SUBSYSTEMS IN THE WILD

Graham Charters

Senior Technical Staff Member

IBM

Contents

…with a demo or two along the way

Motivation

- Many Enterprise Java platforms offer bundle collections
  - Apache Aries – Applications
  - Apache Karaf – Features
  - Eclipse Virgo – Plans, PARs
  - WebSphere – OSGi Applications, Composite, Features
  - Oracle GlassFish – Applications
  - Paremus Service Fabric – System Parts
- Standardization enables
  - Portability
  - Tools
  - Ecosystem

Subsystem Concepts

- A Grouping Resources (Bundles)
- An Identity
- Scoping for visibility
- Group Management
- Documentation (coming in 1.1)

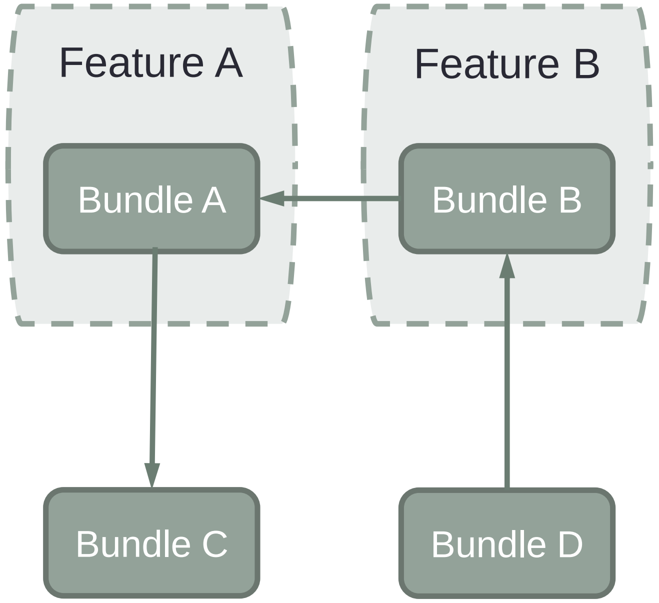
Subsystem Types

- Subsystem types aligned with common sharing policies

Type	Sharing Policy
Feature	Share everything
Application	Share everything in, share nothing out
Composite	Explicit selective sharing

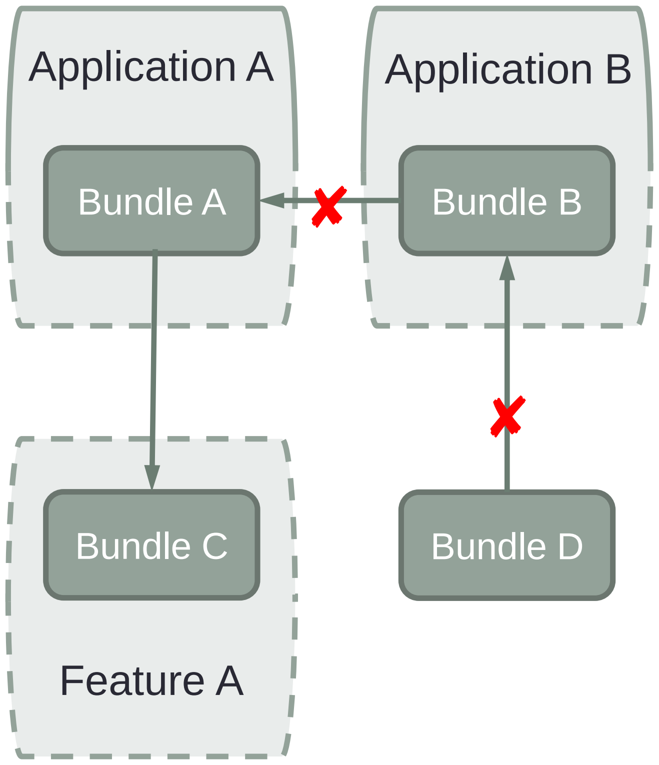
Feature Subsystems

- A grouping of related artefacts
- Simplifies re-use & provisioning
- Raw resources in the runtime
- Useful for frameworks & libraries



