of lines of code from our old program in favor of features provided by the [platform].”
On the power of building your own platform

“[The platform] has also become the centerpiece for a new consortium of operations software development teams”

“…we will see more [platforms] built around the [our platform] in the future as other organizations decide to pool their resources and share the responsibility for things that their programs have in common.”
Even in Banking!

“…they needed to produce a generalized platform in which this and other new applications could be extended, reused and shared”

“The result was [JPMorgan’s] One Bench, a platform for developing and delivering custom banking applications”
Components
On the power of componentization

“… now we can safely pick and choose only those plug-ins that a particular customer needs rather than forcing everyone to use a monolithic “one size fits all” program”

“Death of the Stack”
“[Birth of] Stackless Stacks”
Equinox powers Componentization

• Equinox powers Eclipse
• Small, performant OSGi framework implementation
• Collection of service implementations
  • Standard OSGi services
  • Eclipse services (e.g., Extension Registry)
• Server side infrastructure
• Provisioning infrastructure
• Security infrastructure

Runtime Community at Eclipse.org
A brief history of Equinox

• Introduced in Eclipse 3.0 (2003) as OSGi-based runtime for Eclipse
• Seeded with IBM “SMF” code-base
• Team co-developed OSGi R4 spec
  ♦ Facilitate Eclipse use
  ♦ Reference implementation for OSGi R4.x + JSR 291
• Widespread adoption as the complete, supported, industrial strength framework implementation
System View

Equinox

Java

Your s
3rd Party
Eclipse

symbian
Mac
solaris
Write Once, Run Anywhere?

• Perhaps true across machines
• Ironically not true across Java™ “editions”
• Java ME, SE and EE have different programming models
  • Midlets, Applets, Servlets, EJBs, …
• The same program does not have a hope of running everywhere
The OSGi Component Model

• Bundles are typically JAR files
  - Java classes, Resources, Files, Metadata
• Bundle metadata declaratively defines
  - Java packages exported
  - Dependencies on bundles and Java packages
  - Bundle classpath
  - Bundle lifecycle
• Framework manages dependencies and lifecycle
  - Explicitly supports dynamic scenarios
Example Bundle Metadata

Bundle-SymbolicName: org.eclipse.equinox.registry
Bundle-Version: 3.2.100.v20060918
Bundle-Name: Eclipse Extension Registry
Bundle-Vendor: Eclipse.org

Bundle-ClassPath: .
Bundle-Activator: org.eclipse.core.internal.registry.osgi.Activator

Import-Package: javax.xml.parsers,
org.xml.sax,
org.osgi.framework;version=1.3
Require-Bundle:
org.eclipse.equinox.common;bundle-version="[3.2.0,4.0.0)"
Bundle-RequiredExecutionEnvironment: J2SE-1.3

Export-Package: org.eclipse.equinox.registry
Enterprise Java Dominance

Solo

Eclipse

Java

Equinox

HTTP (Jetty)

JSPs

Servlets

Your

3rd Party

Eclipse

symbian

Mac

solaris
Embedded

- JSPs
- Thin Equinox WAR
- Lite HTTP Service
- Equinox

Application Server

- Eclipse
- 3rd Party
- Your Servlets

Platforms: Symbian, Windows, Mac, Solaris
Demo
But I thought it was for cars?
A Look Under the Covers
Separation of Concerns

**Component.java**

```java
protected void activate(ComponentContext context) {
    this.context = context;
    HttpService http = (HttpService) context.locateService("http");
    IControlCenter center = (IControlCenter) context.locateService("controlCenter");
    HttpServlet servlet = new EmergencyServlet(center);
    http.registerServlet(getAlias(), servlet, null, httpContext);
}

protected void deactivate(ComponentContext context) {
    HttpService http = (HttpService) context.locateService("http");
    http.unregister(getAlias());
}
```

**EmergencyServlet.java**

```java
public EmergencyServlet(IControlCenter center) {
    this.center = center;
}
```

< … Business Logic …>
Services Examples

- swt.emergency
- client.emergency
- dev.gps.fake
  - Business Logic
  - Device Driver
Providing a Service

