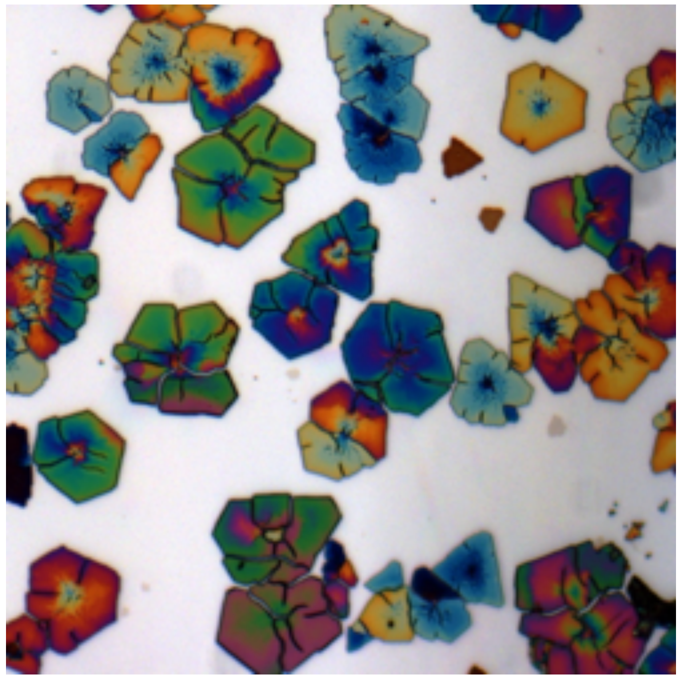
Multi-Bundle Scoping in OSGi

Glyn Normington


50 µm lead sulphide assemblies each containing ~10^8 nanoparticles

Agenda

- Modules in Java
- Multi-bundle modules
- Scoping mechanisms
- Subsystems

Modules in Java

Modularity

the degree to which a system's **components** may be **separated** and **recombined**

the **tightness of coupling** between components

the degree to which the **system architecture** controls the mixing and matching of components

WIKIPEDIA
The Free Encyclopedia

Modularity

the degree to which a system's **components** may be **separated** and **recombined**

the **tightness of coupling** between components

the degree to which the **system architecture** controls the mixing and matching of components

information hiding

Wikipedia
The Free Encyclopedia

Modules in Java

- Class
- Package
- JAR
- Class loader

} OSGi bundle

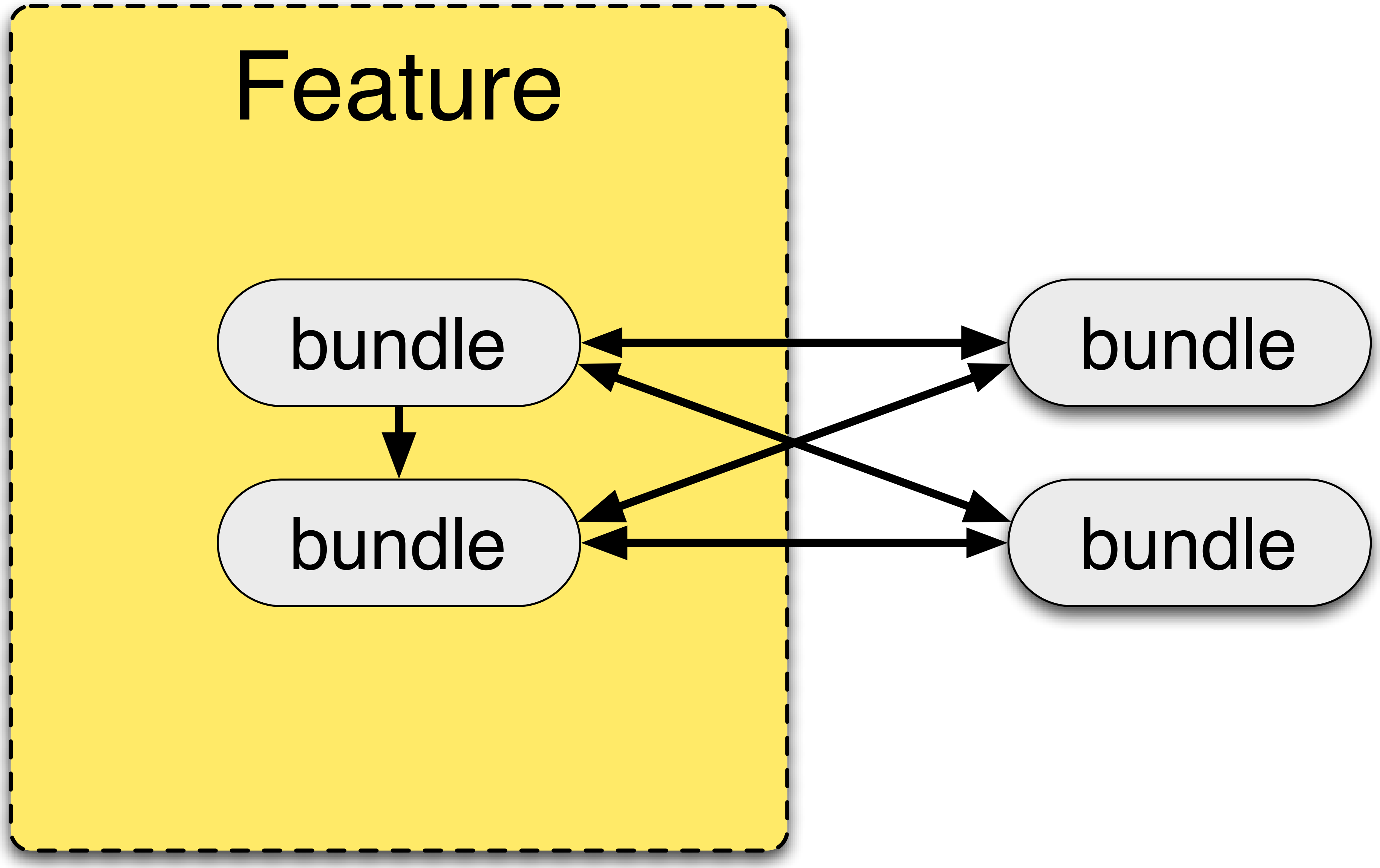
- Module - Java 8
- Larger scale modules?

