Paris, France

OSGi Alliance

2005 Developer Forum & World Congress
Using R4 in Vehicle Embedded Systems

Siemens VDO Automotive

Paris, France

Miguel Lopez, Software Architect
Agenda

- Subject
  - why we decided to use R4 in vehicle system

- Introduction
  - Release 4 is a major step

- How to use R4 in vehicle embedded systems?
  - Illustrated with concrete use cases

- Conclusion
Who Are We?

- Siemens VDO & OSGi Alliance
  - Siemens VDO is involved in OSGi Alliance since 2001
- is the Siemens VDO Platform
  - multimedia system based on OSGi Service Platform
  - used in BMW 1, 3, 5 and 6 Series, X3
  - used in current developments for other Car makers
R4 Is a Major Release

- Important milestones achieved in OSGi Alliance
  - R4 Core (including Compendium) Specifications - released
  - R4 Vehicle and Mobile Specifications – soon
- What are the changes/improvements?
  - Comparison between R3 and R4: two metrics
    - Specification (number of pages)
    - API Changes (number of methods)
R4 Has Been Significantly Increased

- Specification Documents (number of pages)
  - twice as many pages in R4 than R3

- API Changes (number of public methods)
  - twice as many public methods in R4 than R3
R4 Is Adapted and Suitable

• Is it suitable for vehicle system?
  – OSGi Service Platform originally targeted "small devices"

• Now, R4 is applicable in many architectures
  – add powerful new modularity capabilities to Java
  – improved encapsulation of software units
  – R4 plans to address a large device range

• However, R4 is well adapted and suitable for vehicle systems
  – Lets have a look at the use cases
Use Cases

- Gradually Integrate new features
- **Localize applications**
- **Enhance the development of applications**
- **Help inter-bundle communication**
- Manage power changes
- Diagnose the system
Gradually Integrate New Features

- Systems are rapidly becoming complex as their capacities increase.
- Critical step in the software process is integration:
  - Integrate all modules/components/features within a single platform and a given product.
  - Handle complex dependencies.
- Even worse... some are conflicting:
  - Let look at a concrete example.
Gradually Integrate New Features

- No way with R3. What about R4?
Gradually Integrate New Features With R4

Core Framework Modularity

- Manages multi versions of the same package
- Both modules can then coexist
- Advantages
  - Integrators have more flexibility
  - Developers team can defer new baseline integration
Localize Applications

• What is localization?
  – adapting an application to various languages, cultures and environments
    • textual elements are not hard coded
    • support for new languages without recompilation
    • culturally dependent format for dates, currencies ...

• Java SE Platform already provides such features
  – java.util.Locale, java.util.ResourceBundle, etc ...

• Java SE Localization must be adapted to OSGi
  – Both use Java class-loading capabilities for searching resources, classes
Localize Applications With R4

Fragment Bundle (Modularity Layer)

- Fits Java SE Localization concepts
- Ease to understand for Java Developers
- Very Flexible
  - New languages may be added after deployment
  - Good migration of legacy Java Applications
Enhance Development of Applications

- OSGi technology defines a powerful service oriented architecture ... However
  - Number of services grows quickly
- Complex inter-service dependencies
  - Due to dynamic nature of OSGi environment
  - Deal with specific OSGi APIs
- Developer may not prepare to deal with this model
- It seems reasonable to solve in a single place all these issues
META-INF/Manifest.mf
Bundle-Activator: com.osgicongress.Hello

com.osgicongress.Hello.java
public class Hello implements BundleActivator
{
    public void start(BundleContext bc)
    {
        ServiceTracker tracker = new ServiceTracker(bc, "org.osgi.service.log.LogService", null);
        tracker.open();
        LogService log = (LogService)tracker.getService();
        if (log != null)
        {
            log.log(LogService.LOG_INFO,"Hello OSGi Congress");
        }
        else
        {
            // what to do here ...
        }
    }

    public void stop(BundleContext bc) {}
}
Declarative Services

- Simplifies the service oriented programming model (XML description)
- Assists bundle developers in their work
- Handles the dynamic of the service objects
# Service Programming Model – R3 vs. R4