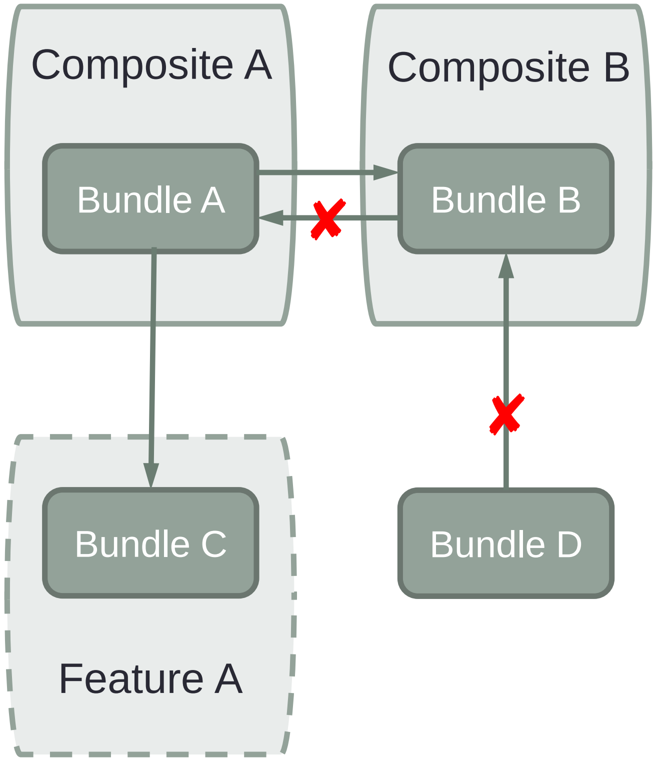
Application Subsystems

- Isolated group of related artefacts
  - Share nothing out
  - Implicitly share all in
- Useful for independent applications in same process
  - cheap/weak multi-tenancy
- Can share common infrastructure
  - e.g. frameworks or libraries