Multi-Bundle Modules

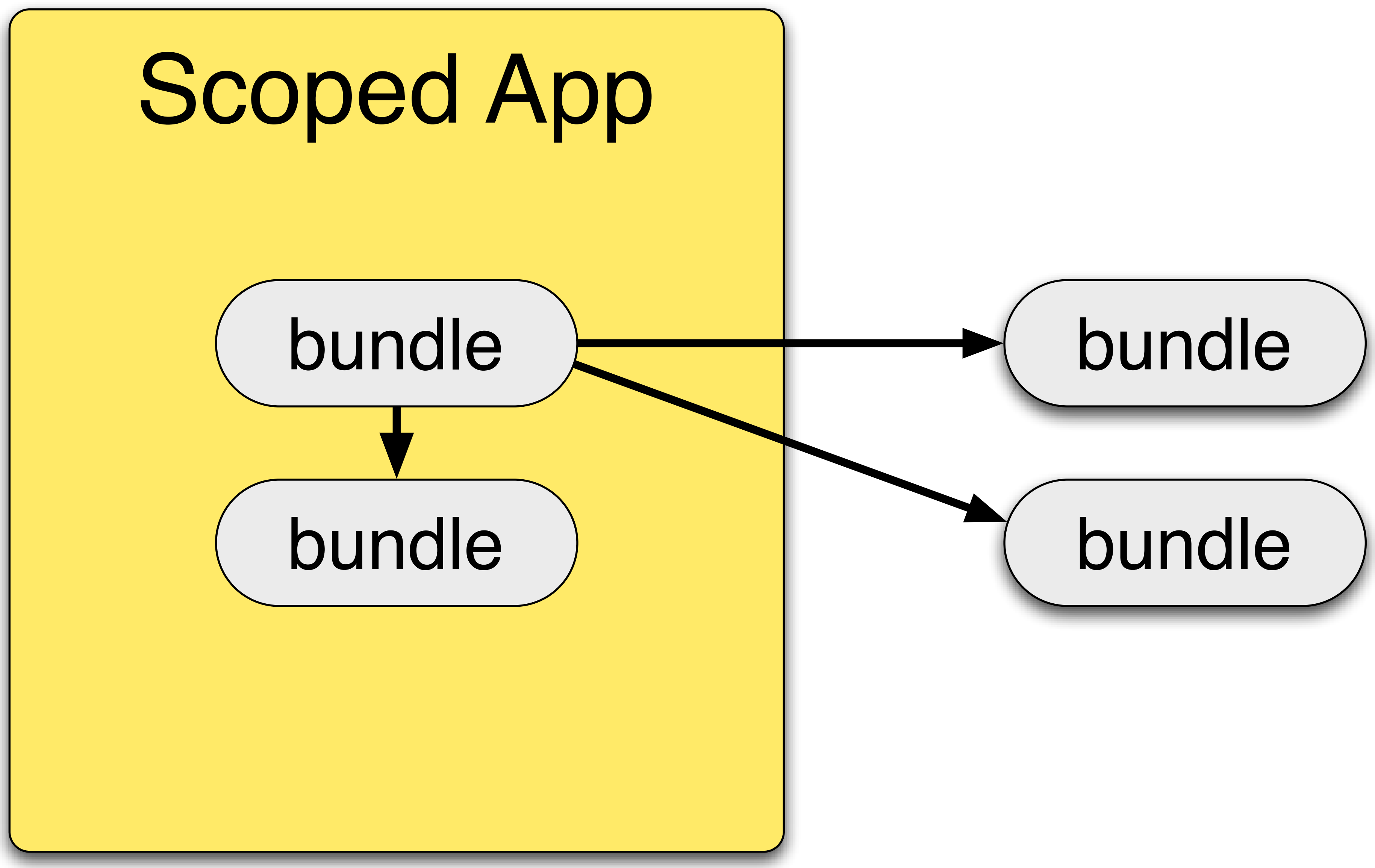
Information to Hide

- Bundle
- Package
- Generic capability
- Service

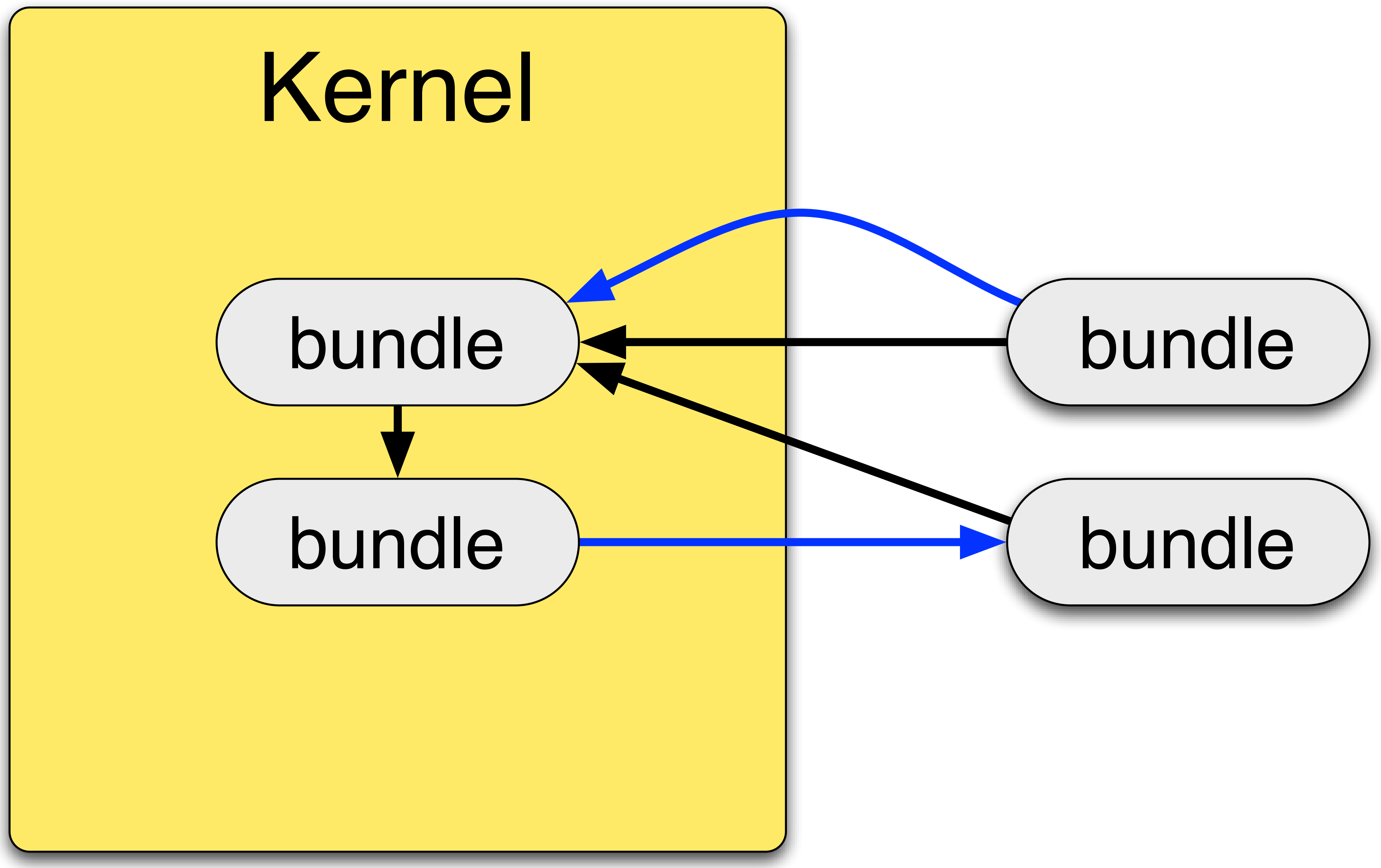
Example 1



Example 2



Example 3



Scoping Mechanisms

Scoping Mechanisms

- x-internal/x-friends
- metadata rewriting
- composite bundle
- framework hooks
- region digraph

Eclipse `x-*` Directives

- Export-Package directives
- Enforced by framework
- `x-internal`
- `x-friends`
- Non-standard
- Naming friends is brittle

➡ Standard, maintainable mechanism

Metadata Rewriting

Hide internals:

- Decorate referents
    - Bundles and packages
    - Generic capabilities
- Fix references
- Use service registry hooks

Metadata Rewriting in Virgo

Demo ...

Metadata Rewriting

Issues:

- Suitable for isolated applications only
- Intrusive
- Complicates bundle update
- Can be coded round
- Non-standard

➡ Standard scoping in the framework

Composite Bundle

- OSGi draft spec
- Prototyped in Equinox, deprecated
- Uses framework instance
- Surrogate bundle represents “parent”
- Used in Virgo 2.1 to isolate the kernel

Composite Bundle

Issues:

- Implemented in the framework
- Cannot expose constituent bundles
- Hard to spec portably

➡ Superseded

Framework Hooks

- Part of OSGi 4.3 core
- Control visibility
- Enable framework to be partitioned
- 5 main hooks:
  - resolver hook
  - bundle find/event hooks
  - service find/event hooks

Framework Hooks

Issues:

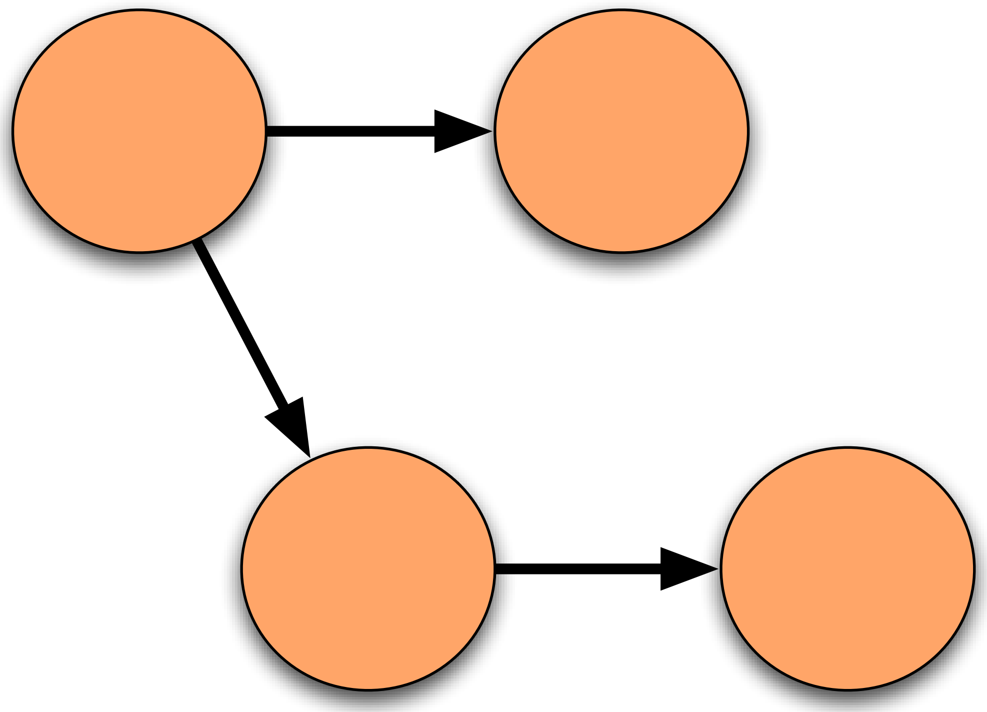
- Low level
- Consistency of hooks
- Coexistence of sets of hooks

