OSGi

Deepak Dhungana
dhungana@ase.jku.at
Institute for System Engineering and Automation

Thomas Wuerthinger
wuerthinger@ssw.jku.at
Institute for System Software

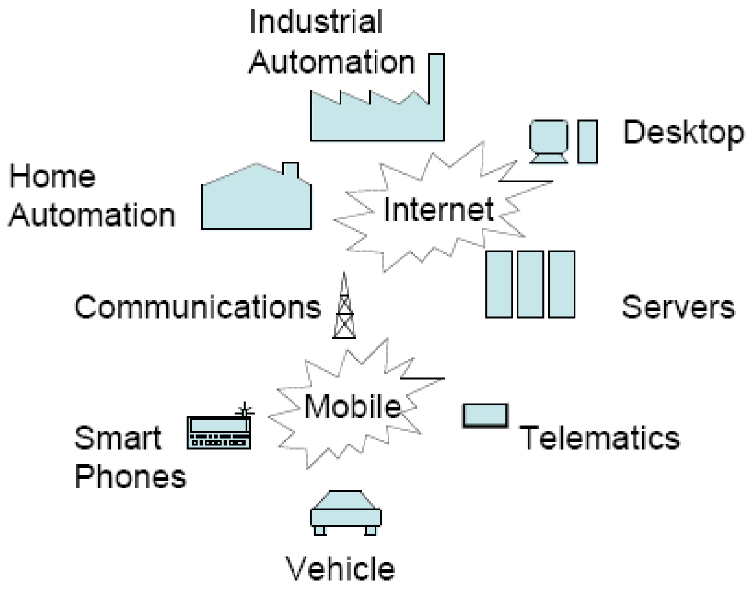
Johannes Kepler University Linz, Austria
http://www.jku.at

Original slides by
Thomas Watson
and Peter Kriens

eclipseCON 2006

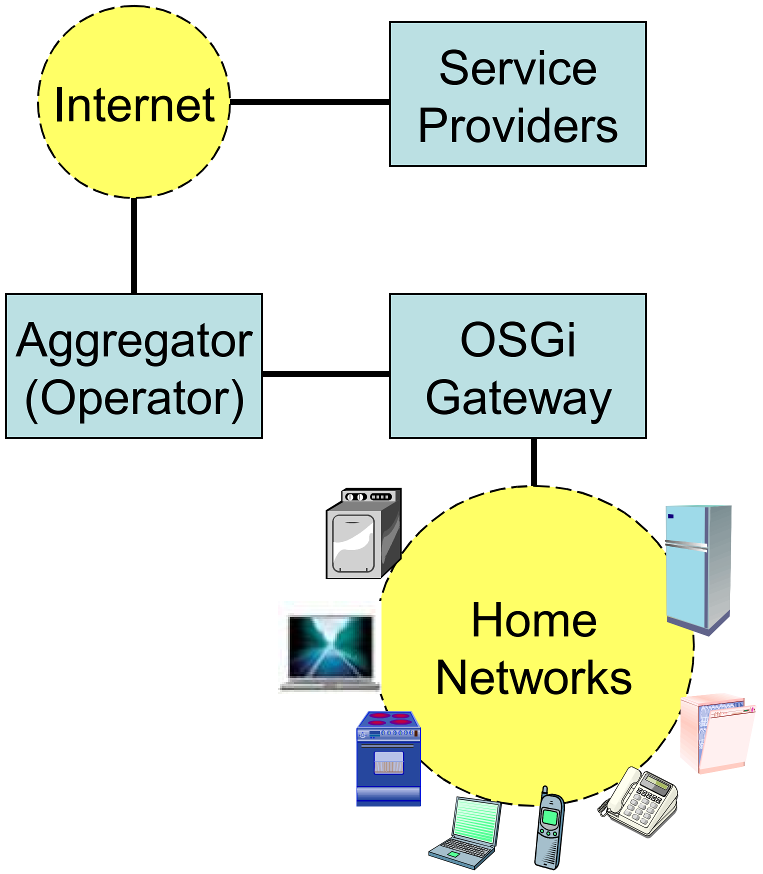
Open Service Gateway Initiative

- Open: Open and Dynamic Platform
- Service: Run-time environment for services and applications
- Gateway: gateway for connecting local devices and networks
- Where is OSGi used?
  - Automotive
  - Smart Home
  - Mobile Phones
  - Facility Management
  - Consumer Electronics
  - Health Care
  - Industry Automation
  - …



Home Automation

- The original use case for OSGi
- OSGi residential gateway controls a number of local networks
- An operator aggregates functions provided by external parties and manages them on the gateway
- The residential user is billed from one point



Vehicles: BMW

New Business Cases:

