Building a Modular Server Platform with OSGi

Dileepa Jayakody

Software Engineer

WSO2 Inc.

WSO2 The open source SOA company

Outline

- Complex Systems
- OSGi for Modular Systems
- OSGi in SOA middleware
- Carbon : A modular server platform for middleware
- Carbon Architecture

Complex Systems

- A Complex Systems is a set of interconnected heterogeneous components
- They are hard to maintain, extend or even to understand!

Problems with Complex Systems

- Hard to maintain
- Code duplication
- Inconsistency
- Lack of interoperability
- Tightly coupled components

Solution : Modular Systems

What is OSGi?

- The dynamic modular system for Java
- Defines a way to create true modules and a way for those modules to interact at runtime to create a modular system
- A module: a bundle
  - A jar + manifest (bundle metadata)
- Can be installed, updated, and uninstalled without restarting the JVM

How does OSGi help?

OSGi helps to break down a complex systems into a collection of interacting modules

- Modularity
  - A bundle can share/hide information at package level
- Lifecycle Management
  - Separate class loader for each bundle
  - A bundle-lifecycle can be managed dynamically
- Services
  - Each bundle provides it's functionality as OSGi services to other bundles
  - Services are simply Java objects that implement a given interface
  - Implementation is loosely coupled
  - Bundles reuse a single Java object registered
  - In VM collaborative SOA model

SOA Middleware

- Middleware : The software which glues/connect different enterprise applications
- A Middleware platform provides;
    - Integration
    - Governance
    - Data Services
    - Business Processes
    - Connectivity Services
    - Identity and Security
    - Application Management
    - API Management

OSGi in Middleware

- Each SOA component can be represented by an OSGi bundle
  - Application Management
  - Mediation
  - Service Hosting etc.

- Separation of concern
  - Each OSGi bundle to provide a set of well-defined services
  - Loosely coupled components

- Dynamic loading of modules
  - Can extend the system dynamically by installing new bundles
  - Is supported by an underlying provisioning platform

WSO2 Carbon

- Carbon: An open-source fully componentized enterprise middleware platform based on OSGi
- Complemented by Stratos : The cloud enabled middleware platform (PaaS)
- Consists of a core set of components providing core services such as;
    - Security
    - Clustering
    - Logging
    - Transports
    - Registry
    - User management etc.
- Other components use the carbon core services and extend system functionality

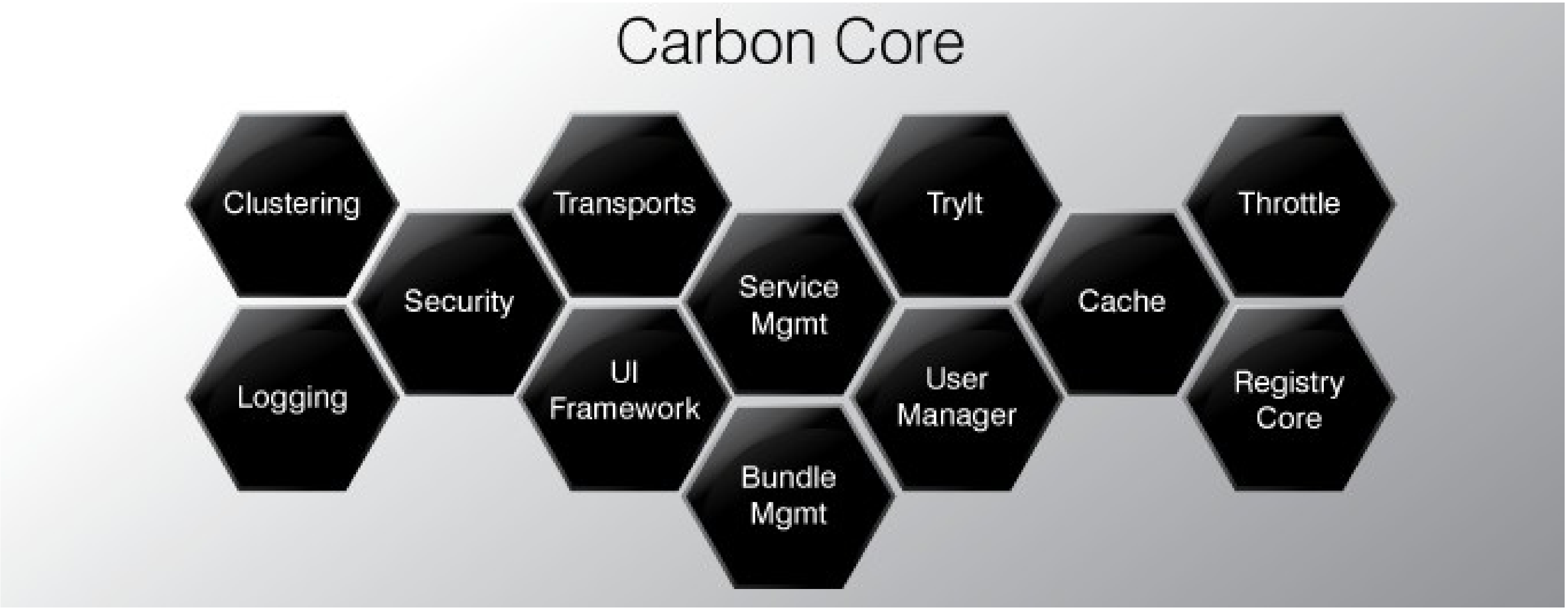
Why did we build Carbon?