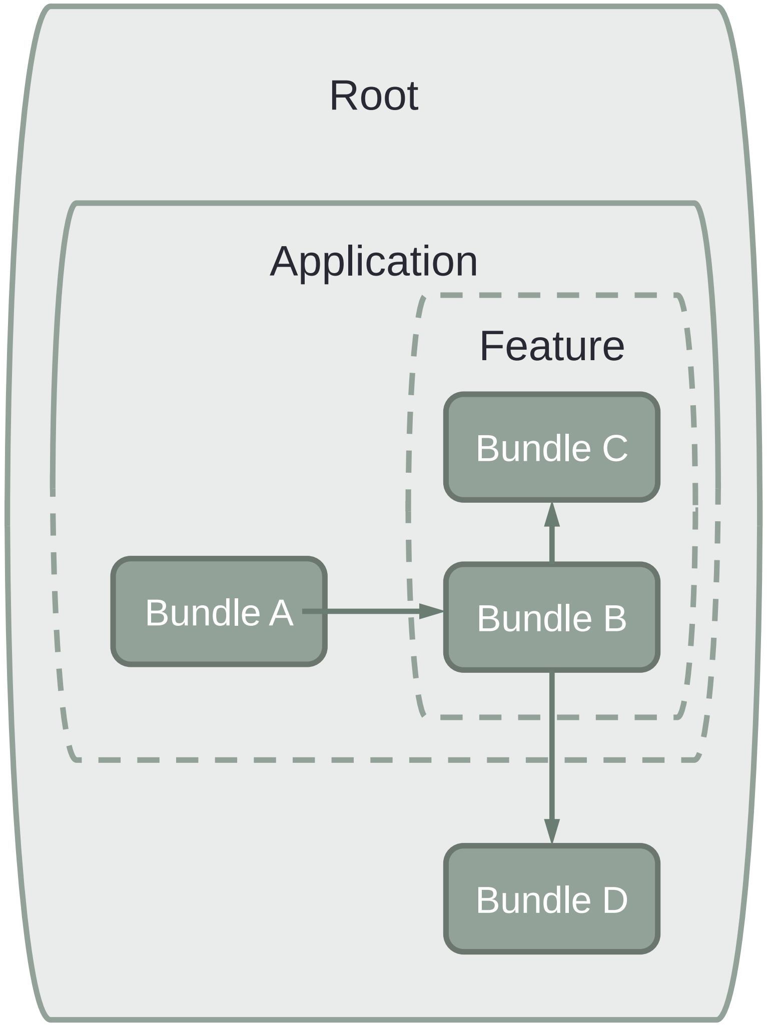
Composite Subsystems

- Isolated group of related artefacts
  - Explicitly share in
  - Explicitly share out
- Useful for controlling coupling of “subsytems” (hide internals)



Nesting

- Subsystems can be nested
- All subsystems are descendants of the “Root Subsystem”
- Enables runtime subsystem ‘sharing’
- Enables logical groupings



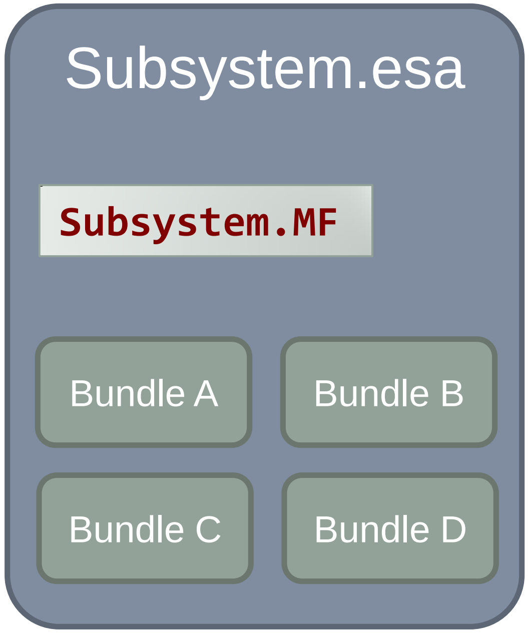
Definition

- Subsystem manifest format
  - Jar manifest without the annoying restrictions
  - Familiar to bundle programmers
- Configuration by exception
  - Can be left out altogether
- Version ranges enabled deployment flexibility

```
Subsystem-ManifestVersion: 1.0
Subsystem-Name: HttpService
Subsystem-Description: This feature enables the http service
Subsystem-SymbolicName: com.acme.httpService
Subsystem-Version: 1.0.0
Subsystem-Type: osgi.subsystem.feature
Subsystem-Content: httpServiceWab;version="1.0.0",
  org.eclipse.equinox.http.servlet;version="1.1.200",
  org.eclipse.osgi.services;version="3.3.0"
```