New features added, new software downloads, change the HMI

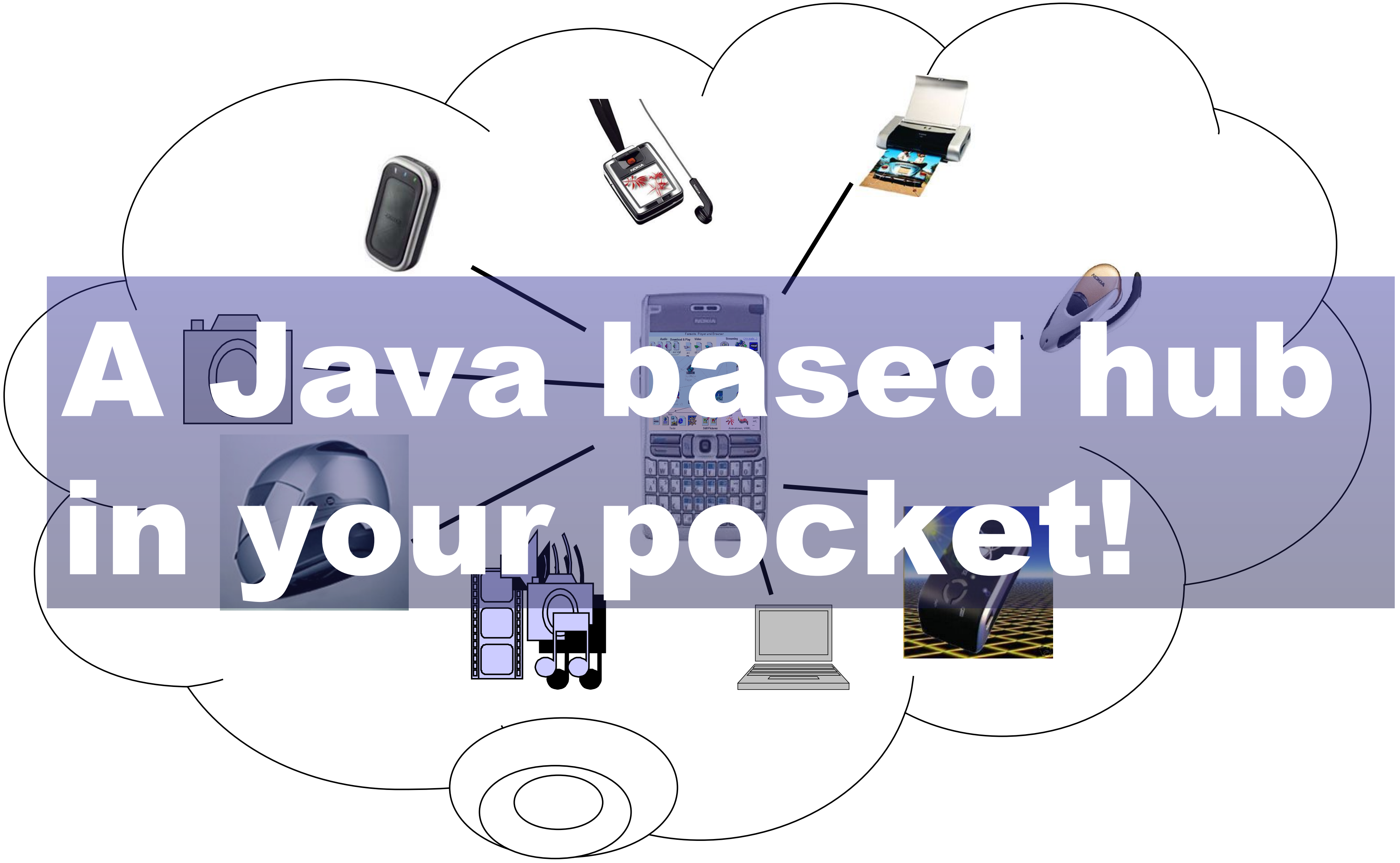
OSGi Service Platform

OSGi Technology inside: adoption for different makes and models

Standard version

- Air conditioning
- GPS Navigation
- Multimedia (CD, tuner…)
- Remote diagnosis
- Address book
- Speech recognition