- Fast growing complexity of the platform
- Overlapping components
- Duplicated functionality
- Difficulty to integrate functionality between products

Carbon Architecture

- Carbon Components
- Carbon Features
- Orbits
- Kernel Services
- Feature Provisioning

Carbon Core Architecture



Carbon Components

- A set of lean and self-consistent OSGi Bundles
- Lives in the Carbon Framework. Hence should adhere to rules defined in the Carbon Framework
- Fundamental Concept : FE-BE Separation
  - Every component has a core runtime, a well-defined front-end console and a clean SOA management interface
  - All completely pluggable and versioned
  - Connected via web-services
- Use Core Carbon Services
  - Via OSGi service registry
  - e.g. Registry Service, UserManager Service, etc.

Carbon Features

- Similar to Eclipse Features
- An aggregate of Carbon components
- Carbon Products are composed using Carbon features (Eclipse for Servers)
- A Carbon Feature
    - Is an installable unit which can be installed into any Carbon based product
    - Allows you to manage bundle and feature level dependencies
    - Can be installed using the Carbon Feature Manager
    - Can be published as a p2-repository
- A Carbon Feature Category
    - A logical grouping of features
    - Represents a Carbon Product

Carbon and Equinox P2

- The provisioning platform for Carbon features
- Using Equinox p2 in Carbon you can;
    - Install
    - Uninstall
    - Revert
    - Update Carbon features.
- Features can be installed from a P2 repository
- P2 feature repository can be either
    - File-based
    - Web-based

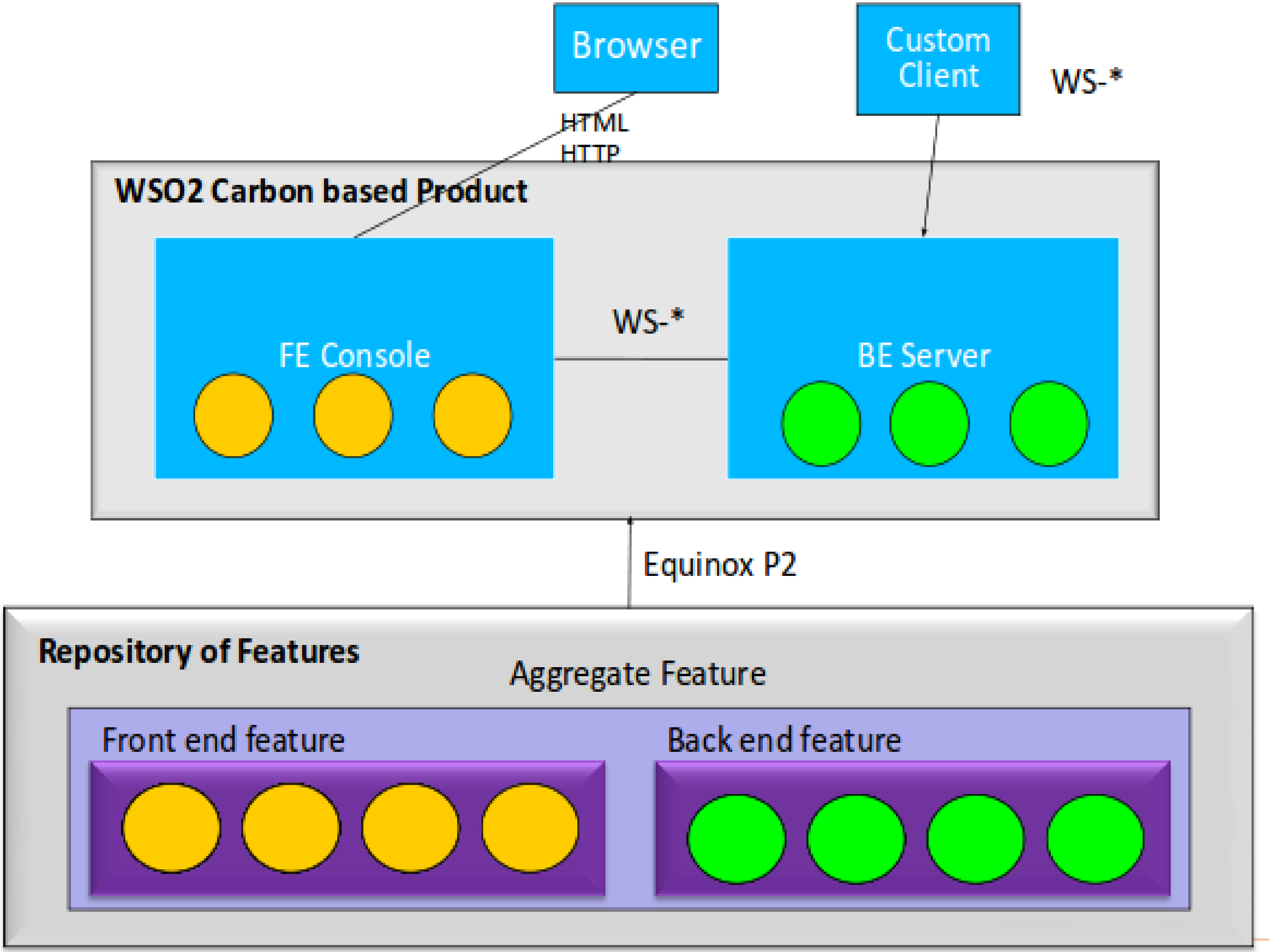
Orbits : External dependencies

- Lots of open source projects smoothly integrated
    - Apache Axis2
    - Apache tomcat
    - Apache ODE
    - Apache synapse
- Bundled as Orbits (Similar to Eclipse Orbit Project)
- Dependencies managed with versions
- External non-OSGi jars are auto-bundled (components/lib)
    - Gives the user more flexibility in using external libraries (eg: jdbc libraries)