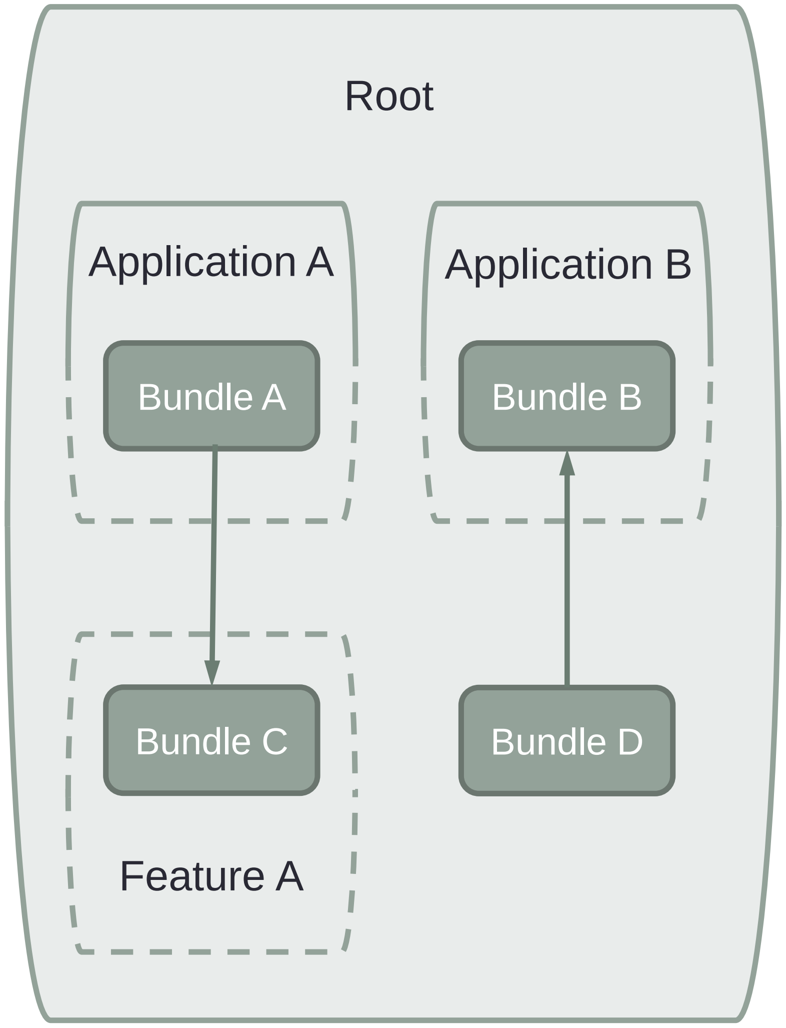
Packaging

- Zip file with .esa extension
- Optionally contains resources (bundles) for provisioning
  - duplicates binaries
  - bloats deployment
  - inhibits update
  - does not influence runtime sharing
  - uses repositories if not
- Optionally contains Subsystem manifest
  - defaulted if not



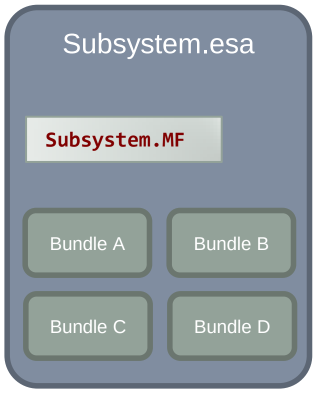
Management

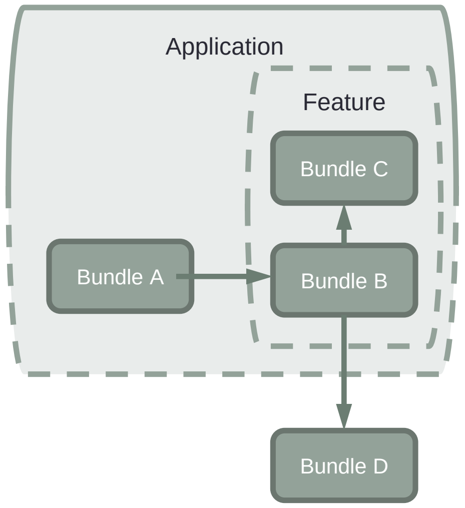
- Subsystem service
  - Introspection of subsystem
  - Lifecycle management of subsystem
  - Management of child subsystems (e.g. install)
- Roots subsystem – solves the bootstrap problem
- Really only for systems programmers – runtimes should simplify management capabilities

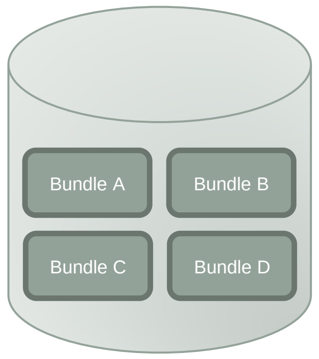


Repositories

- Repositories represent collections of resources considered during provisioning
- The Subsystem archive contents
  - Simplifies operations, bloats deployment
- The visible set of installed things
- Configured repository services
  - Enables single binary for install and service
  - Fits well with ‘continuous delivery’