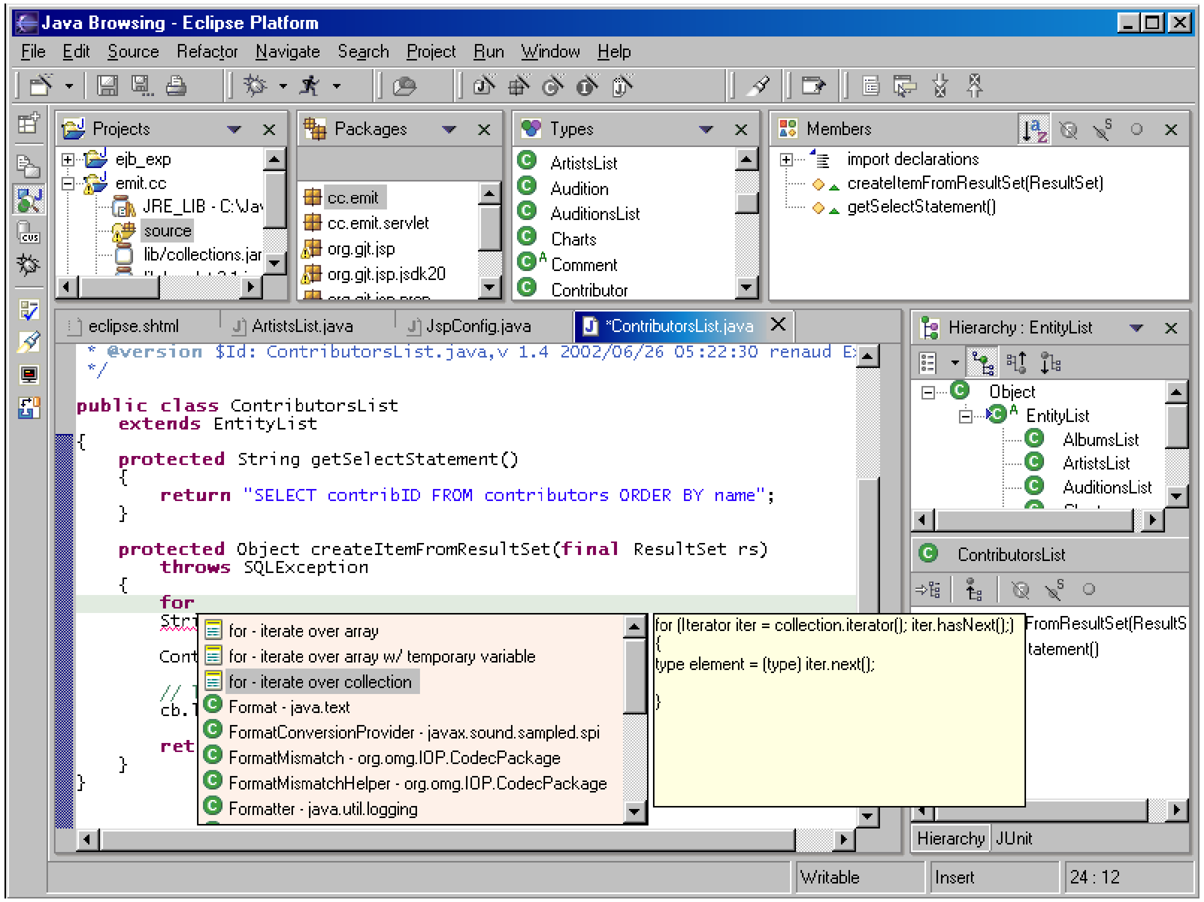
Mobiles: Nokia



A Java based hub in your pocket!

PC/Desktop: Eclipse

- RCP – Rich Client Platform, based on OSGi
- eRCP – Embedded RCP, also based on OSGi



```
 * @version $Id: ContributorsList.java,v 1.4 2002/06/26 05:22:30 renaud E
 */

public class ContributorsList
    extends EntityList
{
    protected String getSelectStatement()
    {
        return "SELECT contribID FROM contributors ORDER BY name";
    }

    protected Object createItemFromResultSet(final ResultSet rs)
        throws SQLException
    {
        for
        Str
        Cont
        // 
        cb.
        ret
    }
}
```

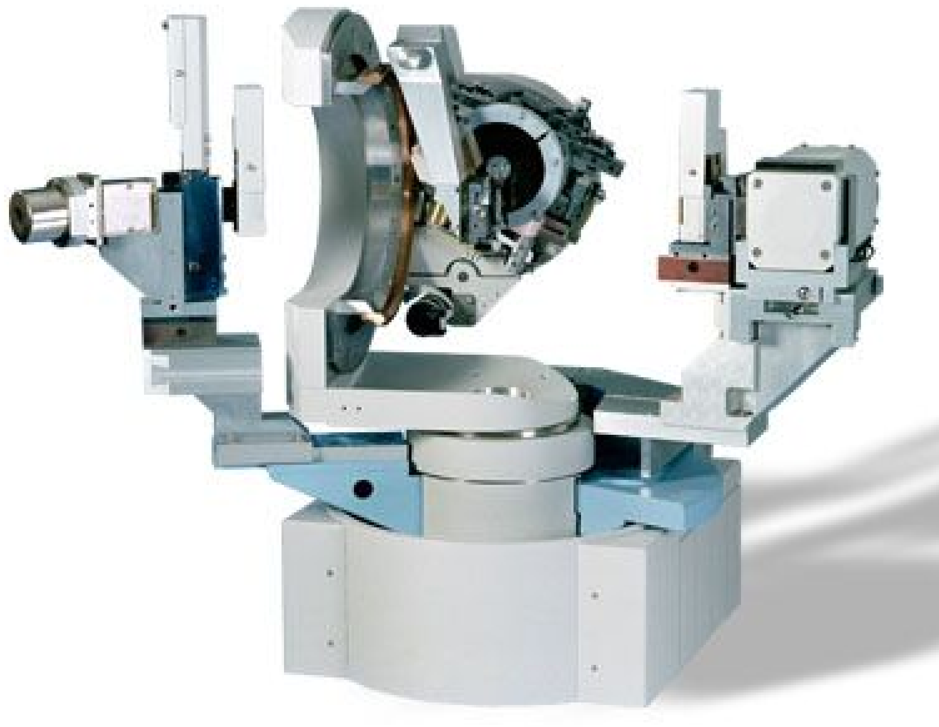
for - iterate over array
for - iterate over array w/ temporary variable
for - iterate over collection
Format - java.text
FormatConversionProvider - javax.sound.sampled.spi
FormatMismatch - org.omg.IOP.CodecPackage
FormatMismatchHelper - org.omg.IOP.CodecPackage
Formatter - java.util.logging

```
for (Iterator iter = collection.iterator(); iter.hasNext();)
{
type element = (type) iter.next();

}
```

Industrial Automation

- Industrial automation is a perfect match for OSGi based platforms
- Cost is less an issue than in hard embedded
- Benefit of better software process, more flexibility, remote management



Application Servers

- Trend in the industry to move application servers to OSGi:
  - IBM
  - JBoss
  - Jonas
  - BEA
- Others are looking …

What is the OSGi service platform?