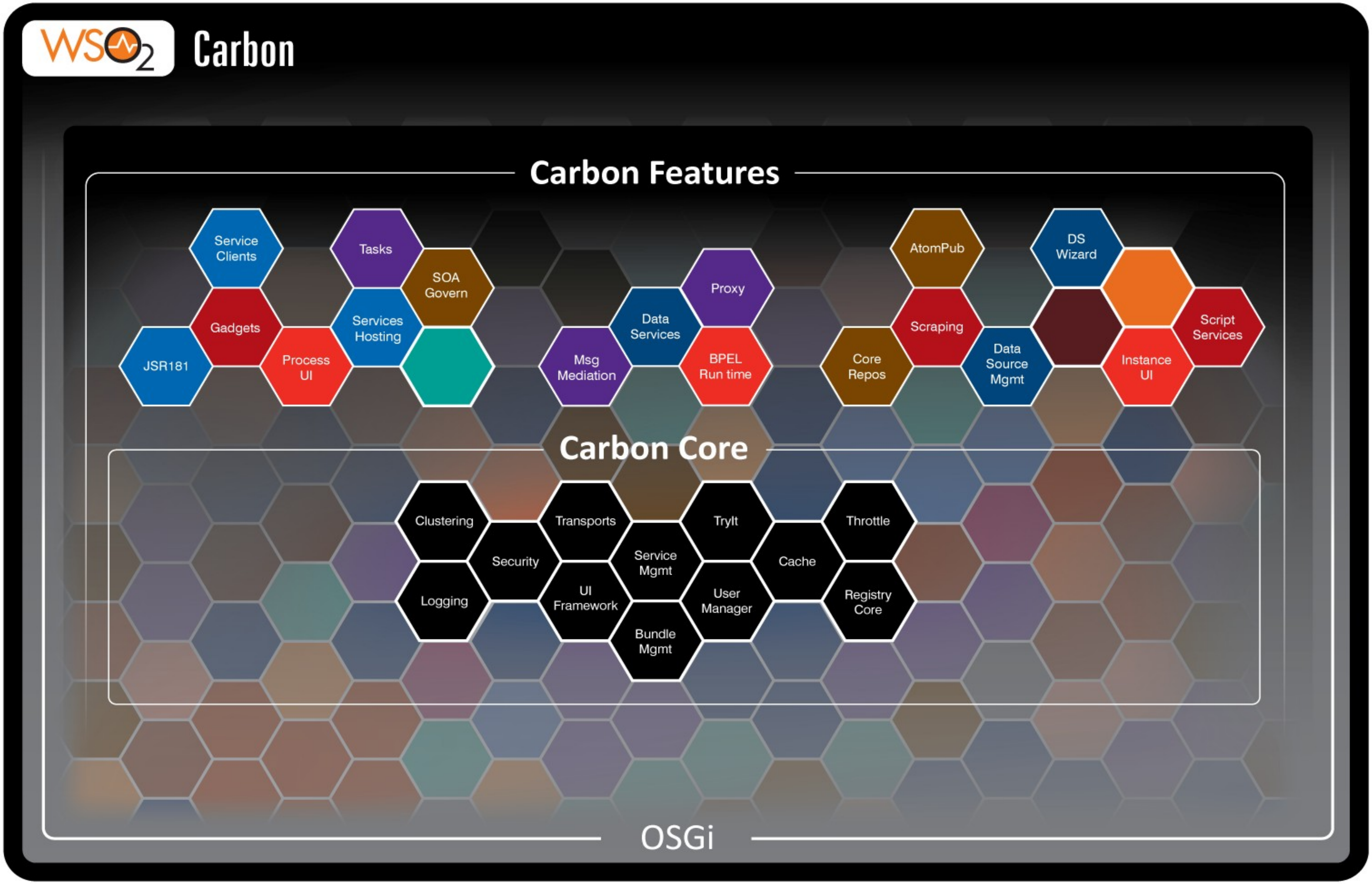
Kernel Services

- Kernel services through Carbon core
  - Execution (supporting services and workflows)
  - Data Storage
  - Security (user management, authentication, authorization)
  - User Interfaces
  - Other Services (monitoring, caching, clustering etc.)
- They are used by most components and simplify development of new components
- OSGi Maturity model : Level 4 (loosely coupled)
  - Separation of interface from implementation
  - Provides a services-based module collaboration
  - Dependencies semantically versioned

The Big Picture



The Big Picture



OSGi best practises in Carbon

- Controlled number of exports from a bundle
- Semantic Versioning for Imports/Exports
    - Version ranges for imports to handle backward compatibilities
    - To manage dependencies between components
    - To host different versions of the same package and correctly handle dependencies
- Avoided usage of Required-Bundle
    - To avoid tight-coupling
- Declarative services as the dependency injection model
    - To manage dependencies between components dynamically
- Use of OSGi HttpService
    - To consume http requests by bundles

Carbon Component: Development Process

- Develop the Carbon component
  - Back-end component (BE OSGi bundles)
  - Front-end component (FE OSGi bundles)
  - Common bundles, if any