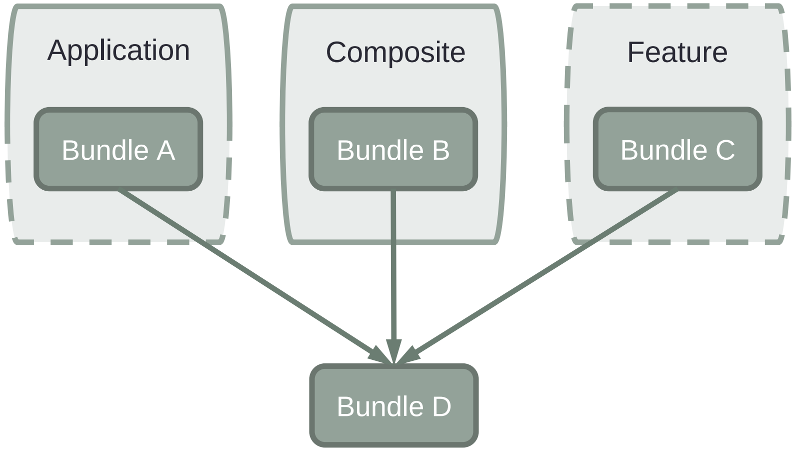






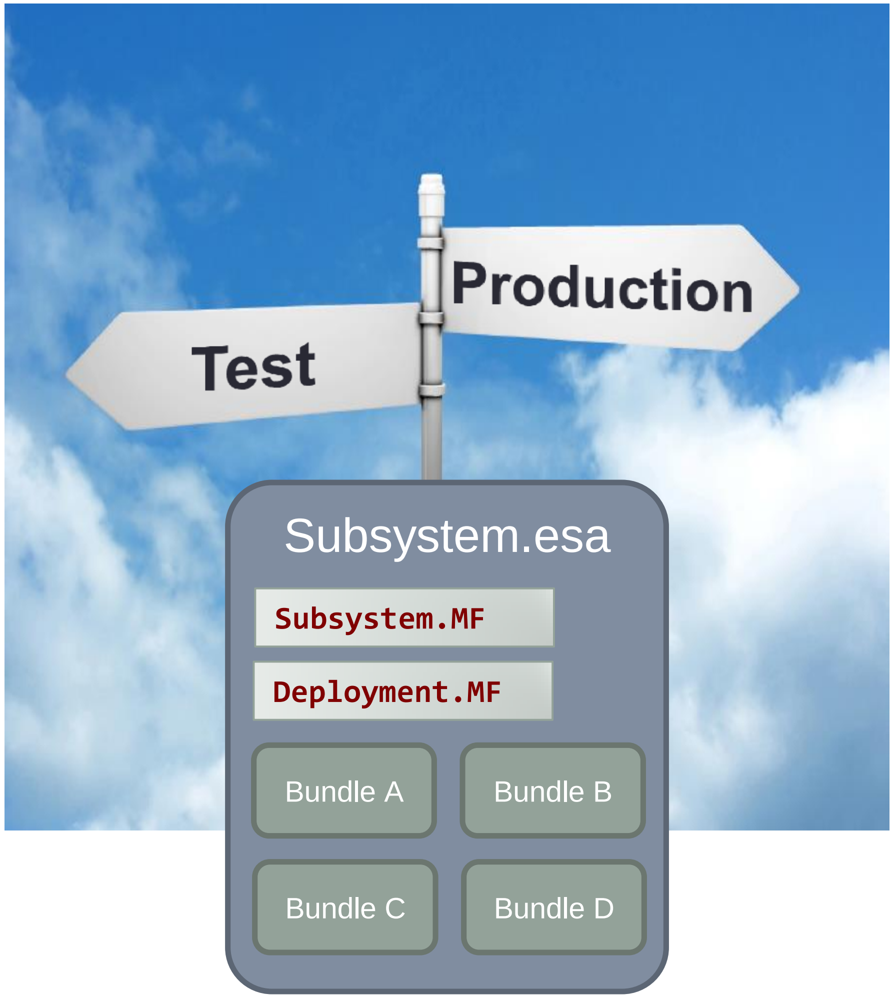
Dependency Management

- Installs missing dependencies
- Removes redundant dependencies
- Dependency installation maximises runtime sharing
- Dependency types
  - Package
  - Services via declarative models (optional)
  - Require-Capability