- A Java™ framework for applications, that require:
  - Reliability
  - Large scale distribution
  - Wide range of devices
  - Collaborative
- Created through collaboration of industry leaders
- Spec 4.0 publicly available at www.osgi.org …

The big 3 Open Source Frameworks are:
Equinox (a subproject of Eclipse)
Knopflerfish (Gatespace Telematics)
Felix (Apache)

See http://www.aqute.biz/osgi for an overview

Why the OSGi Service Platform?

- What problems does the OSGi Service Platform address?

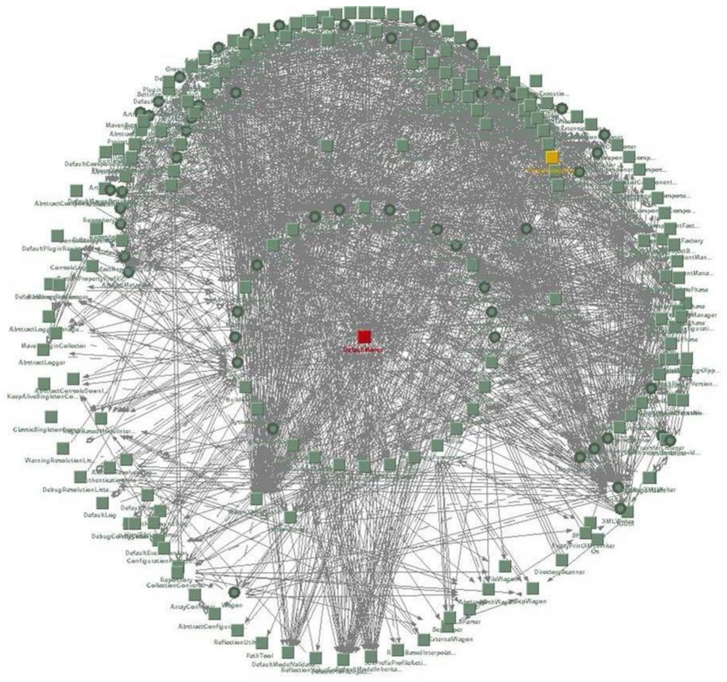
- A unified software market:
  - The limited (binary) software portability problem
  - The complexity of building heterogeneous software systems
  - Managing the software life-cycle on the device

Limited Binary Software Portability

- Lack of portability causes
  - Market friction: No large market of reusable components and applications
  - Reduced quality
- Unnecessary constraints on hardware and software architectures
  - CPUs differ widely in cost and performance
  - Linux is nice, but it is sub-optimal for smaller devices
- Benefits of the OSGi Platform
  - Applications run unmodified on different hardware and software

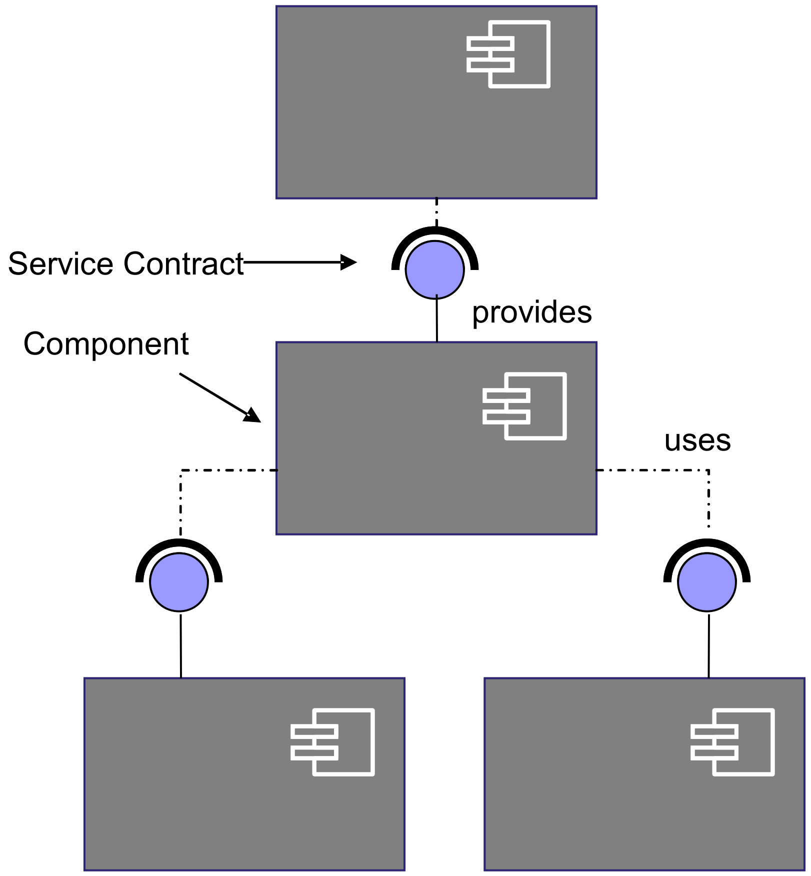
Limits of Object Oriented Technology

- Tangled web of Objects
- Coupling limits reusability
- Creates overly large systems
- Flexibility must be built in by the programmer
  - *Plug-in* architectures
  - Factories, Dependency Injection
- OSGi minimizes the coupling that is created by OO