- Develop the corresponding feature
  - BE/Server Feature
  - FE/UI Feature
  - Composite Feature

- Install into a Carbon based product
  - By integrating with the product build system or;
  - By developing a feature repository and installing using Feature Manager

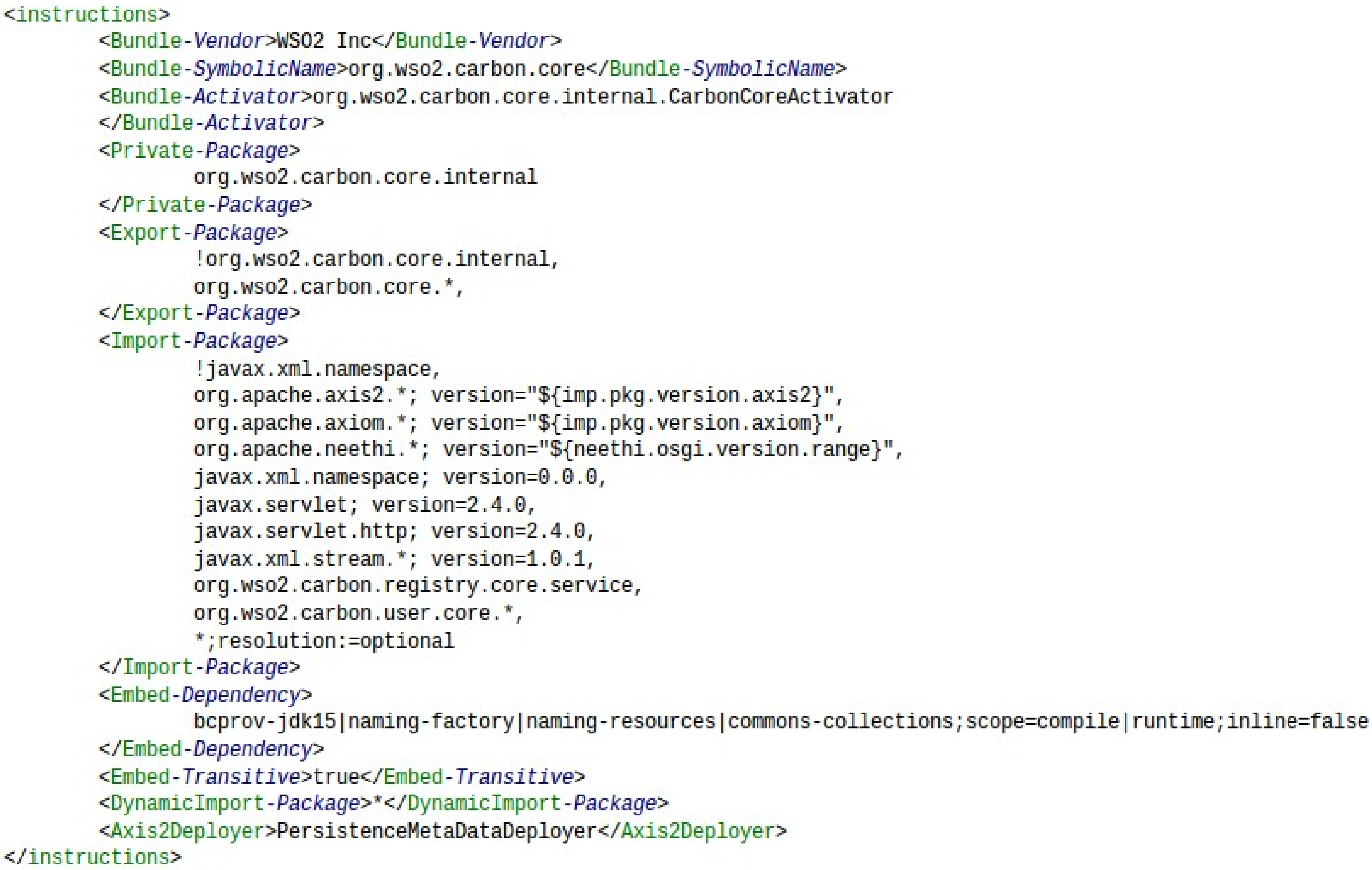
Tools for Carbon

- Maven
  - To build Carbon source
- Maven bundle plugin
  - To build bundles
  - Manage dependencies
- Maven scr plugin (scr : service components runtime)
  - Service Components are defined through annotations
  - Plugin creates the necessary descriptors for the OSGi Declarative Services
- Carbon P2 plugin (Developed at WSO2)
  - To build features & feature categories
  - To build feature repositories
  - To build product profiles

Maven Bundle Plugin

- Is the Maven version of BND tool by Peter Kriens
- The primary goal of BND is to relieve the bundle developer from the pain of creating the bundle manifest
- Wraps BND to make it work with Maven project structure
- BND instructions;
  - Manifest headers :
    - These instructions are copied to the manifest file as manifest headers. Values of these instruction are either copied, or generated by the Plugin.
  - Variables :
    - These instructions act as variables and can be used for property substitution
  - Directives :
    - These perform some special processing