➡ Higher level API needed

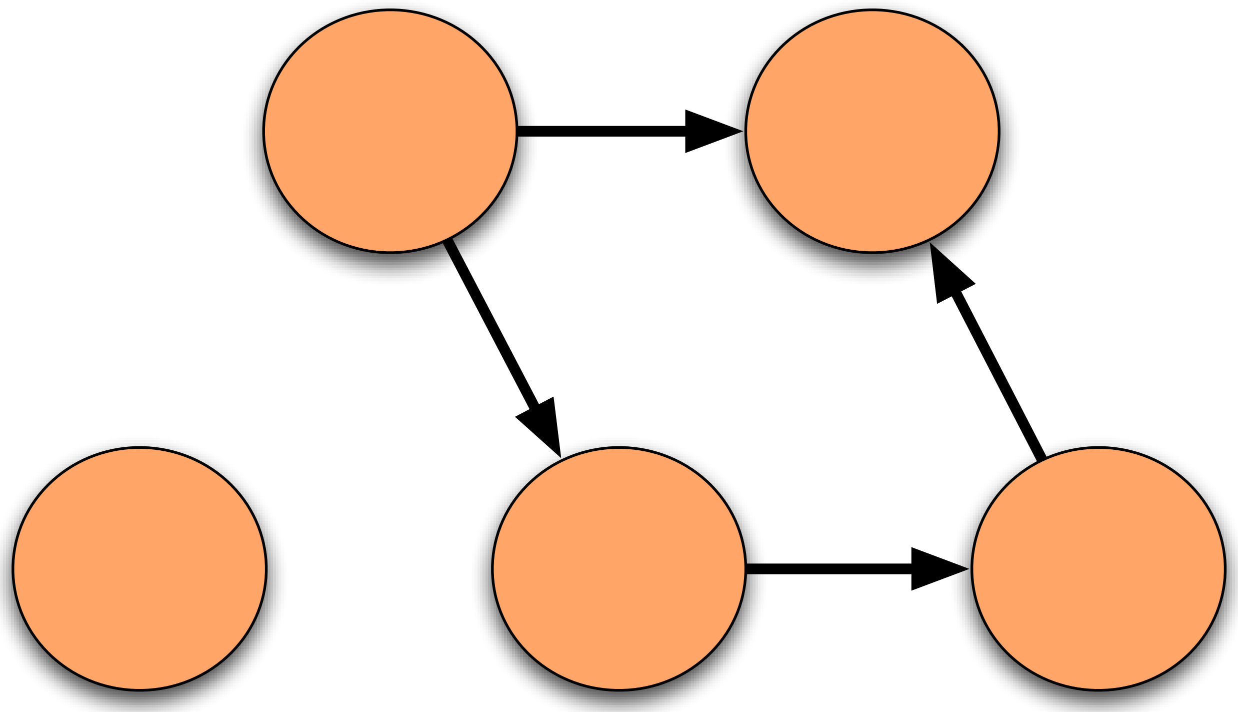
Region Digraph

Graph Theory Interlude

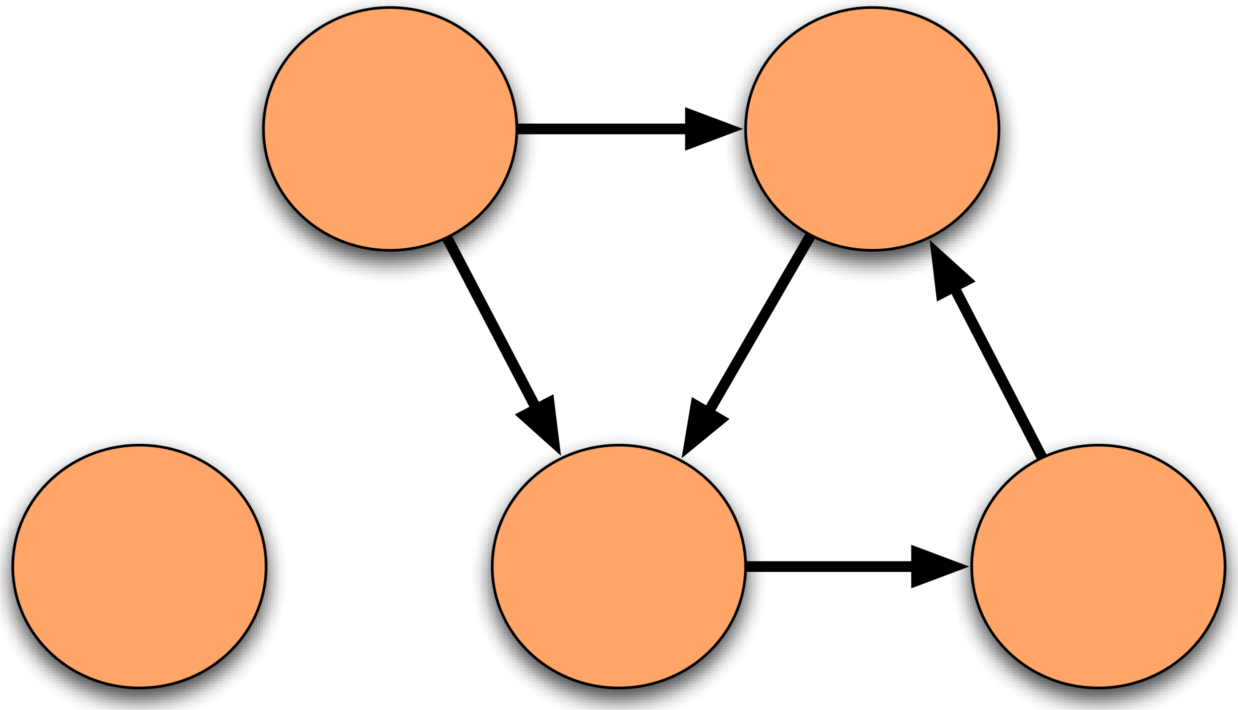
Tree



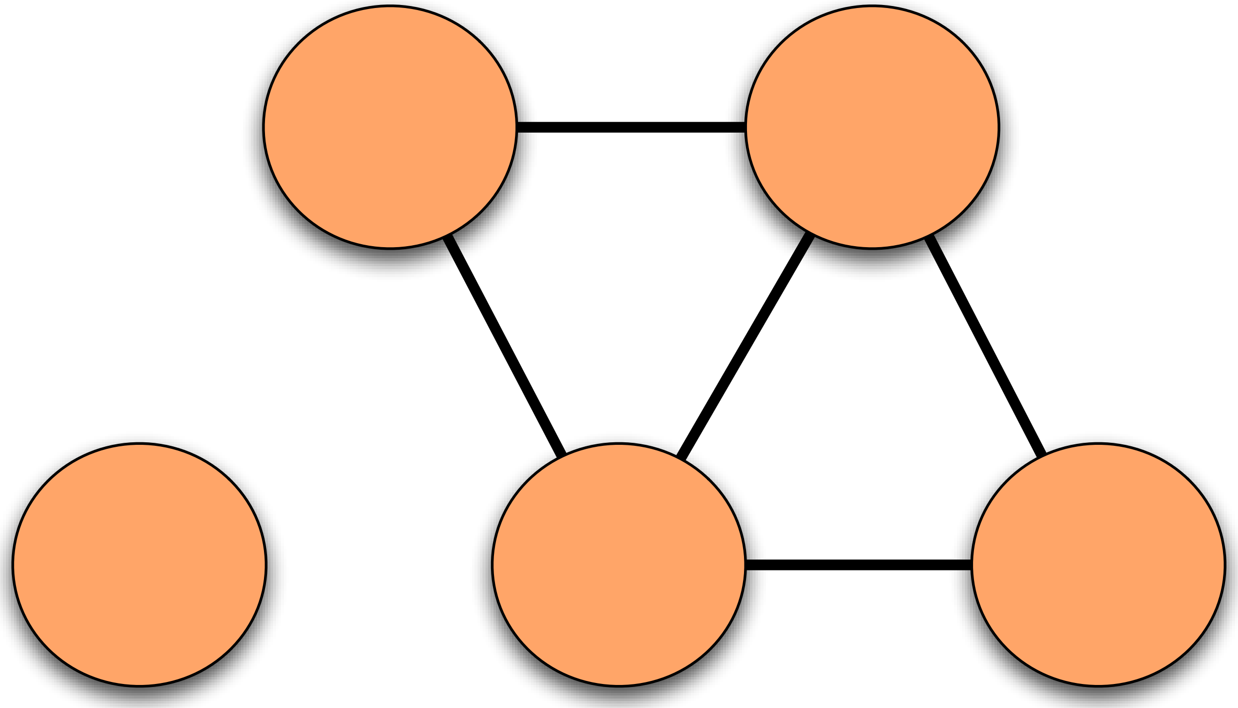
DAG



Digraph



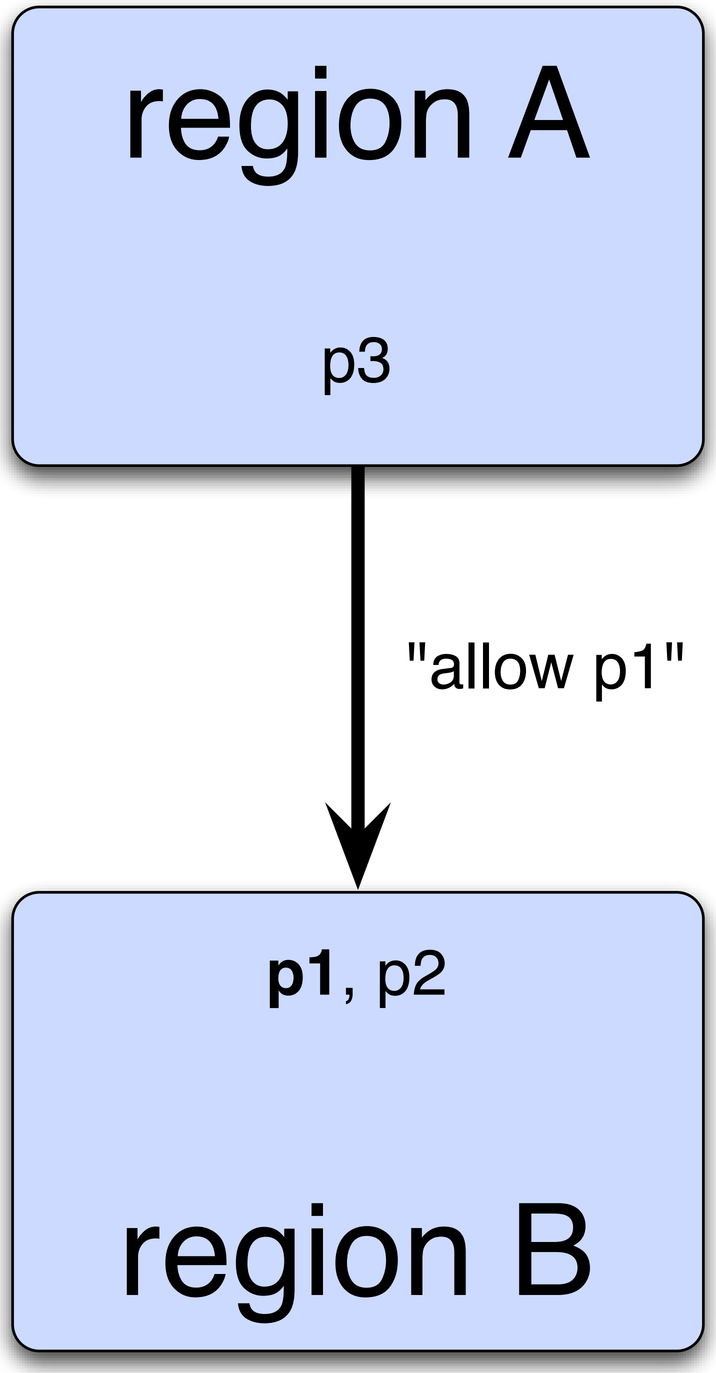
Graph



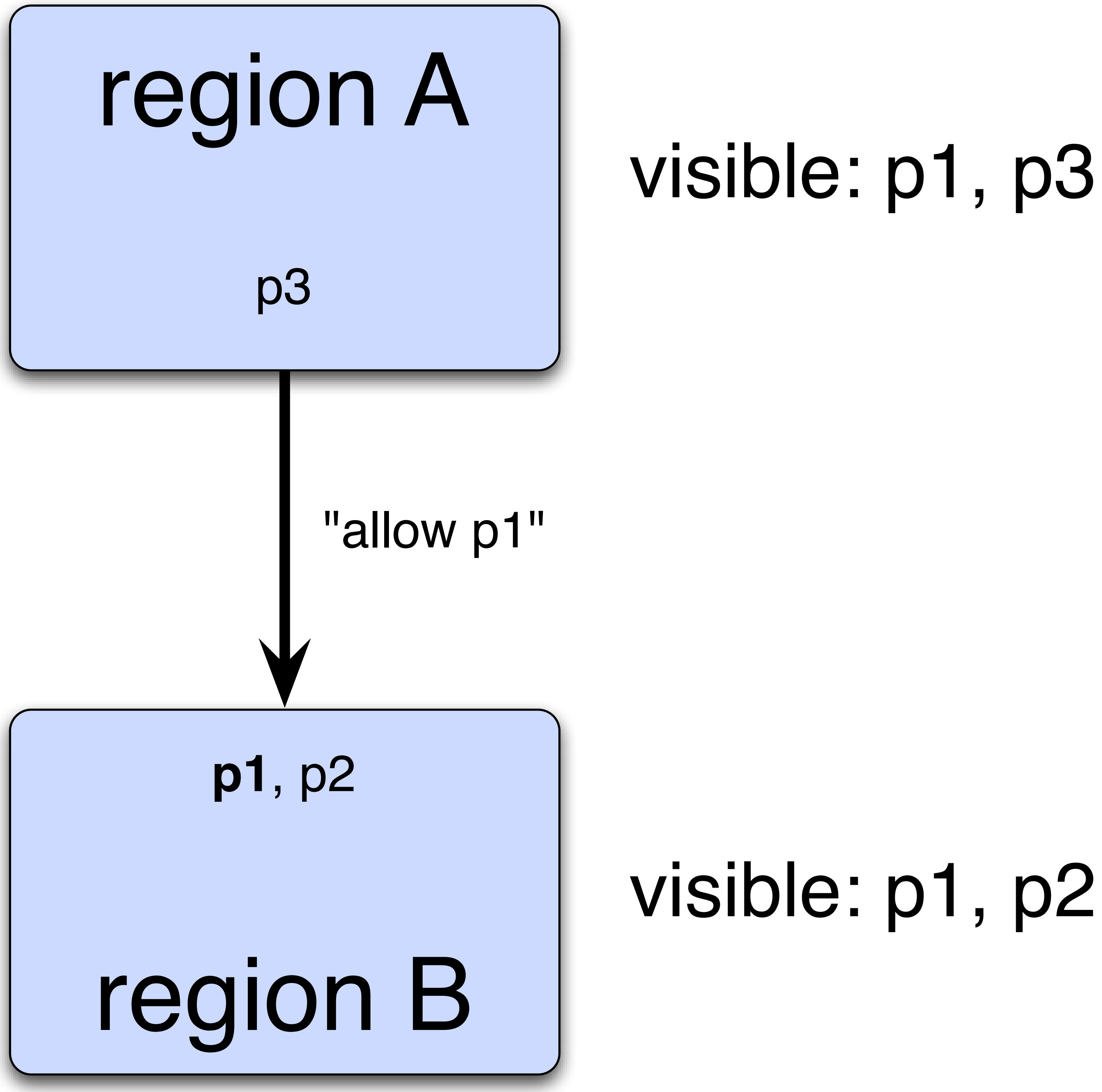
Region Digraph

- Framework hook abstraction
- Bundles partitioned into *regions*
- Regions connected by *filters*
  - Bundles
  - Packages
  - Generic capabilities
  - Services