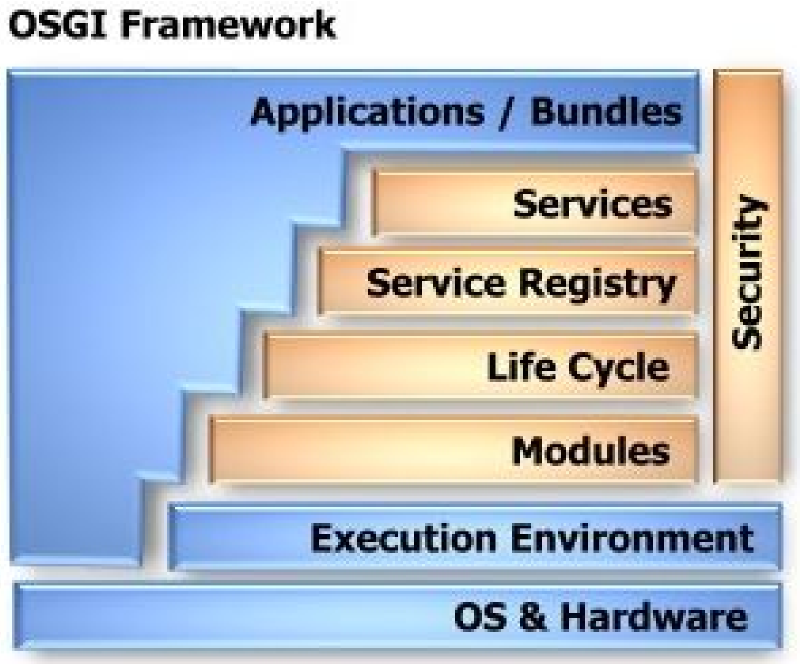
Service Oriented Architectures

- Separate the contract from the implementation
- Allows alternate implementations
- Dynamically discover and bind available implementations
- Based on contract (interface)
- Components are reusable
- Not coupled to implementation details



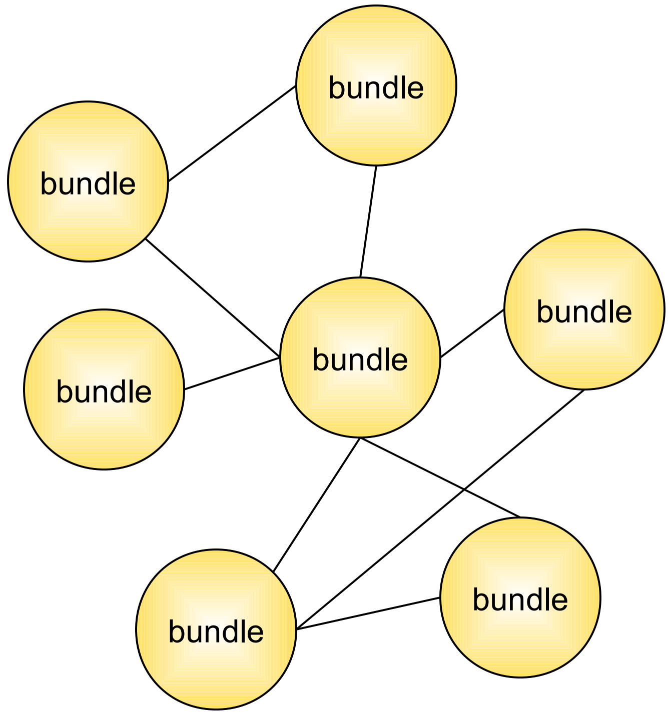
OSGi Framework

- Allows applications to share a single Java VM
- Classloading
- Isolation/Security
- Communication `& Collaborations between applications
- Life cycle management
- Policy free
  - Policies are provided by bundles
- API is fully self managed



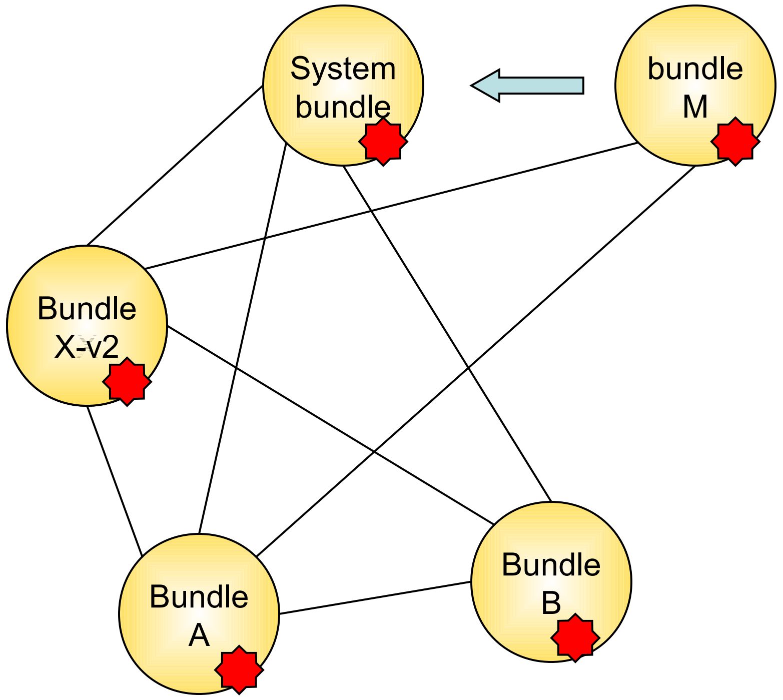
Module Layer

- Packaging of applications and libraries in *Bundles*
  - Raw Java has significant deployment issues
- Class Loading modularization
  - Raw Java provides the Class Path as an ordered search list, which makes it hard to control multiple applications
- Protection
  - Raw Java can not protect certain packages and classes
- Versioning
  - Raw Java can not handle multiple versions of the same package