Maven Bundle Plugin (Apache Felix Plugin)



```
<instructions>
        <Bundle-Vendor>WSO2 Inc</Bundle-Vendor>
        <Bundle-SymbolicName>org.wso2.carbon.core</Bundle-SymbolicName>
        <Bundle-Activator>org.wso2.carbon.core.internal.CarbonCoreActivator
        </Bundle-Activator>
        <Private-Package>
                org.wso2.carbon.core.internal
        </Private-Package>
        <Export-Package>
                !org.wso2.carbon.core.internal,
                org.wso2.carbon.core.*,
        </Export-Package>
        <Import-Package>
                !javax.xml.namespace,
                org.apache.axis2.*; version="${imp.pkg.version.axis2}",
                org.apache.axiom.*; version="${imp.pkg.version.axiom}",
                org.apache.neethi.*; version="${neethi.osgi.version.range}",
                javax.xml.namespace; version=0.0.0,
                javax.servlet; version=2.4.0,
                javax.servlet.http; version=2.4.0,
                javax.xml.stream.*; version=1.0.1,
                org.wso2.carbon.registry.core.service,
                org.wso2.carbon.user.core.*,
                *;resolution:=optional
        </Import-Package>
        <Embed-Dependency>
                bcprov-jdk15|naming-factory|naming-resources|commons-collections;scope=compile|runtime;inline=false
        </Embed-Dependency>
        <Embed-Transitive>true</Embed-Transitive>
        <DynamicImport-Package>*</DynamicImport-Package>
        <Axis2Deployer>PersistenceMetaDataDeployer</Axis2Deployer>
</instructions>
```

Carbon P2 Plugin

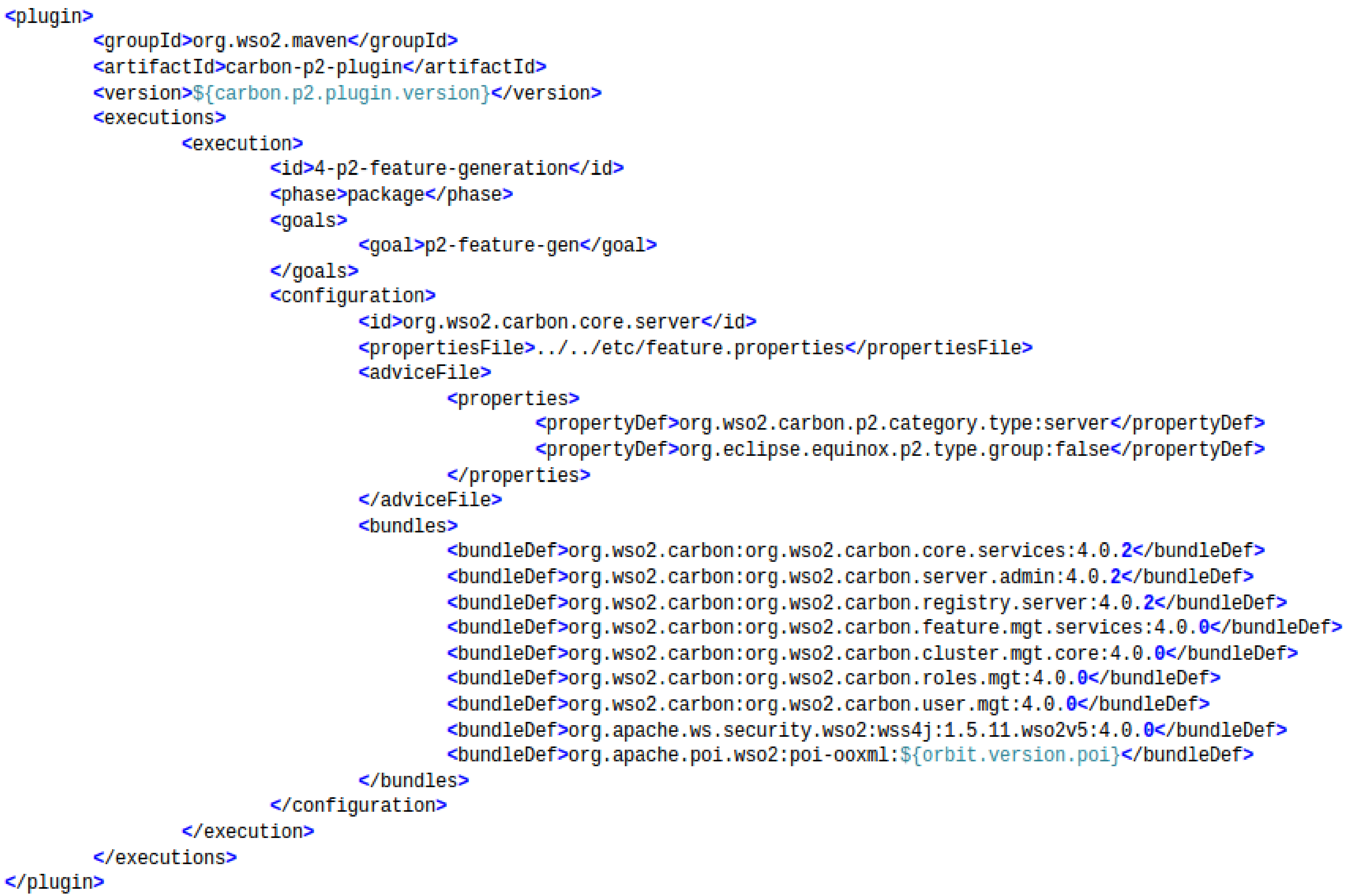
- Maven tool for creating features, feature-repositories and carbon-products

- Uses Eclipse FeaturesAndBundles Publisher tool under the hood

- Instructions
  - bundleDef : includes a bundle
  - IncludedFeatureDef : includes a feature as a sub-feature
  - ImportFeatureDef : defines a dependency to an external feature