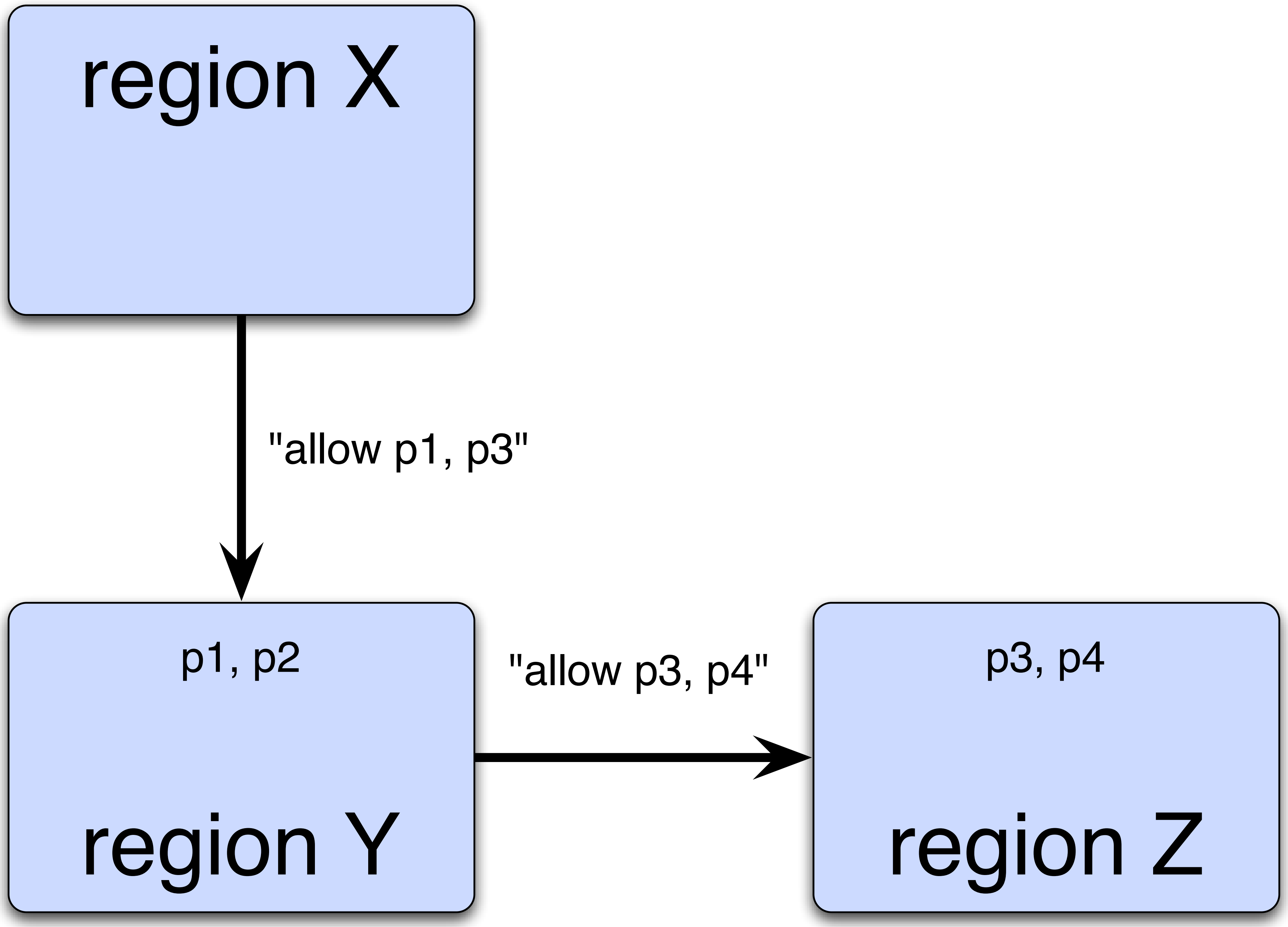
Example Digraph



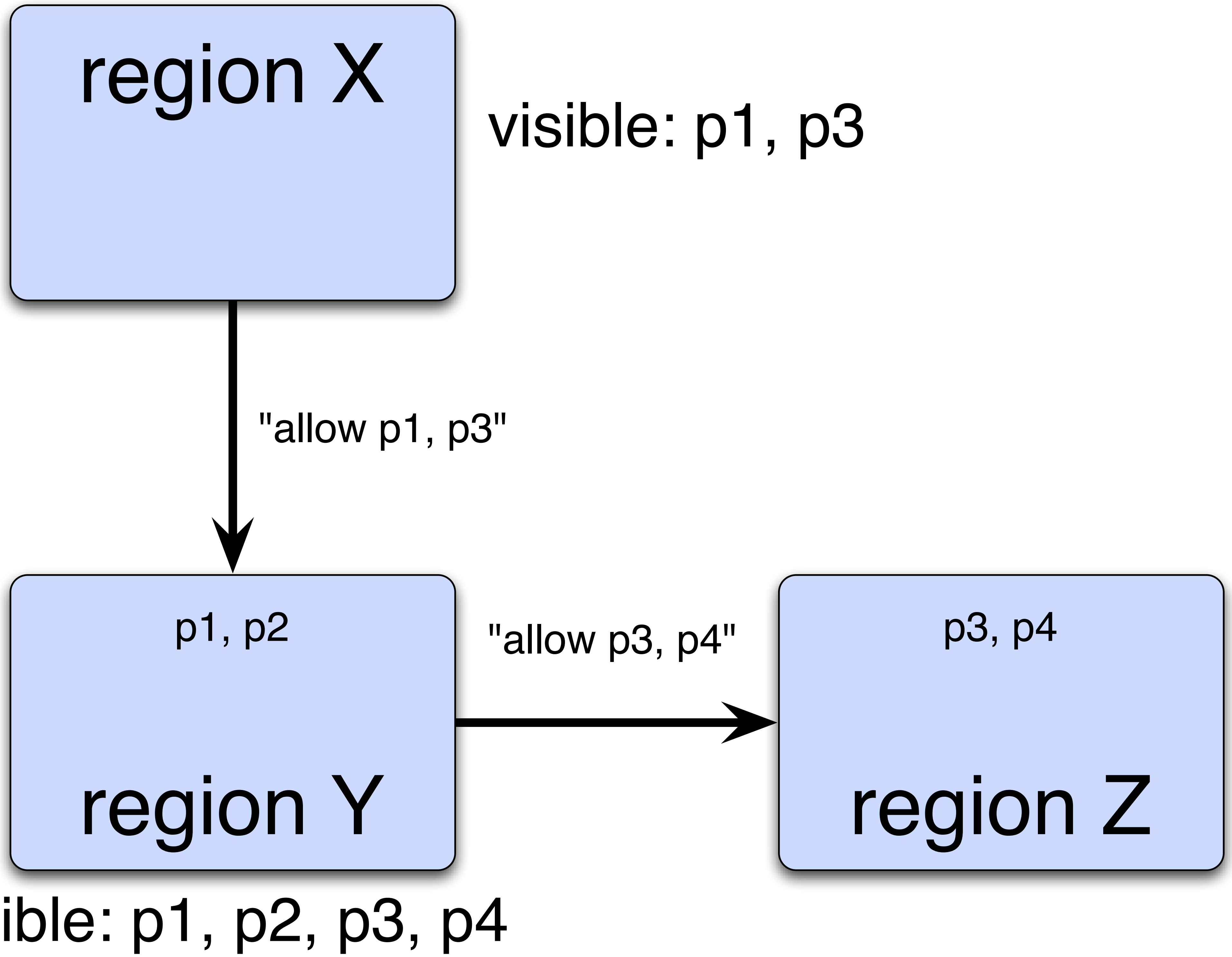
Example Digraph



Example Digraph 2



Example Digraph 2



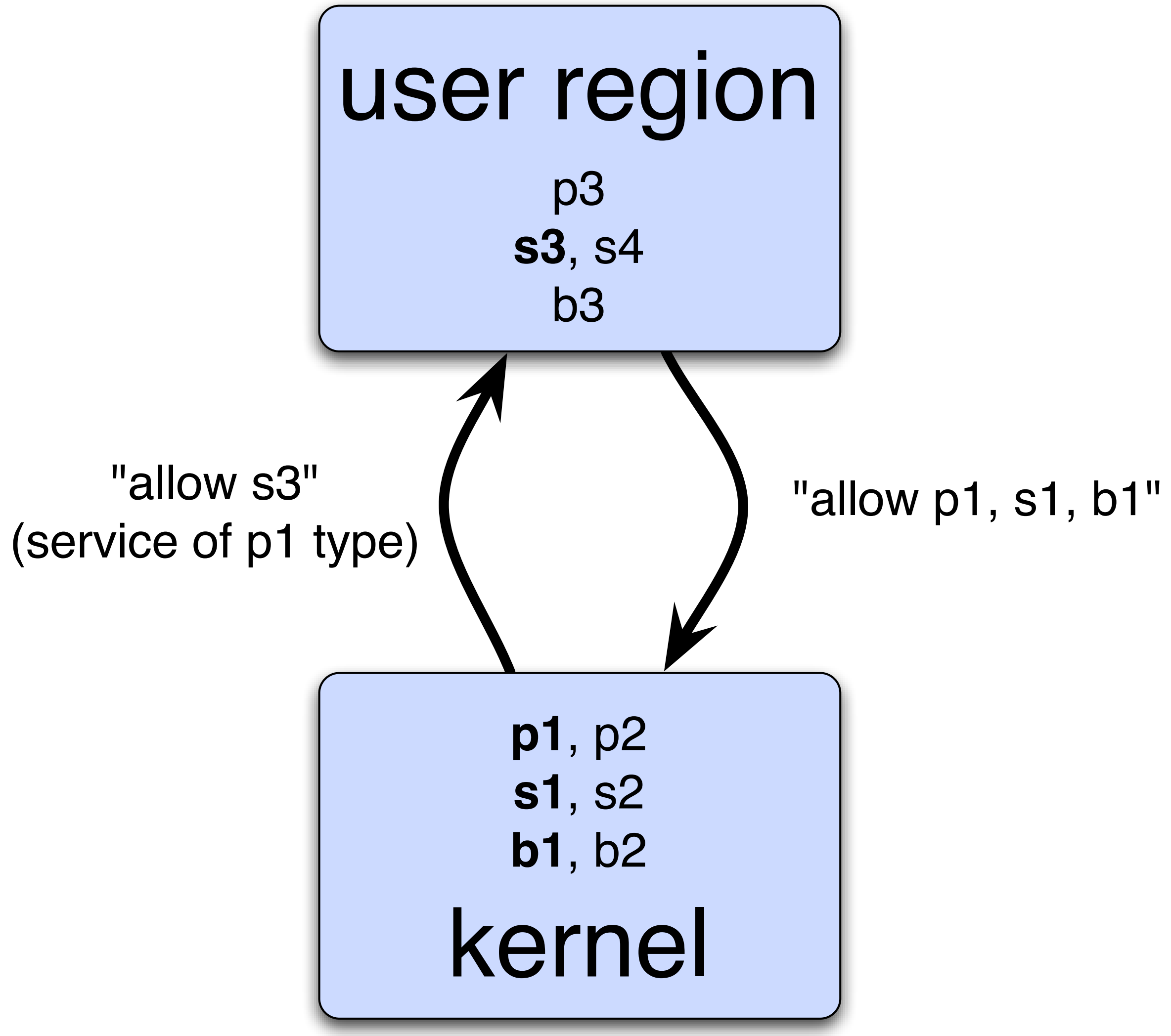
Region Digraph

- Equinox bundle
- Runs on standard framework hooks
- Persistent
- JMX

Region Digraph Uses

- Virgo 3.0 kernel/user region
- Apache Aries "subsystems" prototype
- Standalone:

> "We're using [the region digraph] essentially for a fully multi-tenant plugin system to Web applications."
>
> Robert Sauer



Region Digraph in Virgo Demo ...