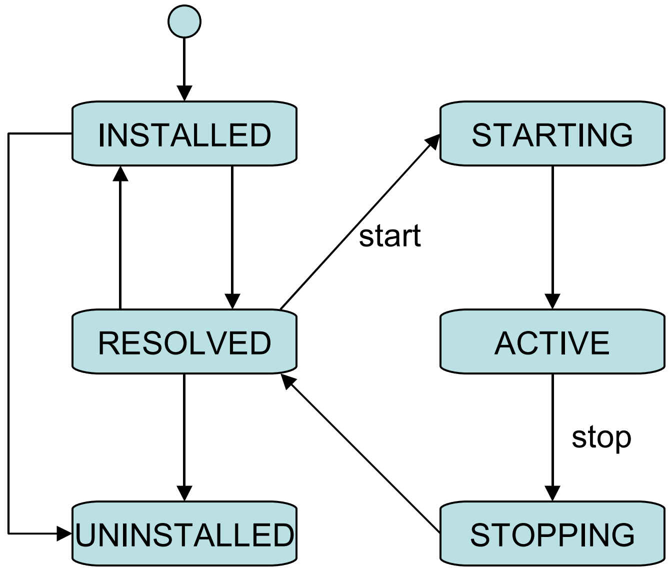
Life Cycle Layer

- System Bundle represents the OSGi Framework
- Provides an API for managing bundles
  - Install
  - Resolve
  - Start
  - Stop
  - Refresh
  - Update
  - Uninstall
- Based on the module layer



Life Cycle Layer

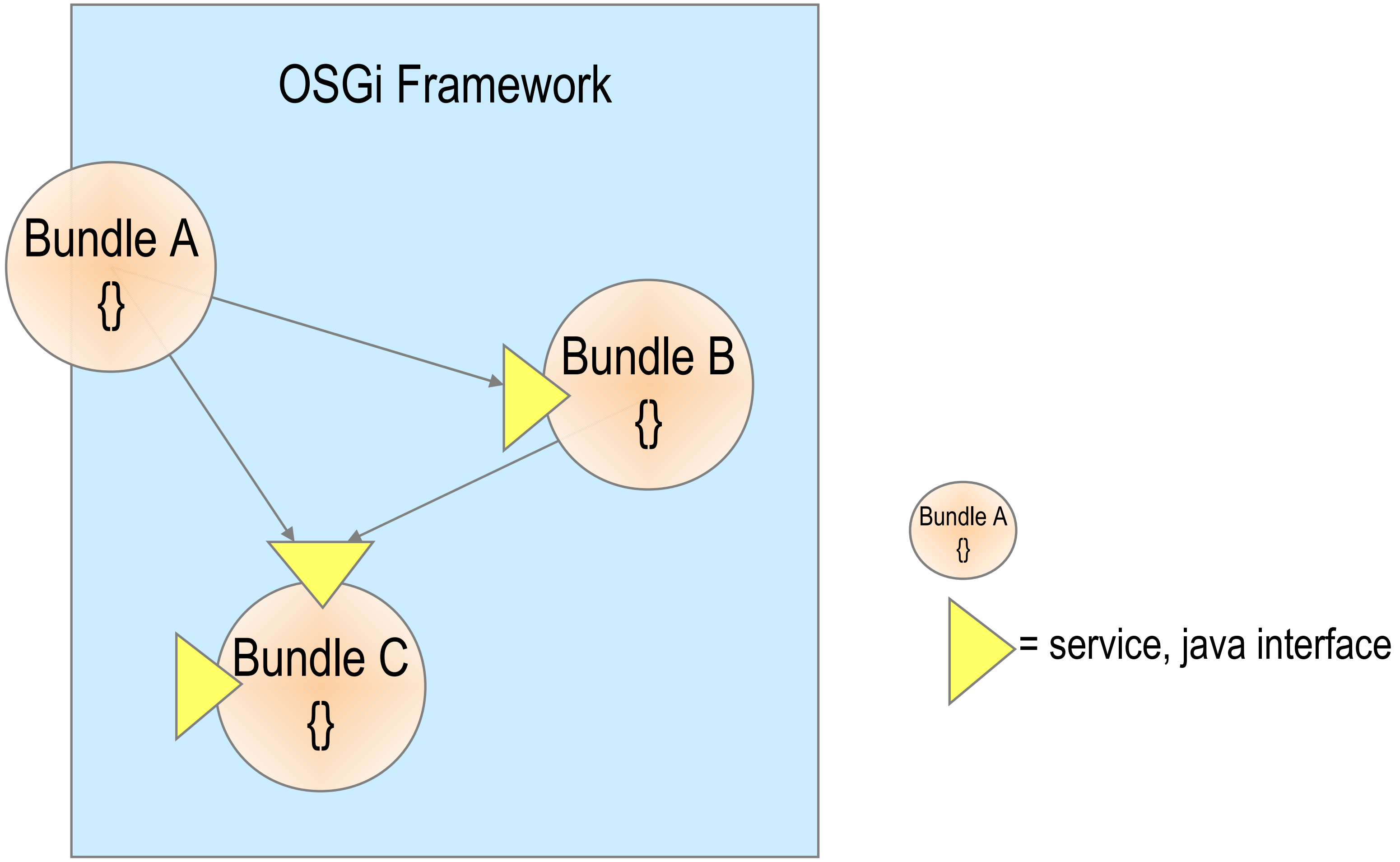
- Bundle is started by the Bundle Activator class
- Header in Manifest refers to this class
- Interface has 2 methods
  - Start: Initialize and return immediate
  - Stop: Cleanup
- The Activator gets a Bundle Context that provides access to the Framework functions
- Framework provides Start Level service to control the start/stop of groups of applications