- Type of the Carbon Feature can be defined as a p2-property
  - org.wso2.carbon.p2.category.type:server
  - org.wso2.carbon.p2.category.type:console

Generating features with p2-plugin

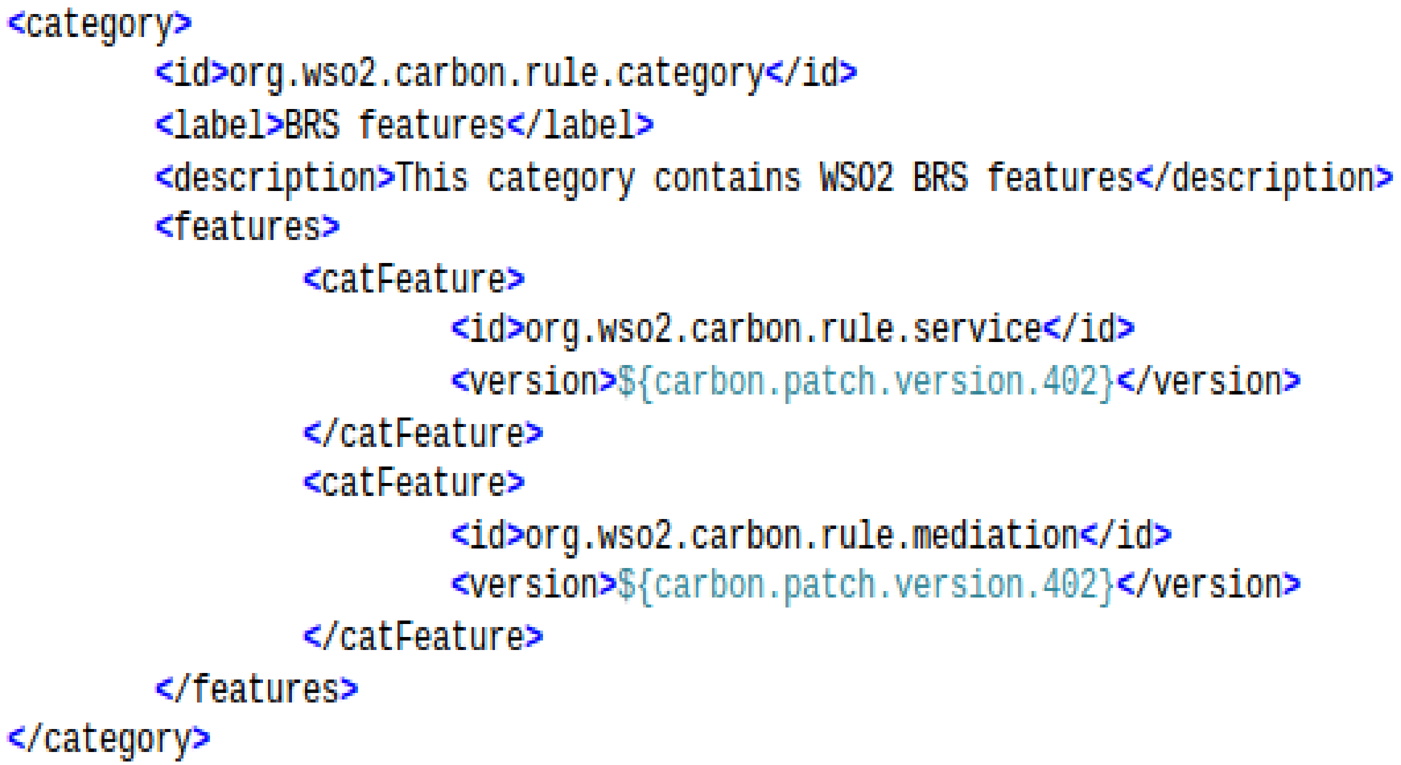


```
<plugin>
    <groupId>org.wso2.maven</groupId>
    <artifactId>carbon-p2-plugin</artifactId>
    <version>${carbon.p2.plugin.version}</version>
    <executions>
        <execution>
            <id>4-p2-feature-generation</id>
            <phase>package</phase>
            <goals>
                <goal>p2-feature-gen</goal>
            </goals>
            <configuration>
                <id>org.wso2.carbon.core.server</id>
                <propertiesFile>../../etc/feature.properties</propertiesFile>
                <adviceFile>
                    <properties>
                        <propertyDef>org.wso2.carbon.p2.category.type:server</propertyDef>
                        <propertyDef>org.eclipse.equinox.p2.type.group:false</propertyDef>
                    </properties>
                </adviceFile>
                <bundles>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.core.services:4.0.2</bundleDef>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.server.admin:4.0.2</bundleDef>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.registry.server:4.0.2</bundleDef>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.feature.mgt.services:4.0.0</bundleDef>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.cluster.mgt.core:4.0.0</bundleDef>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.roles.mgt:4.0.0</bundleDef>
                    <bundleDef>org.wso2.carbon:org.wso2.carbon.user.mgt:4.0.0</bundleDef>
                    <bundleDef>org.apache.ws.security.wso2:wss4j:1.5.11.wso2v5:4.0.0</bundleDef>
                    <bundleDef>org.apache.poi.wso2:poi-ooxml:${orbit.version.poi}</bundleDef>
                </bundles>
            </configuration>
        </execution>
    </executions>
</plugin>
```