Region Digraph API

```
public interface RegionDigraph ... {

   Region createRegion(String regionName) throws BundleException;

   RegionFilterBuilder createRegionFilterBuilder();

   ...
}

public interface Region {

   void connectRegion(Region headRegion, RegionFilter filter)
      throws BundleException;

   ...
}
```

Region Digraph

Issues:

- Too low level for application use
- Non-standard

➡ Standard, high-level programming model

Subsystems

Why Subsystems?

5 projects have multi-bundle constructs

- Poor portability
- Inconsistent terminology
- Diverse function

➡ Standard

Subsystems

- OSGi Enterprise Expert Group
- Scoping based on ~~composite bundles~~ framework hooks
- Public draft available ("RFC 152")

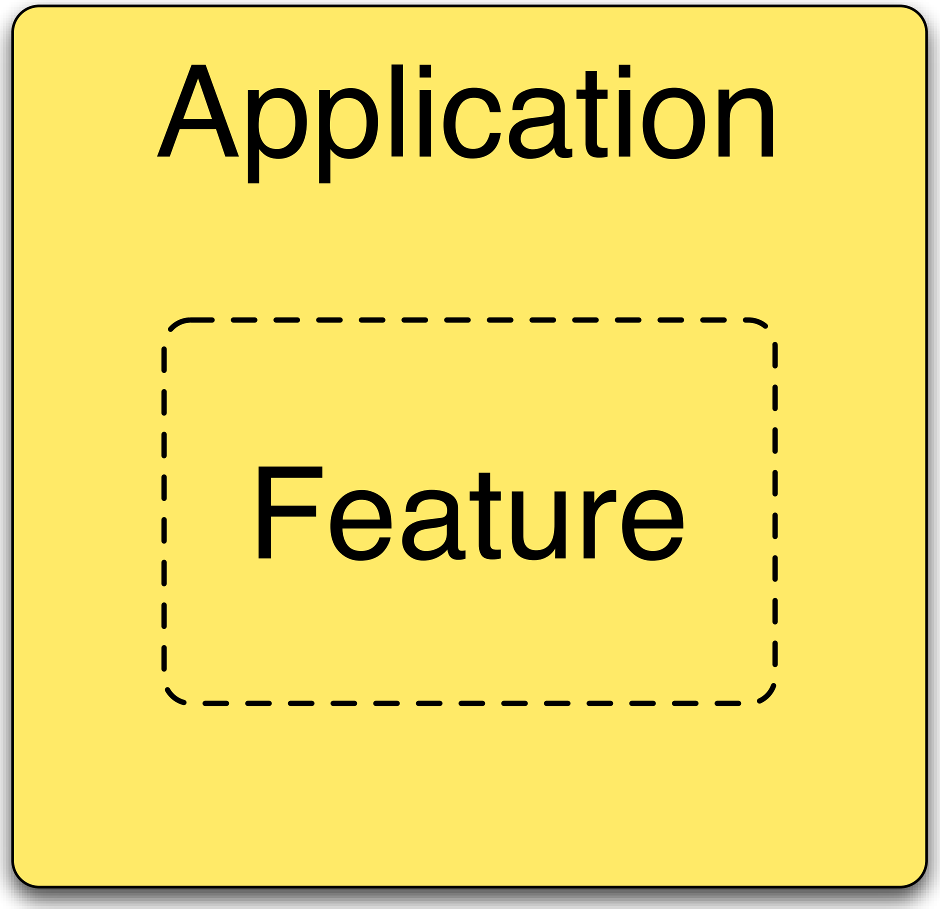
What is a Subsystem?

- Group of *constituents* with *dependencies*
  - Constituents: bundles, subsystems, ...
  - Dependencies: loosely coupled pre-reqs
- Named and versioned
- Defined lifecycle

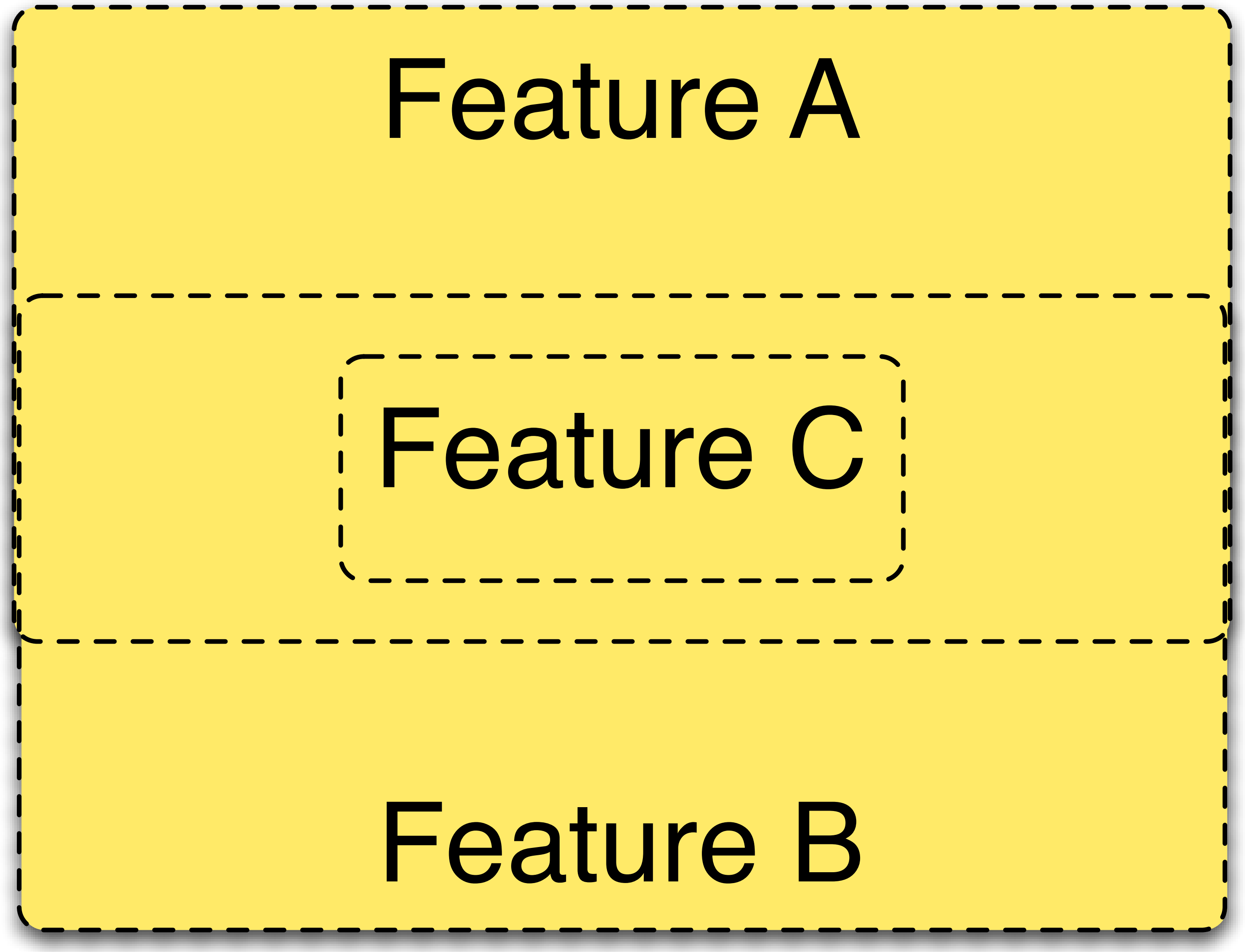
Subsystem Scoping

Subsystem type	Scoping
application	contents hidden
composite	configurable
feature	none