Service Layer

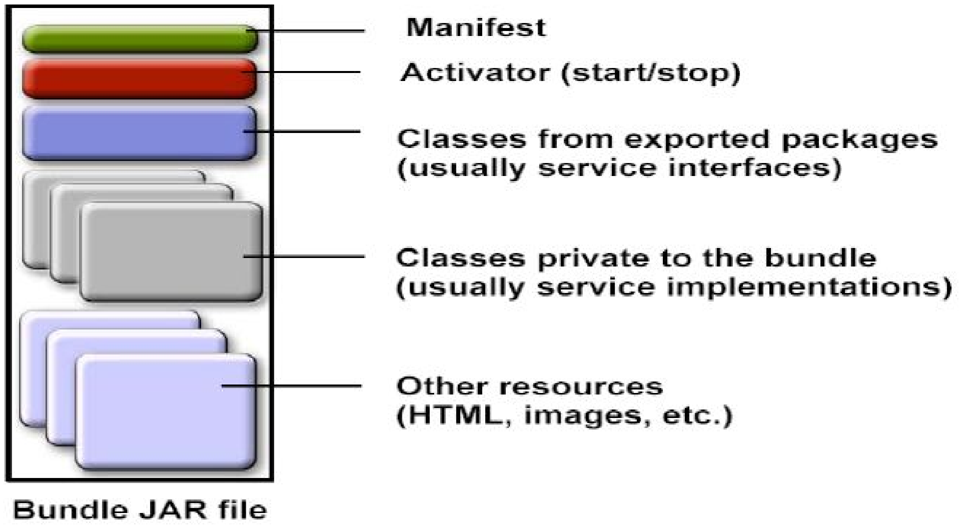
- Provides an in-VM service model
  - Discover (and get notified about) services based on their interface or properties
  - Bind to one or more services by
    - program control,
    - default rules, or
    - deployment configuration
- SOA Confusion
  - Web services bind and discover over the net
  - The OSGi Service Platform binds and discovers inside a Java VM
- The OSGi Alliance provides many standardized services

Framework Entities



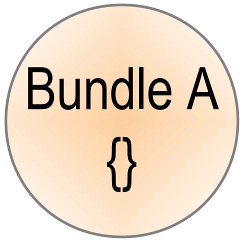
Bundles

- A *bundle* is the deliverable application
  - Like a Windows EXE file
  - Content is a JAR file
- A bundle registers zero or more services
  - A service is specified in a Java interface and may be implemented by multiple bundles
  - Services are bound to the bundle life-cycle
- Searches can be used to find services registered by other bundles
  - Query language


Manifest
Activator (start/stop)
Classes from exported packages (usually service interfaces)
Classes private to the bundle (usually service implementations)
Other resources (HTML, images, etc.)
Bundle JAR file

What is in a Bundle?

- A Bundle contains (normally in a JAR file):
  - Manifest
  - Code
  - Resources
- The Framework:
  - Reads the bundle's manifest
  - Installs the code and resources
  - Resolves dependencies
- During Runtime:
  - Calls the Bundle Activator to start the bundle
  - Manages java classpath
  - Handles the service dependencies
  - Calls the Bundle Activator to stop the bundle



Real code! Hello World (and Goodbye)

- This class implements the BundleActivator so that the Framework can start/stop the class
- The activator is referenced in the manifest

HelloWorld.java

```
package helloworld
public class HelloWorld
   implements BundleActivator {
   public void start(
      BundleContext context)
      throws Exception{
      System.out.println(
            "Hello world!!");
   }

   public void stop(
      BundleContext context)
      throws Exception {
            System.out.println(
            "Goodbye world!!");
   }
}
```