The open source SOA company

Feature Categories

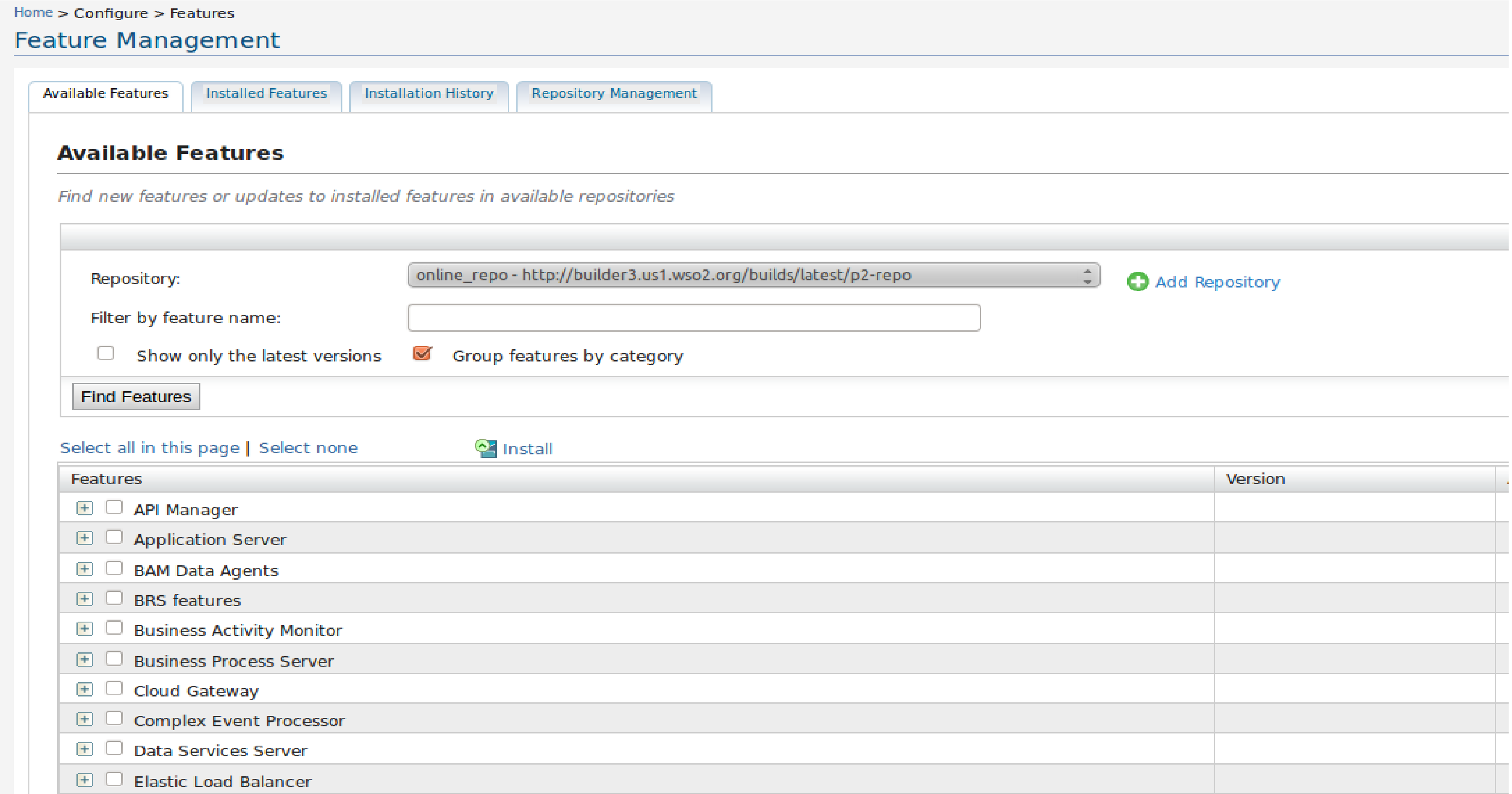
- Uses P2 Category publisher under the hood
- Generates the category.xml and provide it to the category publisher



```
<category>
        <id>org.wso2.carbon.rule.category</id>
        <label>BRS features</label>
        <description>This category contains WSO2 BRS features</description>
        <features>
                <catFeature>
                        <id>org.wso2.carbon.rule.service</id>
                        <version>${carbon.patch.version.402}</version>
                </catFeature>
                <catFeature>
                        <id>org.wso2.carbon.rule.mediation</id>
                        <version>${carbon.patch.version.402}</version>
                </catFeature>
        </features>
</category>
```

Feature Manager

- UI based tool to perform provisioning actions on Carbon
- Helps to compose carbon products with the required features



