Robusta : An approach to building dynamic applications

Walter RUDAMETKIN

Dissertation defense
21 February 2013

Pr. Noel DE PALMA	President	UJF, Grenoble
Pr. Luciano BARESI	Reviewer	Politecnico di Milano
Dr. Benoit BAUDRY	Reviewer	INRIA, Rennes
Dr. François EXERTIER	Examiner	Bull, S.A.S
Pr. Eric GRESSIER-SOUDAN	Examiner	CNAM-CEDRIC
Dr. Jacky ESTUBLIER	Supervisor	CNRS, Grenoble

Outline

Devices, Mobility & Services

Business

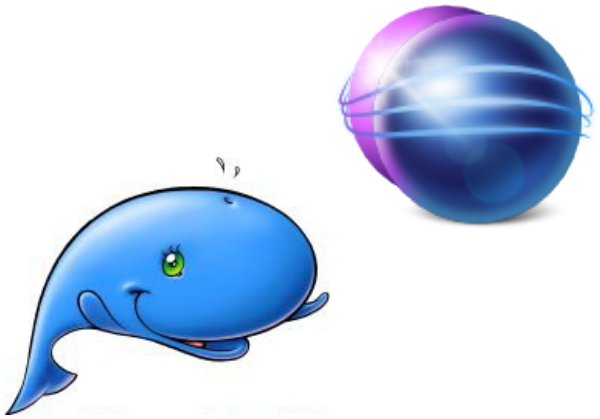