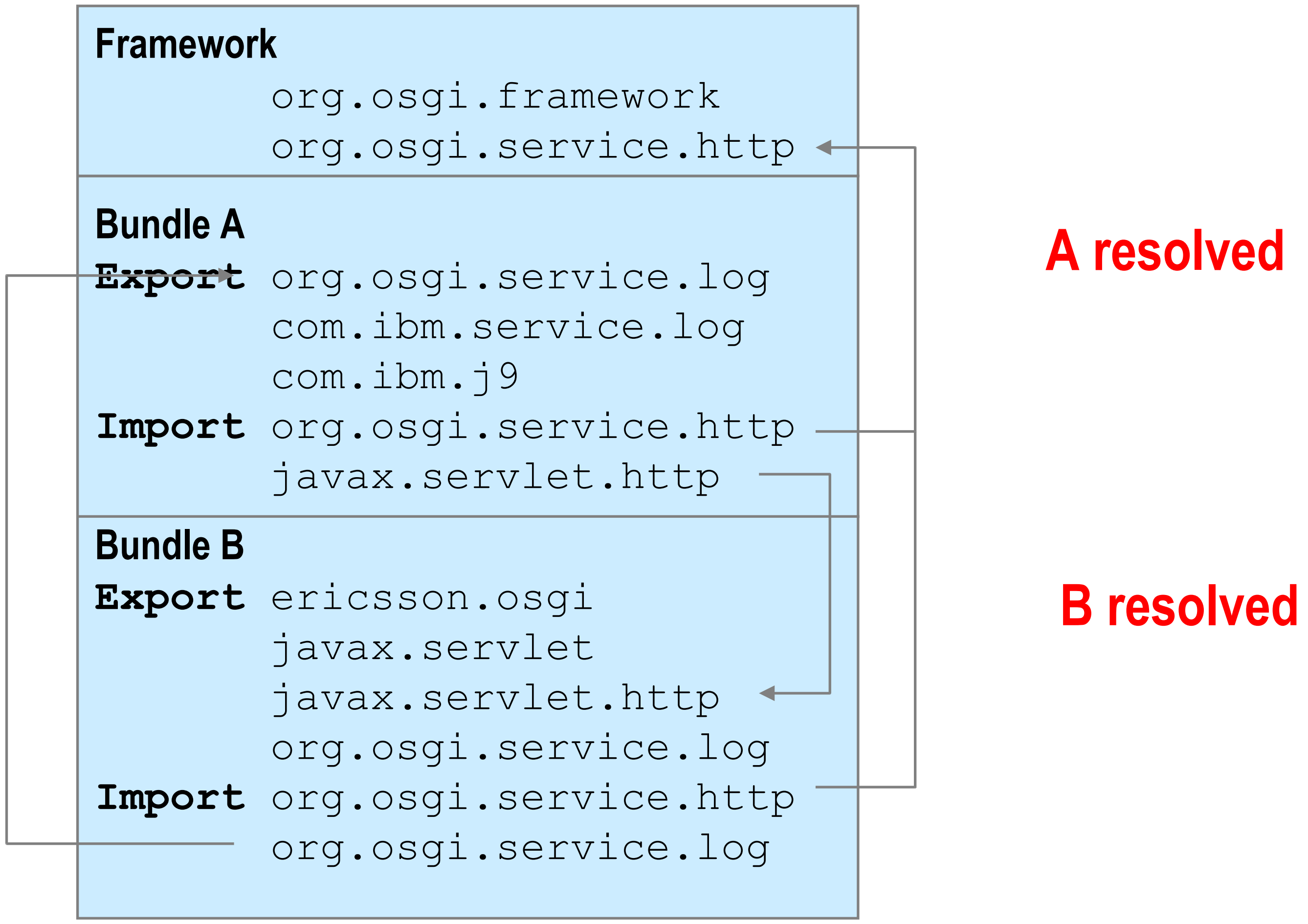
Real code! Hello World (and Goodbye)

- Bundle-Activator (used to notify the bundle of lifecycle changes)
- Import-Package (dependencies)

META-INF/MANIFEST.MF

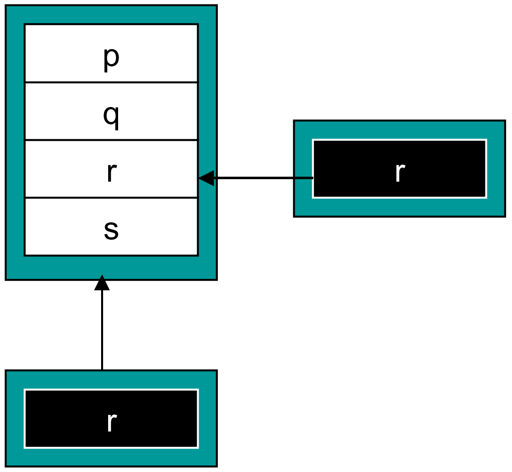
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Helloworld Plug-in
Bundle-SymbolicName: helloworld
Bundle-Version: 1.0.0
Bundle-Localization: plugin
Bundle-Activator: helloworld.Activator
Import-Package:
 org.osgi.framework;version="1.3.0"
```

OSGi dependency resolution



Package or Bundle Dependencies?

- The OSGi Specifications supports both Require-Bundle and Import-Package
- Require-Bundle creates a dependency on a complete bundle
  - Simple to use
  - Imports packages that are not used
- Import-Package creates a dependency on just a package
  - Creates less brittle bundles because of substitutability
  - More cumbersome to use (Tools!)
- In almost all cases, Import-Package is recommended because it eases deployment and version migration
- The specifications detail a number of additional problems with Require-Bundle



What Did We Learn

- The OSGi Service Platform is kind of a Java Operating System
- It simplifies:
  - Deployment Problems
  - Software composition
  - Software management
- Eclipse provides a development environment for OSGi Bundles
- Eclipse provides open source implementations of the OSGi specifications in the Equinox project

Conclusion

- The OSGi R4 Specifications consists of considerable more details than elucidated in this lecture
- There are many independent OSGi implementations on the market, both commercial and open source
  - Apache Felix, Atinav, Eclipse Equinox, Espial, IBM SMF, Knopflerfish/Ubiserv of Gatespace, ProSyst, …
- The OSGi specification are today running on mobile phones, PDAs, embedded computers, desktops, and mainframes
- The OSGi Alliance is working on making the OSGi specifications *the* standard for portable applications.

Benefits of Using the OSGi Service Platform

- Components are smaller
  - Easier to make
- Components are not coupled to other components
  - Gives reusability
- Excellent model for the myriad of customizations and variation that are required of today’s devices
- Collaborative model
  - Allows reuse of other components for most problems

The End

Further reading:

http://www.osgi.org

http://bundles.osgi.org

http://www.eclipse.org/osgi

http://www.aqute.biz