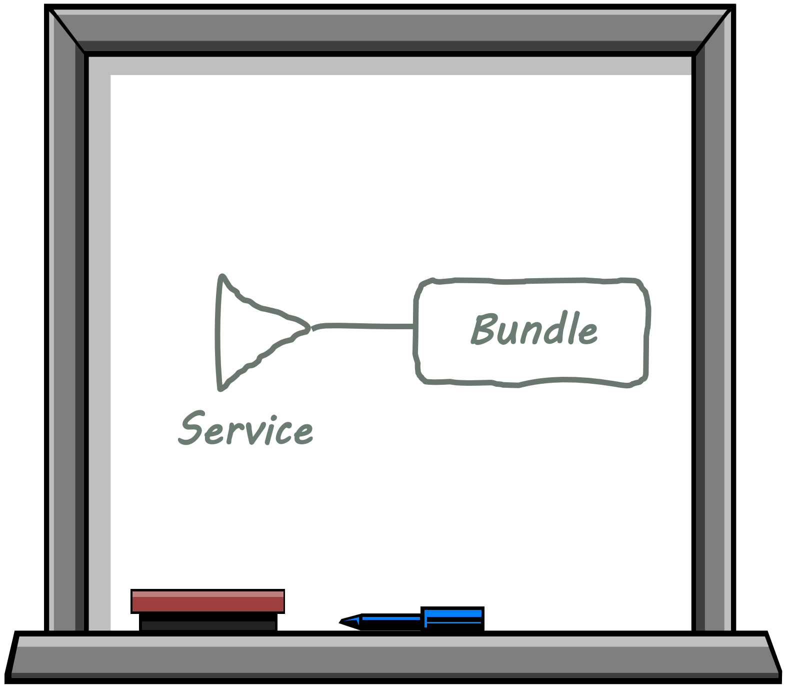
From Test to Production

- Product deployment resolution uncertainty
- Deployment manifests lock down deployment
- Repository and install governance necessary for guarantees