Home > Configure > Features

Feature Management

Available Features

Find new features or updates to installed features in available repositories

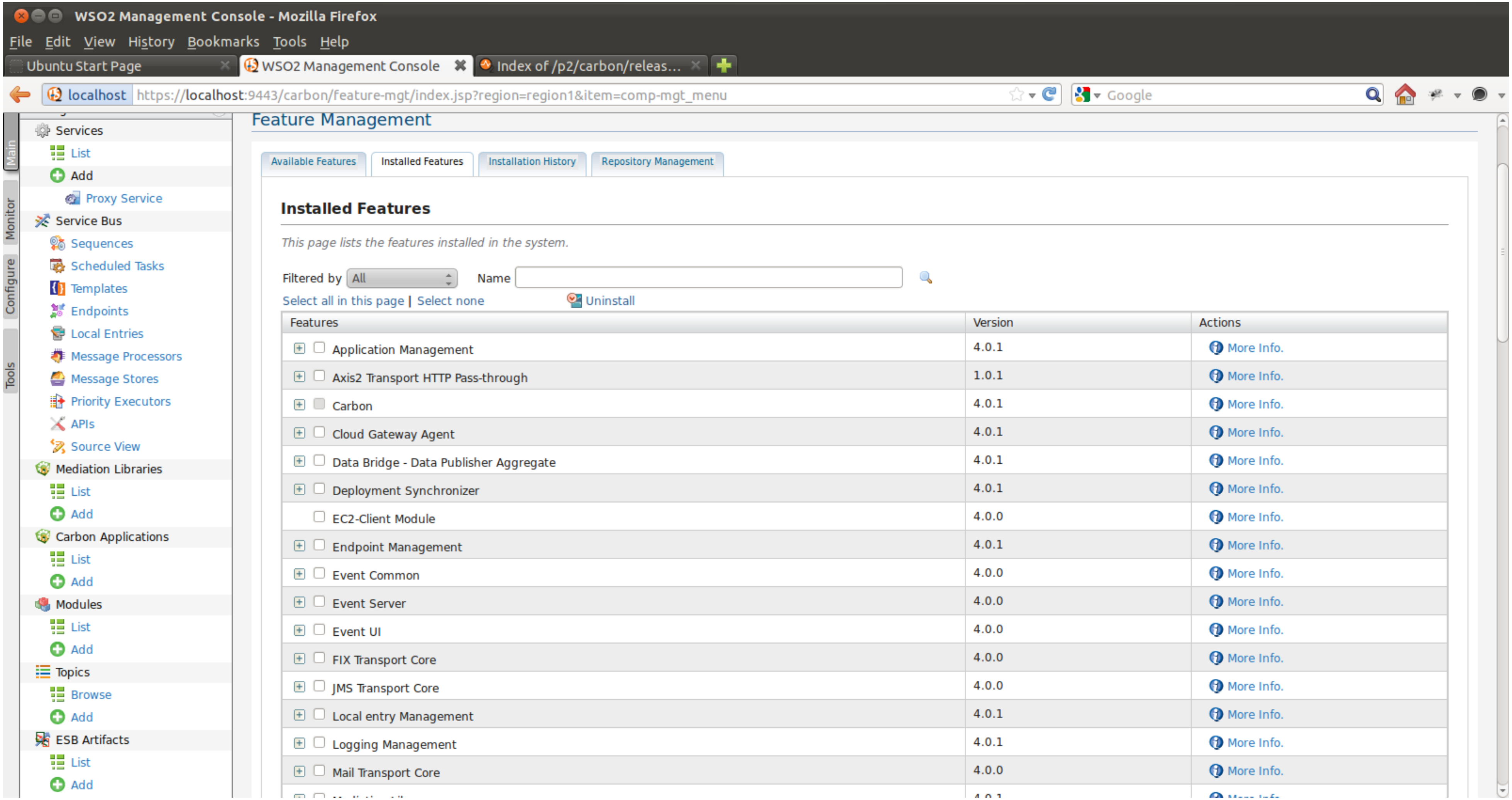
Repository: online_repo - http://builder3.us1.wso2.org/builds/latest/p2-repo

Filter by feature name:

□ Show only the latest versions ☑ Group features by category

Features	Version
⊞ □ API Manager	
⊞ □ Application Server	
⊞ □ BAM Data Agents	
⊞ □ BRS features	
⊞ □ Business Activity Monitor	
⊞ □ Business Process Server	
⊞ □ Cloud Gateway	
⊞ □ Complex Event Processor	
⊞ □ Data Services Server	
⊞ □ Elastic Load Balancer	

Installed Features



Feature Management

Installed Features

This page lists the features installed in the system.

Filtered by: All Name:

Select all in this page | Select none Uninstall

Features	Version	Actions
⊞ □ Application Management	4.0.1	More Info.
⊞ □ Axis2 Transport HTTP Pass-through	1.0.1	More Info.
⊞ □ Carbon	4.0.1	More Info.
⊞ □ Cloud Gateway Agent	4.0.1	More Info.
⊞ □ Data Bridge - Data Publisher Aggregate	4.0.1	More Info.
⊞ □ Deployment Synchronizer	4.0.1	More Info.
□ EC2-Client Module	4.0.0	More Info.
⊞ □ Endpoint Management	4.0.1	More Info.
⊞ □ Event Common	4.0.0	More Info.
⊞ □ Event Server	4.0.0	More Info.
⊞ □ Event UI	4.0.0	More Info.
⊞ □ FIX Transport Core	4.0.0	More Info.
⊞ □ JMS Transport Core	4.0.0	More Info.
⊞ □ Local entry Management	4.0.1	More Info.
⊞ □ Logging Management	4.0.1	More Info.
⊞ □ Mail Transport Core	4.0.0	More Info.

WSO2 The open source SOA company

OSGi Future in Carbon

- Multi-tenant OSGi
    - Regions for each tenant
    - SOA artifacts can be partitioned per tenant in OSGi runtime

Summary

- Complex systems are hard to maintain and extend
- The solution: modular systems
- OSGi : a true dynamic modular system for Java
- OSGi in SOA middleware
- Carbon : a modular server platform for middleware

Questions?

dileepa@wso2.com
dev@wso2.org

Thank you!