## R3 Programming Model

### META-INF/Manifest.mf

Bundle-Activator: com.osgi.congress.Hello

```java
com.osgi.congress.Hello.java
public class Hello implements BundleActivator {
    public void start(BundleContext bc) {
        ServiceTracker tracker = new ServiceTracker(bc, "org.osgi.service.log.LogService", null);
        tracker.open();
        LogService log = (LogService) tracker.getService();
        if (log != null) {
            log.log(LogService.LOG_INFO, "Hello OSGi Congress");
        } else {
            // ??? what to do here
        }
    }

    public void stop(BundleContext bc) {}
}
```

## R4 Programming Model

### META-INF/Manifest.mf

Service-Component: OSGI-INF/activator.xml

```xml
OSGI-INF/activator.xml
<?xml version ="1.0" encoding="UTF-8"?>
<component name="example.Hello">
    <implementation class="com.osgi.congress.Hello"/>
    <reference name="LOG" interface="org.osgi.service.log.LogService"/>
</component>
```

```java
com.osgi.congress.Hello.java
public class Hello {
    protected void activate(ComponentContext cc) {
        LogService log = (LogService) cc.locateService("LOG");
        log.log(LogService.LOG_INFO, "Hello OSGi Congress");
    }
}
```
Enhance Development of Applications

• Main Advantages
  – Better developer productivity
  – Less error prone for developers
  – Developers focus on business logic
  – May reduce memory footprint (lazy activation)
    • Creation and activation of the component can be delayed
Communication between bundles is essential
Almost, all bundles and applications must react on signals
Several designs may be used to implement such communication
  – Popular Java Event/Listener pattern (Java Bean Model)
  – OSGi whiteboard approach
Open Issues:
  – Minimize proliferation of Event/Listener interfaces
  – Is there a way to find and promote a generic mechanism?
Generic Event Mechanism With R4

Event Admin Service

- Based on Publish-Subscribe pattern
- Easy system monitoring
- Core events: Framework/Service/Bundle/Log
- Vehicle events: Power Management
- Any event may be published through this simple and flexible API
Manage Power Changes

- Systems must carefully manage power consumption
- Power Information from user, application, or hardware device must be collected to enable a better power management decision and execution
- Applications must adjust their activities
- Some hardware devices may have their own power management
- Open Issues:
  - How to organize and control the whole system to enable a better power decision and execution?
  - No standard model
Power Management Service in Vehicle Specification

**Power Manager Service**

- Provides a common approach for managing the platform's energy consumption
  - Coordinates and consistently maintains the system and device power states
  - Defines the power state transitions
  - System and device power states notifications
  - Policy free in order to adapt as efficient as possible the power strategy
Diagnose System

- Need to perform diagnostics
  - Configure, monitor and verify the system

- Diagnostician requirements are:
  - Get device parameters
  - Change device parameters
  - Execute self-tests
  - Monitor device data

- Open Issues:
  - Is there a common approach for discovery and invocation of diagnostic commands
  - Standard approach is required
Diagnostic Service

- Full description of diagnostic commands using Meta typing (e.g. XML file)
- Easy to publish diagnostic commands for newly installed devices
- Local and remote invocations
Conclusion

- R4 platform brings real added value in vehicle systems
  - Core enhancements
  - New vehicle services
- R4 solves significant vehicle platform use cases
- TLA is an excellent platform for using R4 Specifications in the next coming years
SAN DIEGO Is Our Reference Application

- Visit our booth!
Questions?

Miguel Lopez
Software Architect
Siemens VDO Automotive
Sophia-Antipolis, France
mailto: miguel.lopez@siemens.com
+33 (0) 492 381 151