Extenders and whiteboards

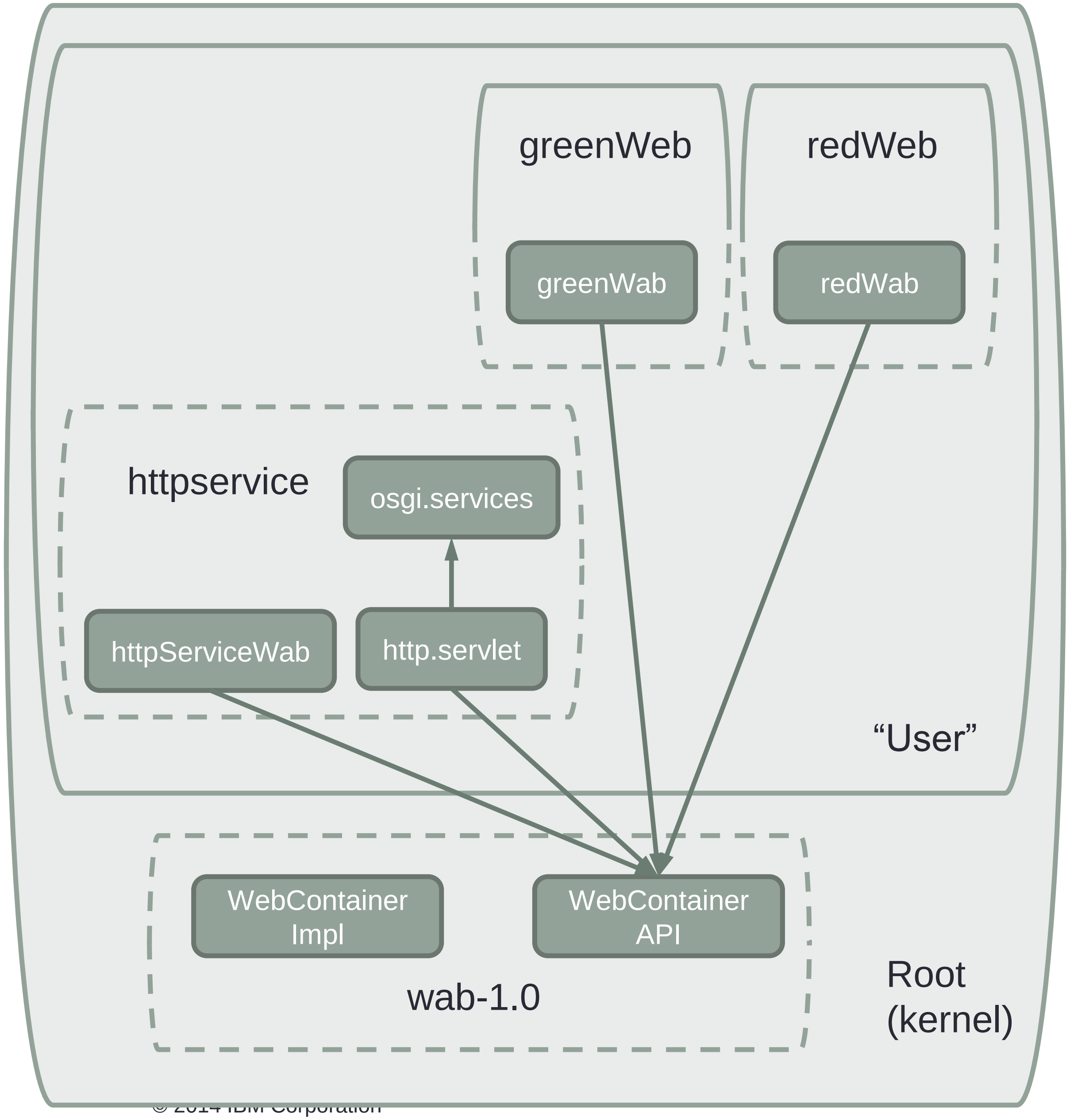
- Extender & whiteboard patterns commonplace
- Rely on visibility of bundle & service lifecycle events
- Subsystem scoping can hide these events from extenders
- The System Bundle is your friend
  - Sees all events
  - Can be found by any bundle using its location string
  - Use this to register trackers



```
Bundle systemBundle =
bc.getBundle(org.osgi.framework.Constants.SYSTEM_BUNDLE_LOCATION);
```

Demo

- **httpservice:** http service whiteboard
- **greenWeb, redWeb:** register http services
- **wab-1.0:** web container