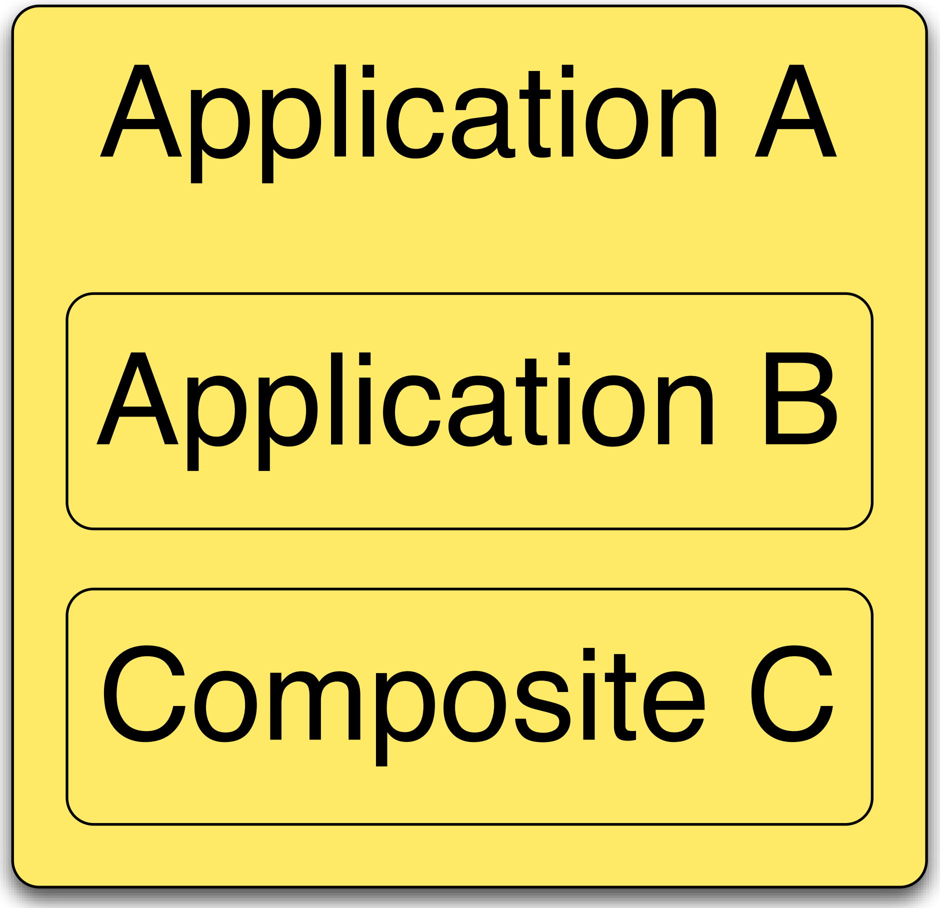
Example 1



Example 2



Example 3



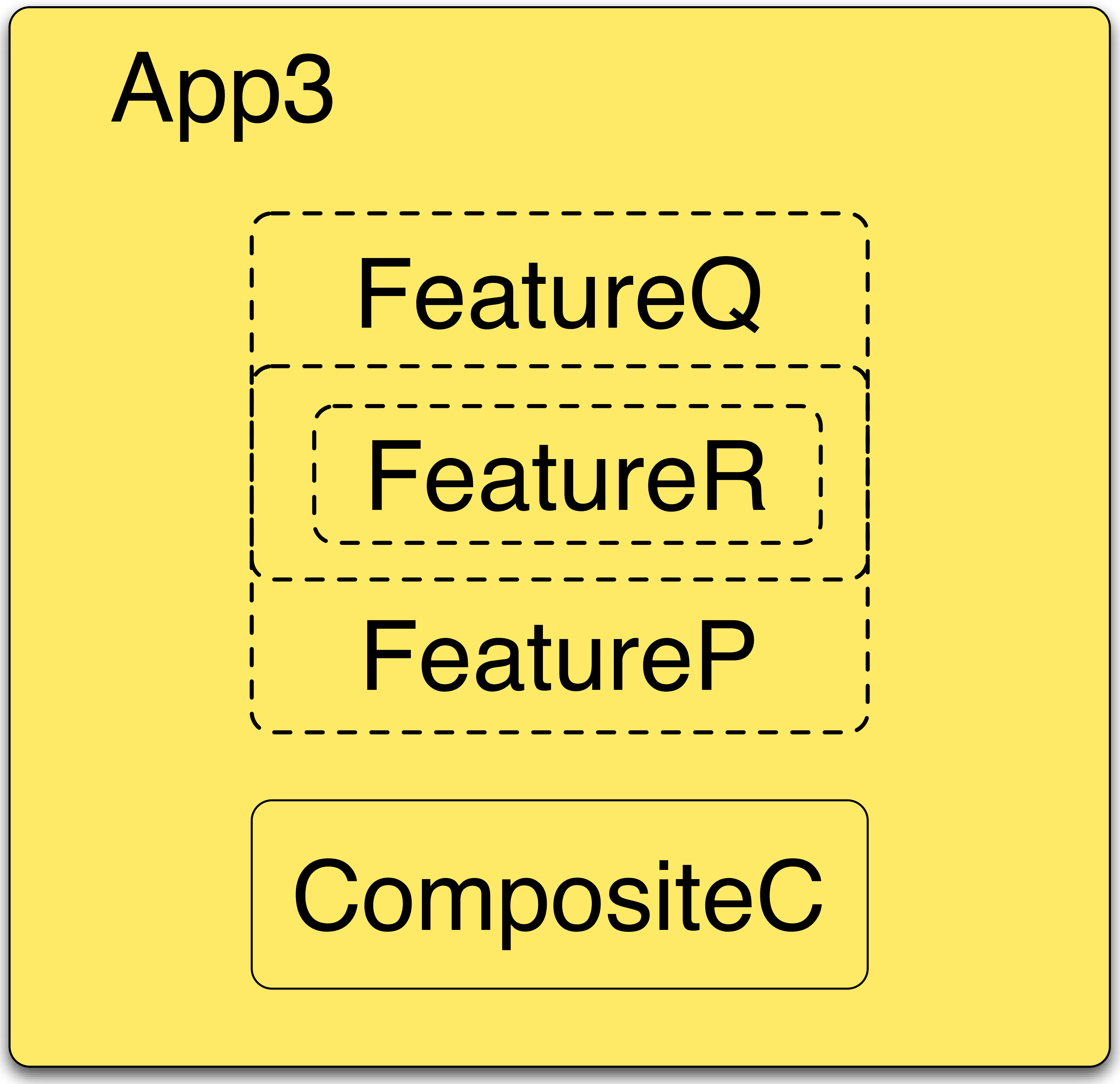
Scoping and Regions

Subsystem scoping can be

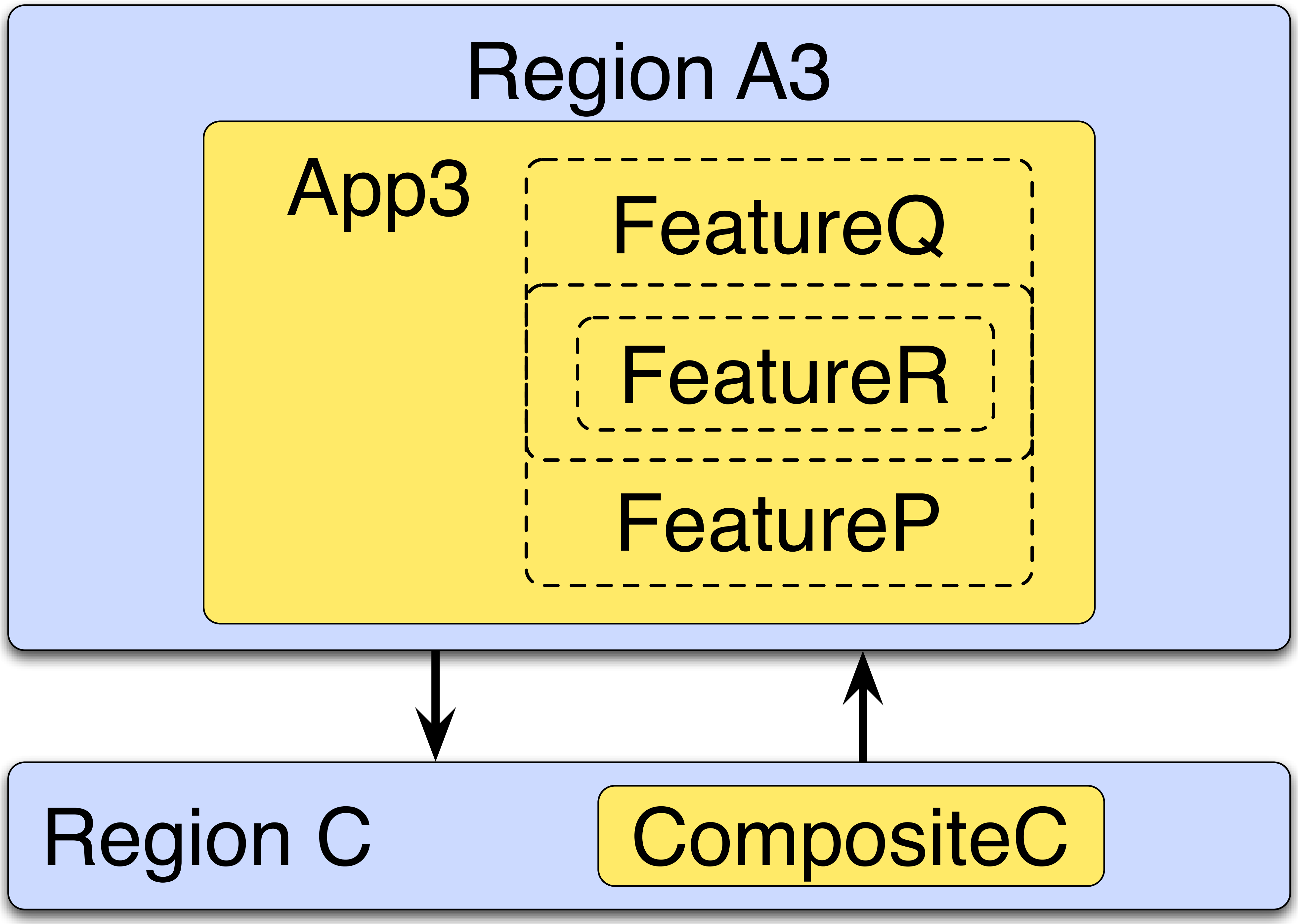
- defined in terms of
- implemented using