MANIFEST.MF
Bundle-Name: Toast Fake Gps Plug-in
Bundle-SymbolicName: org.equinoxosgi.toast.dev.gps.fake
Bundle-Version: 1.0.0
Service-Component: OSGI-INF/component.xml
Import-Package: org.equinoxosgi.toast.dev.gps
Providing a Service

```xml
<component name="org.equinoxosgi.toast.dev.gps.fake">
  <implementation class="org.equinoxosgi.toast.dev.gps.fake.internal.FakeGps"/>
  <service>
    <provide interface="org.equinoxosgi.toast.dev.gps.IGps"/>
  </service>
</component>
```
Providing a Service

FakeGps.java
public class FakeGps implements IGps {
    public int getLatitude() {
        return 3888746;
    }

    public int getLongitude() {
        return -7702192;
    }
}
Requiring a Service

**component.xml**

```xml
<component name="org.equinoxosgi.toast.swt.emergency">
  <implementation class="org.equinoxosgi.toast.swt.emergency.internal.bundle.Component"/>
  <reference name="emergency" interface="org.equinoxosgi.toast.client.emergency.IEmergencyMonitor"/>
  <reference name="shell" interface="org.equinoxosgi.crust.shell.ICrustShell"/>
</component>
```
Requiring a Service

**Component.java**

```java
public class Component {
    private EmergencyScreen screen;

    protected void activate(ComponentContext context) {
        ICrustShell crustShell = (ICrustShell) context.locateService("shell");
        IEmergencyMonitor monitor = (IEmergencyMonitor) context.locateService("emergency");
        screen = new EmergencyScreen();
        screen.bind(crustShell, monitor);
    }

    protected void deactivate(ComponentContext context) {
        screen.unbind();
        screen = null;
    }
}
```
Requiring a Service

EmergencyScreen.java
public class EmergencyScreen {
    private IEmergencyMonitor monitor;
    private ICrustShell crustShell;

    public void bind(ICrustShell crustShell, IEmergencyMonitor monitor) {
        this.crustShell = crustShell;
        this.monitor = monitor;
        crustShell.installScreen(...);
        monitor.addListener(this);
    }

    public void unbind() {
        monitor.removeListener(this);
        crustShell.uninstallScreen(...);
    }

    < ... Business logic ... >
Requiring & Providing a Service

**component.xml**

```xml
<component name="org.equinoxosgi.toast.client.emergency">
    <implementation class="org.equinoxosgi.toast.client.emergency.internal.bundle.Component"/>
    <service>
        <provide interface="org.equinoxosgi.toast.client.emergency.IEmergencyMonitor"/>
    </service>
    <reference name="gps" interface="org.equinoxosgi.toast.dev.gps.IGps"/>
    <reference name="airbag" interface="org.equinoxosgi.toast.dev.airbag.IAirbag"/>
</component>
```
Requiring & Providing a Service

**Component.java**

```java
public class Component implements IEmergencyMonitor {
    private EmergencyMonitor monitor;
    protected void activate(ComponentContext context) {
        IGps gps = (IGps) context.locateService("gps");
        IAirbag airbag = (IAirbag) context.locateService("airbag");
        monitor = new EmergencyMonitor();
        monitor.bind(gps, airbag);
    }
    protected void deactivate(ComponentContext context) {
        monitor.unbind();
    }
    public void emergency() {
        monitor.emergency();
    }
    public void addListener(IEmergencyMonitorListener listener) {
        monitor.addListener(listener);
    }
    public void removeListener(IEmergencyMonitorListener listener) {
        monitor.removeListener(listener);
    }
}
```
Requiring & Providing a Service

**EmergencyMonitor.java**

```java
public class EmergencyMonitor implements IAirbagListener, IEmergencyMonitor {
    private IGps gps;
    private IAirbag airbag;

    public void setGps(IGps gps) {
        this.gps = gps;
    }

    public void setAirebag(IAirbag airbag) {
        this.airbag = airbag;
    }

    public void activate() {
        airbag.addListener(this);
    }

    public void deactivate() {
        airbag.removeListener(this);
    }

    < ... Business logic ... >
```
Broader Implications
Component Oriented Development and Assembly

Component Producers | Customize Components | Individual Solutions

Create | Extend | Assemble

Equinox

symbian | Mac | solaris
Wrap-up

• Equinox is a platform for building platforms
• Platforms
  ✷ Promote innovation that matters to you
  ✷ Leave the “gorp” to others
• Equinox in the runtime space is real

• Stop coercing monolithic of-the-shelf stacks
• Start designing and assembling stacks just for you