In the Wild

- Confession time…

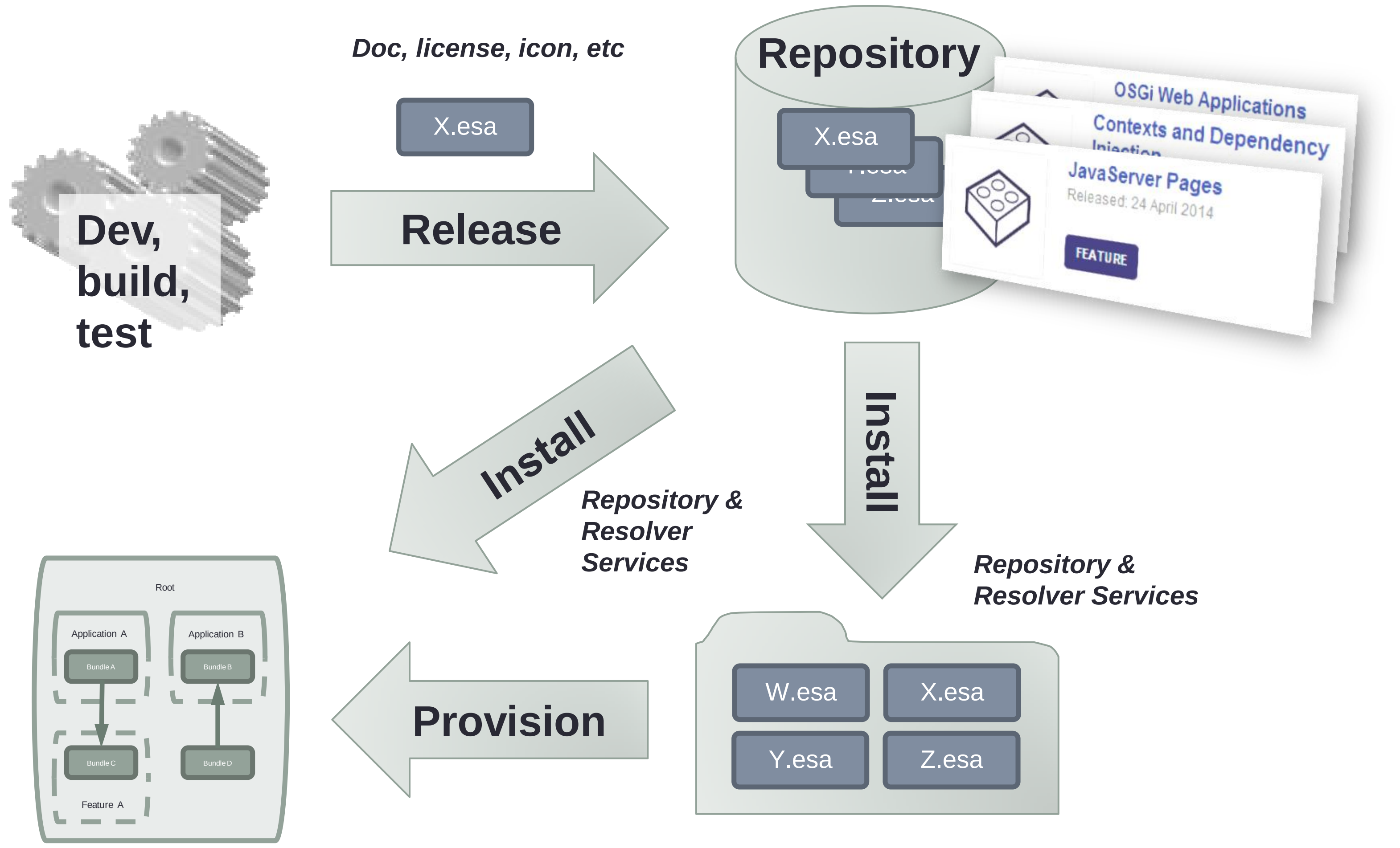
Reuse/release equivalence principle

- The unit of reuse is the unit of release
- A matter of perspective; jar, library, framework, product
- Higher levels take on ‘product’ qualities
  - Licensing
  - Documentation
  - Install
- Subsystems 1.1 adds many headers for product qualities
  - Icons, Docs, License, Localization

Use Case

- Decomposition of a product into re-usable capabilities
- Independent release of product capabilities
- Users care about capabilities, not implementation
- Users want to consume capabilities piecemeal
- ‘Stack’ products re-use subset of capabilities
- ‘Stack’ products add their own capabilities

Subsystem *Asset* Life-cycle



Demo

Summary

- The standard way to manage groups of resources
- Subsystem ‘types’ defines sharing for most common use cases
- Deployment resolution enables sharing of runtime resources
- Repositories-based resolution enables sharing of binaries
- New 1.1 features enable more ‘product’ and marketplace-like qualities

Find out more

- Reference Implementation:
  - Subsystems 1.0
    https://svn.apache.org/repos/asf/aries/trunk/subsystem/
  - Subsystems 1.1
    https://svn.apache.org/repos/asf/aries/branches/subsystemsR6/subsystem/
- New Early Draft Specification of OSGi Enterprise Release 6
  - http://www.osgi.org/Specifications/Drafts
- New OSGi Core Release 6 Specification
  - http://www.osgi.org/Specifications/HomePage
- OSGi Alliance Design Documents
  - https://github.com/osgi/design