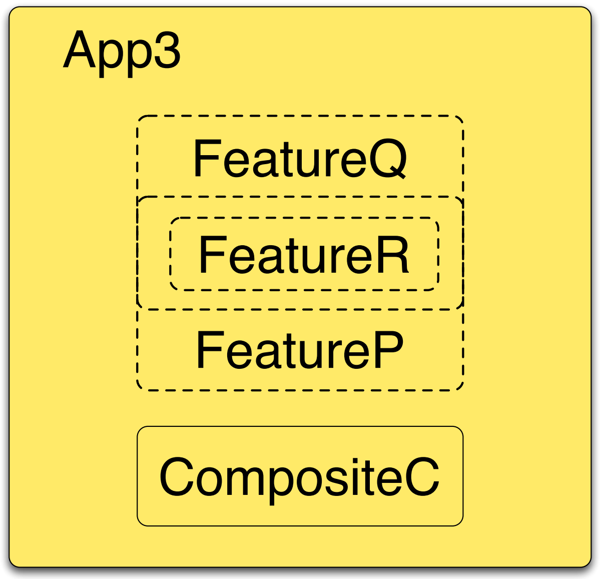
a region digraph

Complex Scoping 1

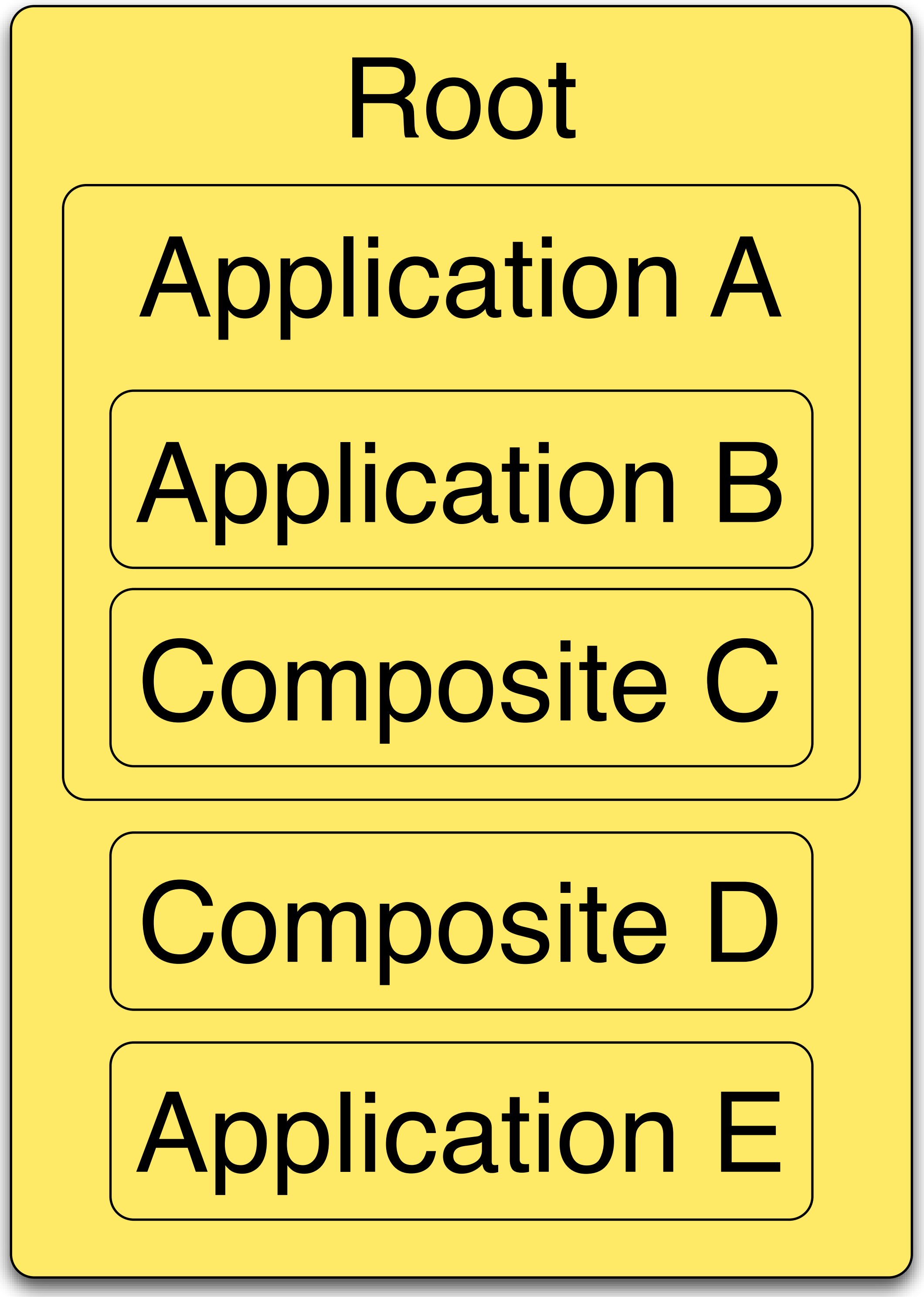


Region Digraph

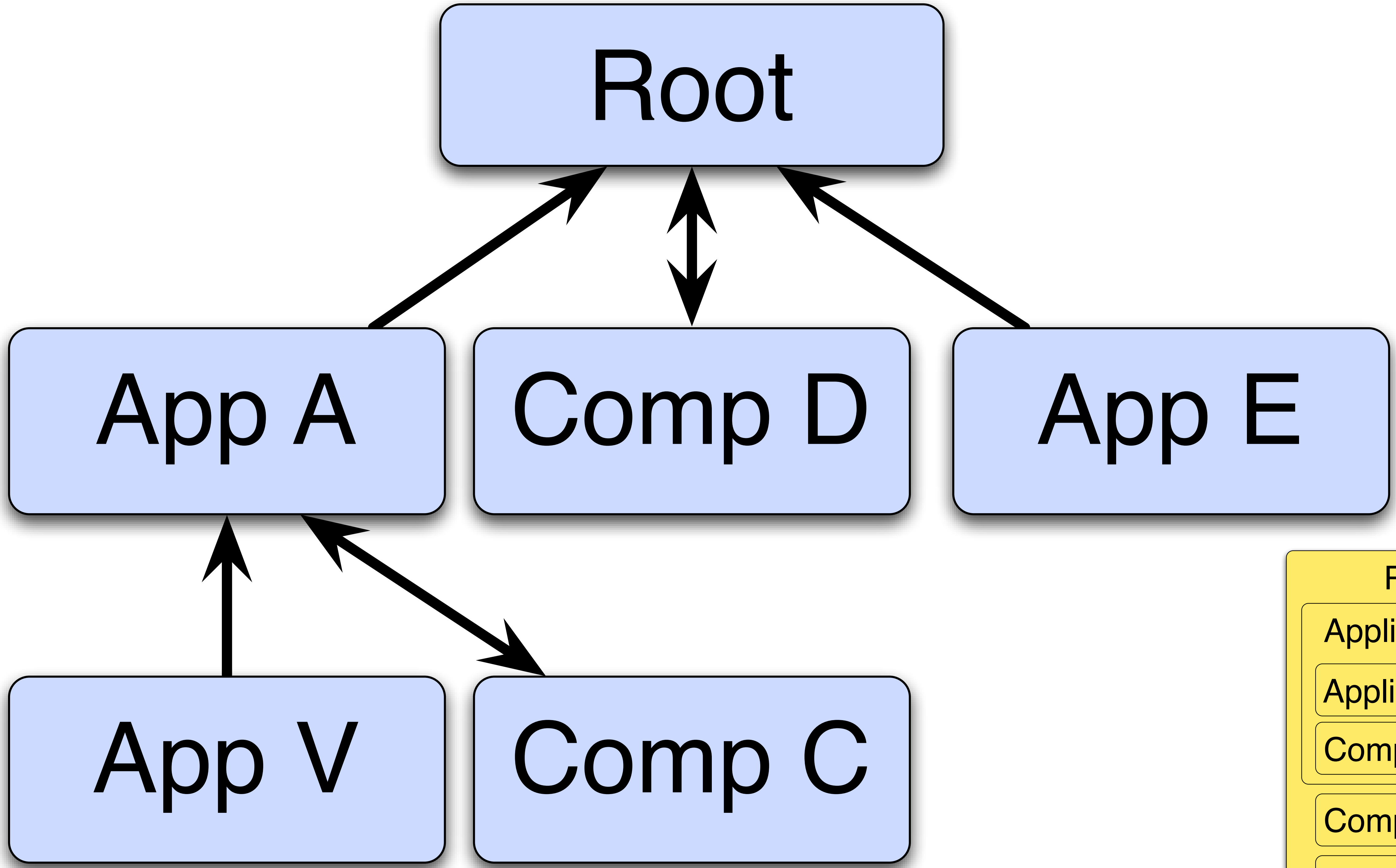


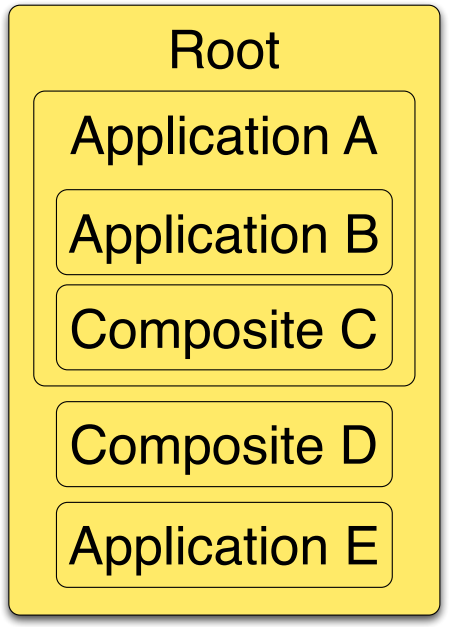


Complex Scoping 2

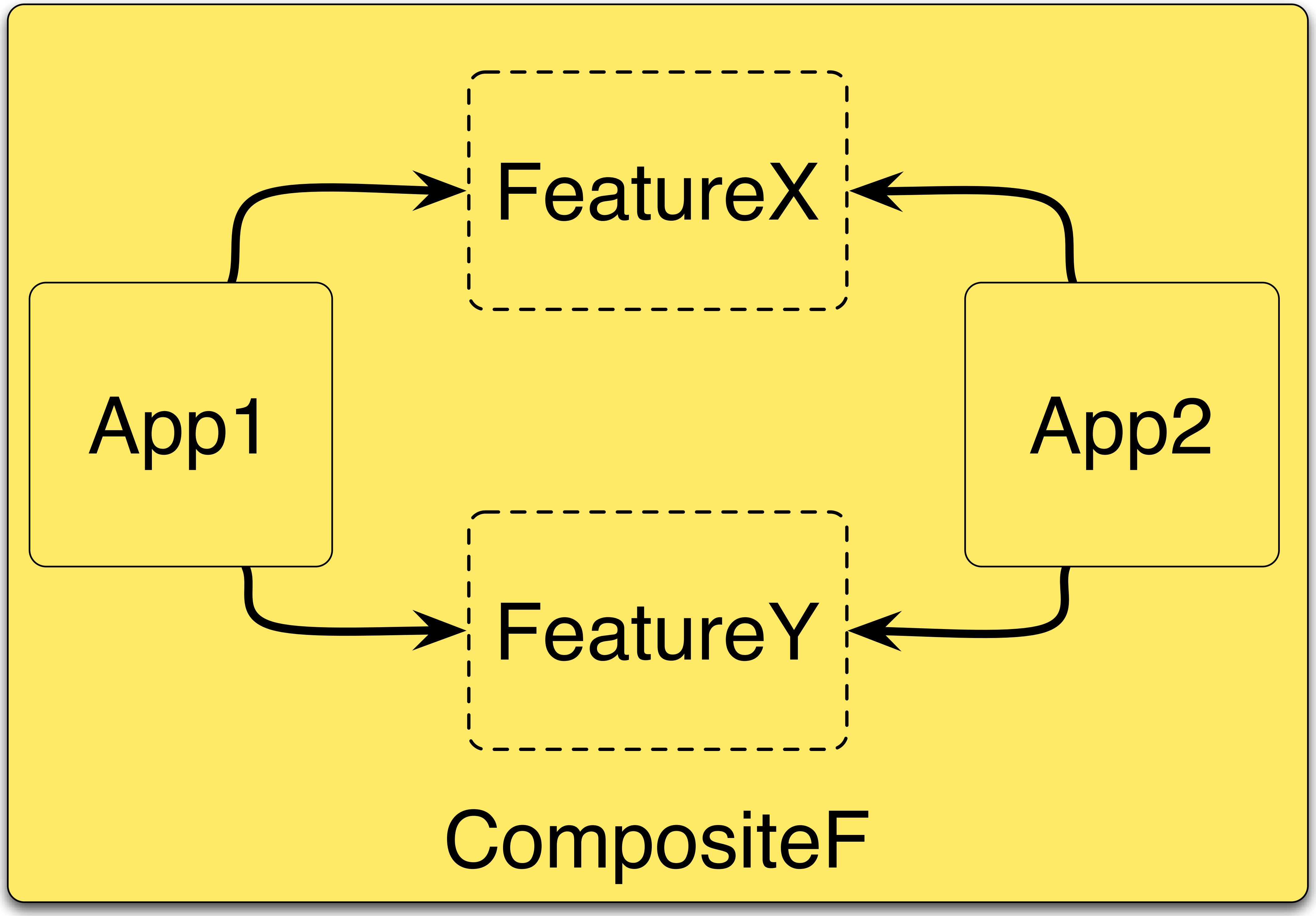


Region Digraph

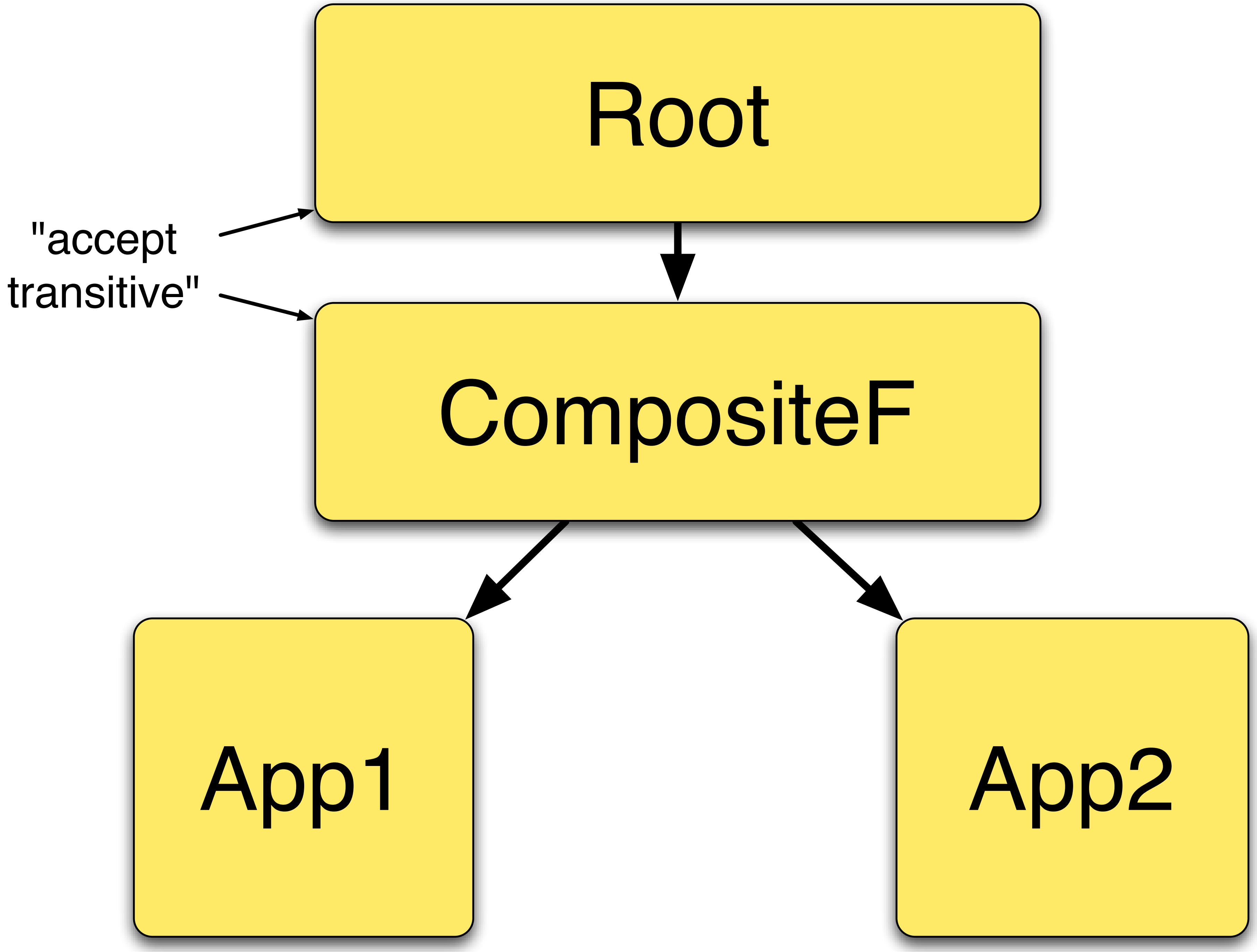




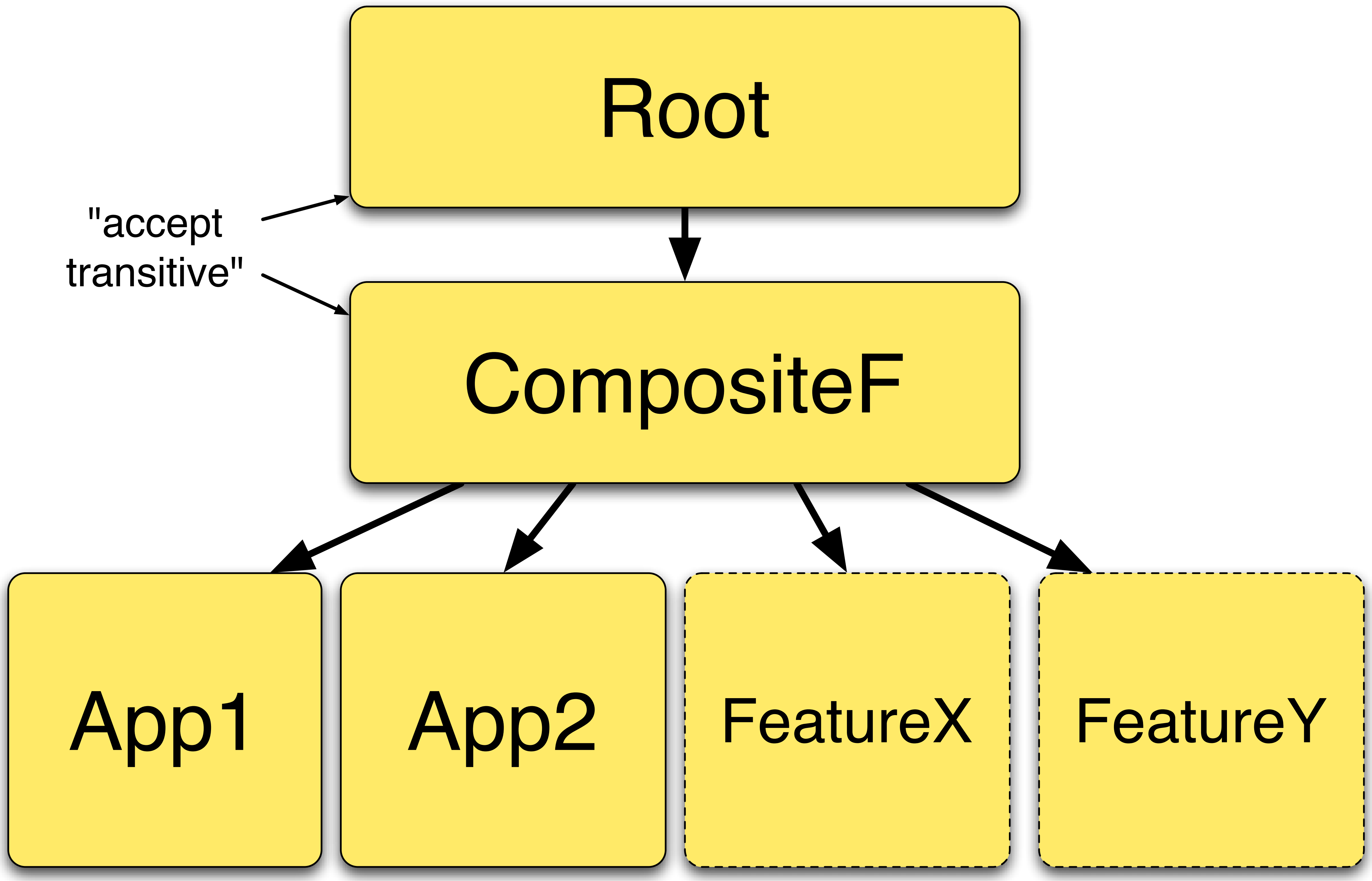
Dependencies



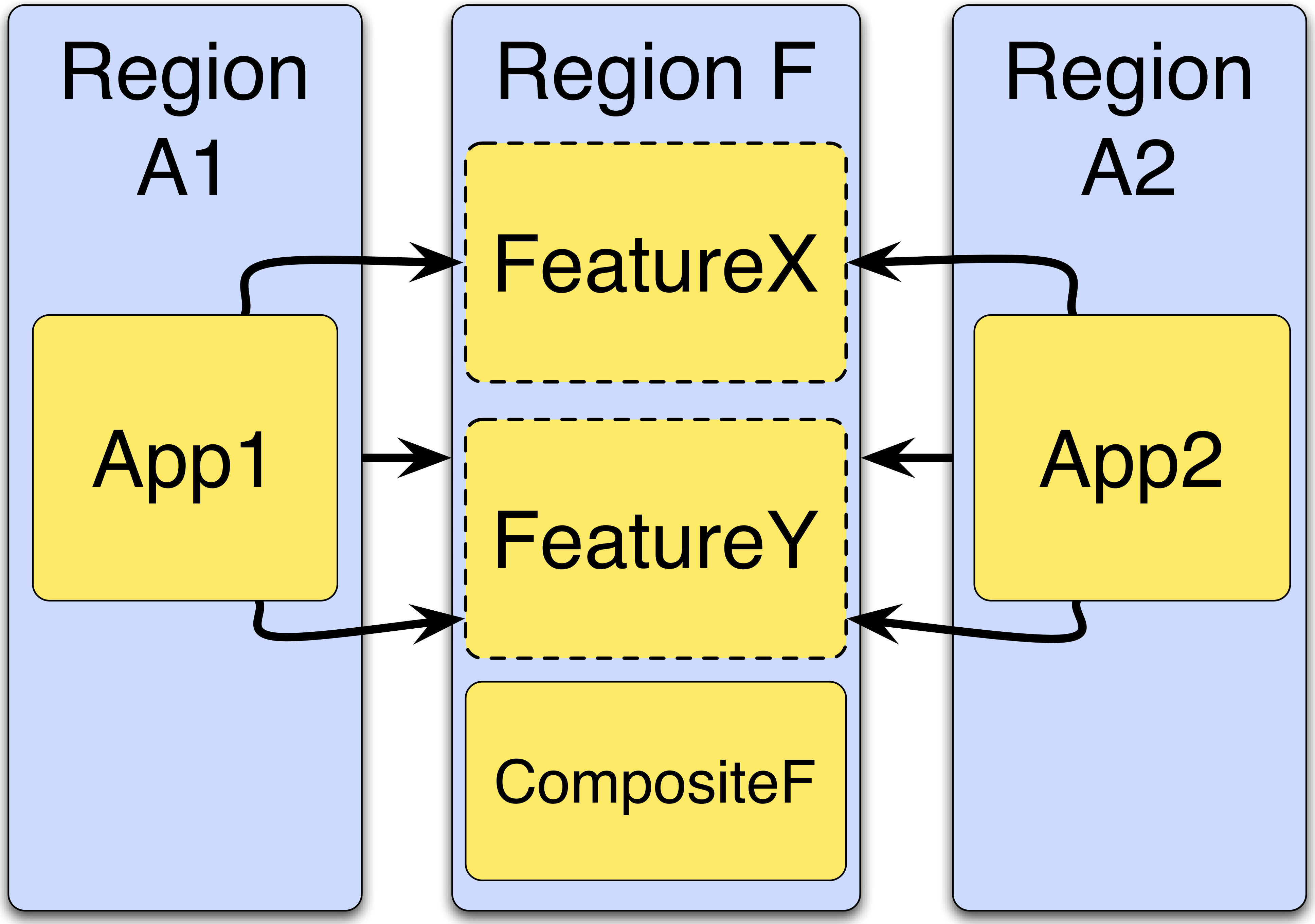
Installing Dependencies

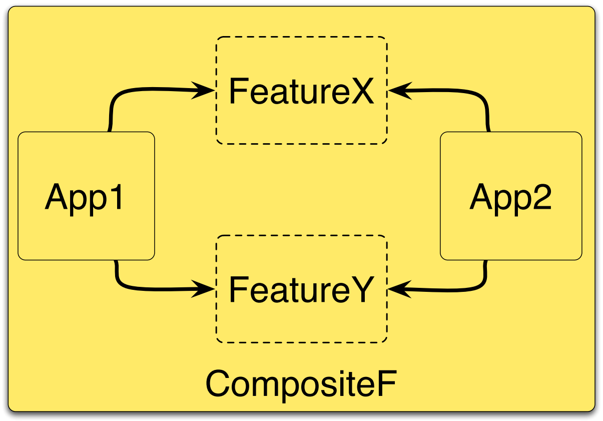


Installing Dependencies



Region Digraph





Conclusion

Modularity in Java:

- Class, package, JAR, class loader
- Modules in Java 8
- Larger modules?

Modularity in OSGi:

- Bundles
- Multi-bundle modules
- Subsystems

Further Information

- Public draft of subsystems (RFC 152)
  - http://www.osgi.org/Download/File?url=/download/osgi-early-draft-2011-09.pdf
- Region digraph
  - http://underlap.blogspot.com/2011/05/equinox-digraph-ready-for-use.html
- Eclipse Virgo and the region digraph
  - https://bugs.eclipse.org/bugs/show_bug.cgi?id=330776
- Apache Aries use of region digraph
  - https://issues.apache.org/jira/browse/ARIES-644
- Multi-module modules and Java 8
  - http://mail.openjdk.java.net/pipermail/jigsaw-dev/2011-October/001564.html

Thanks to...

- Wikipedia image courtesy of Wikipedia.org
- Field of Flowers by Paul Podsiadlo and Elena Shevchenko, courtesy of Argonne National Laboratory