Home Automation

Entertainment

Large apps

Application Servers

Plugin based applications



Context-aware apps

Mobility and change

Device-oriented

These applications need

Patches & Updates

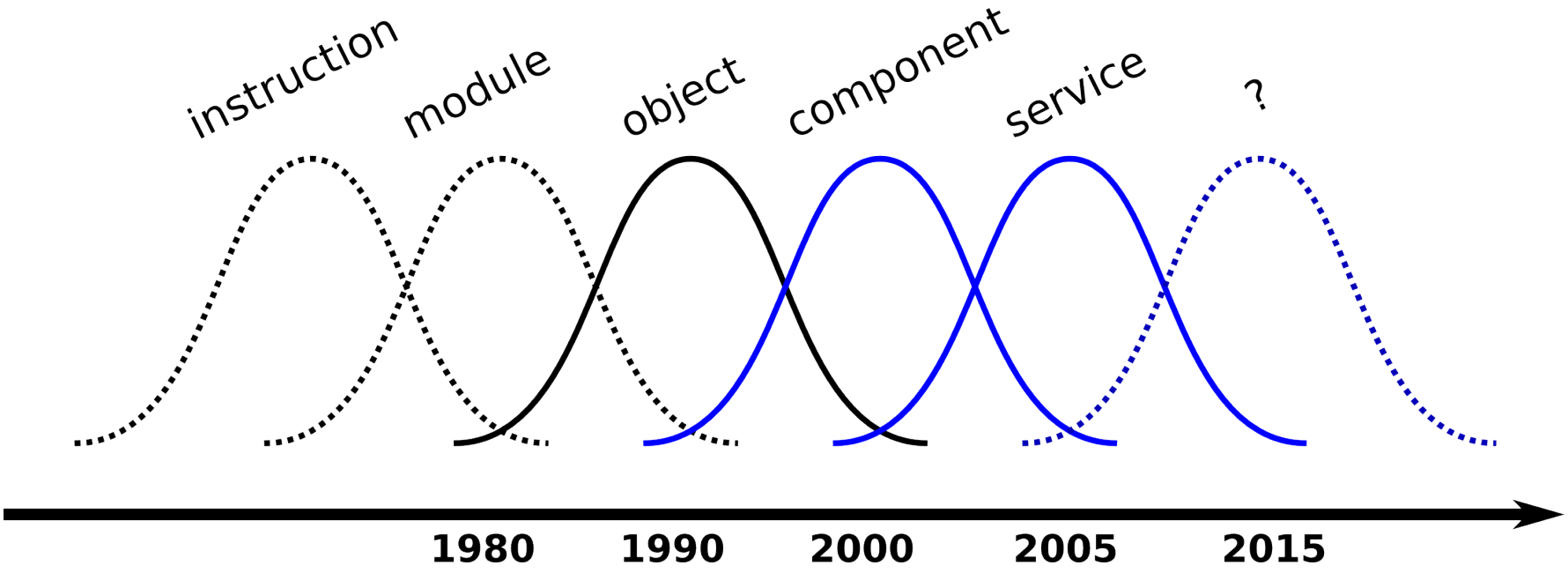
New features

These applications need

Minimal Downtime

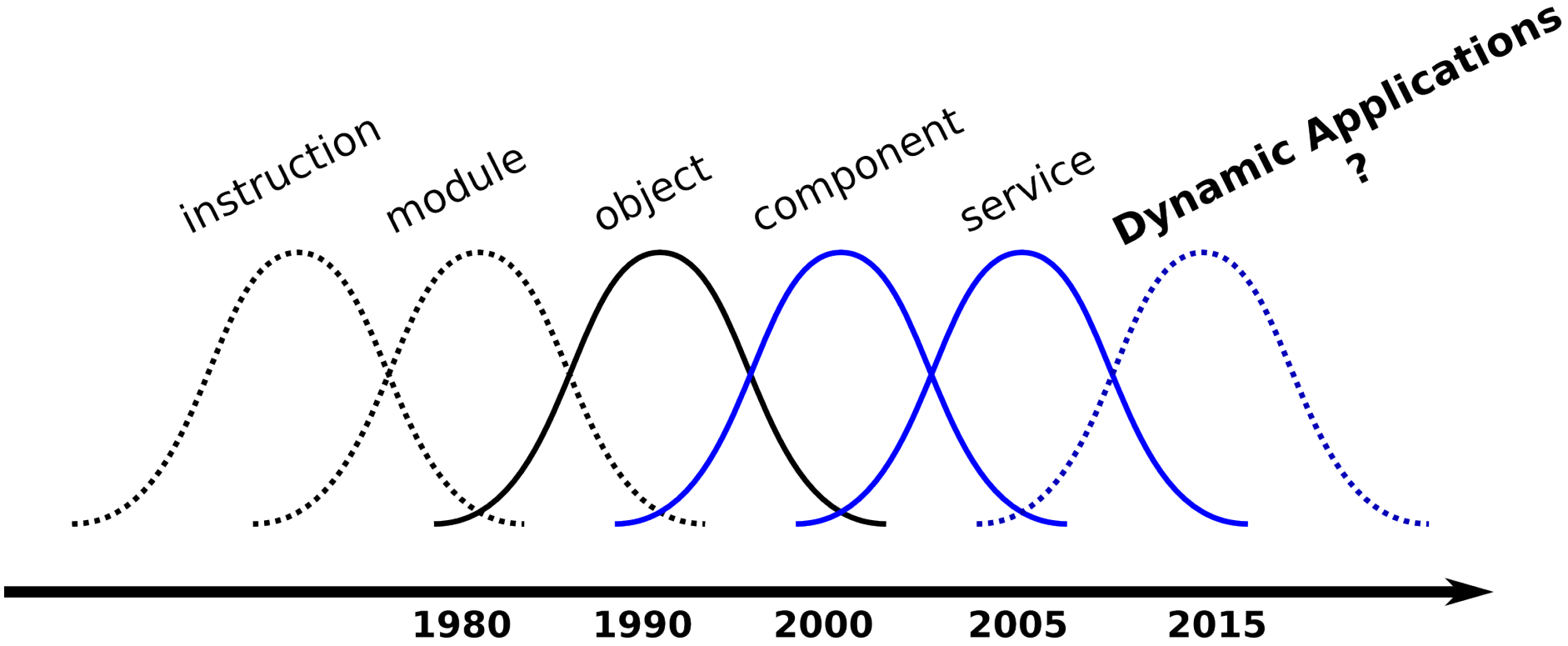
Reactivity

Complexity, structure, decoupling



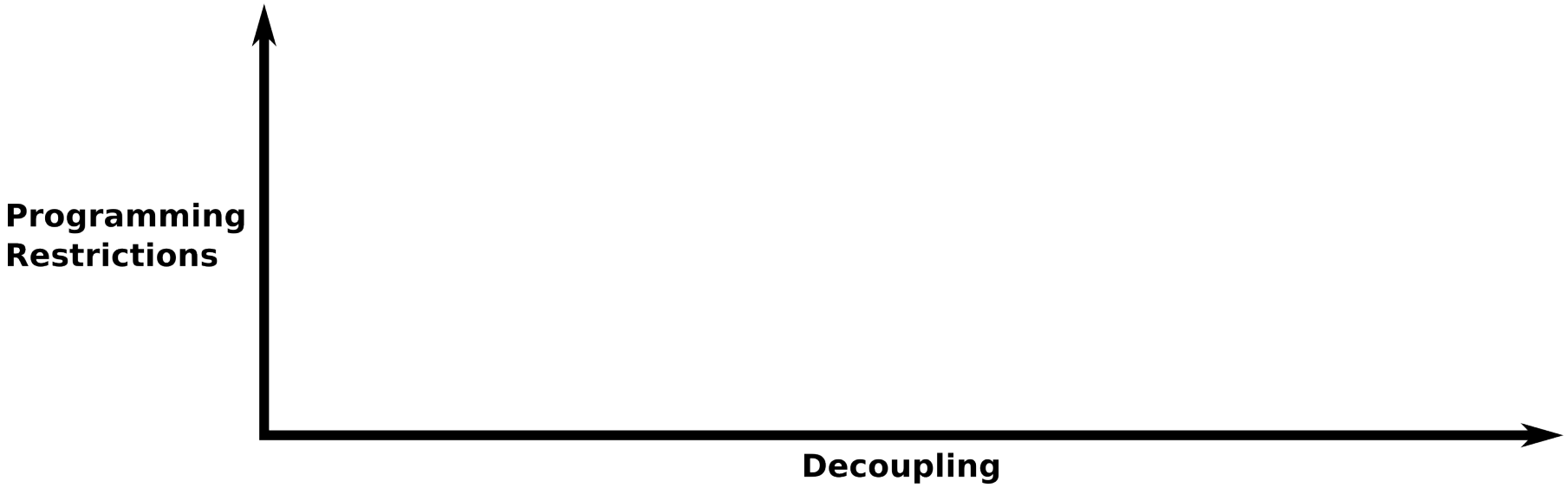
Raccoon [1997] Revisited

Complexity, structure, decoupling

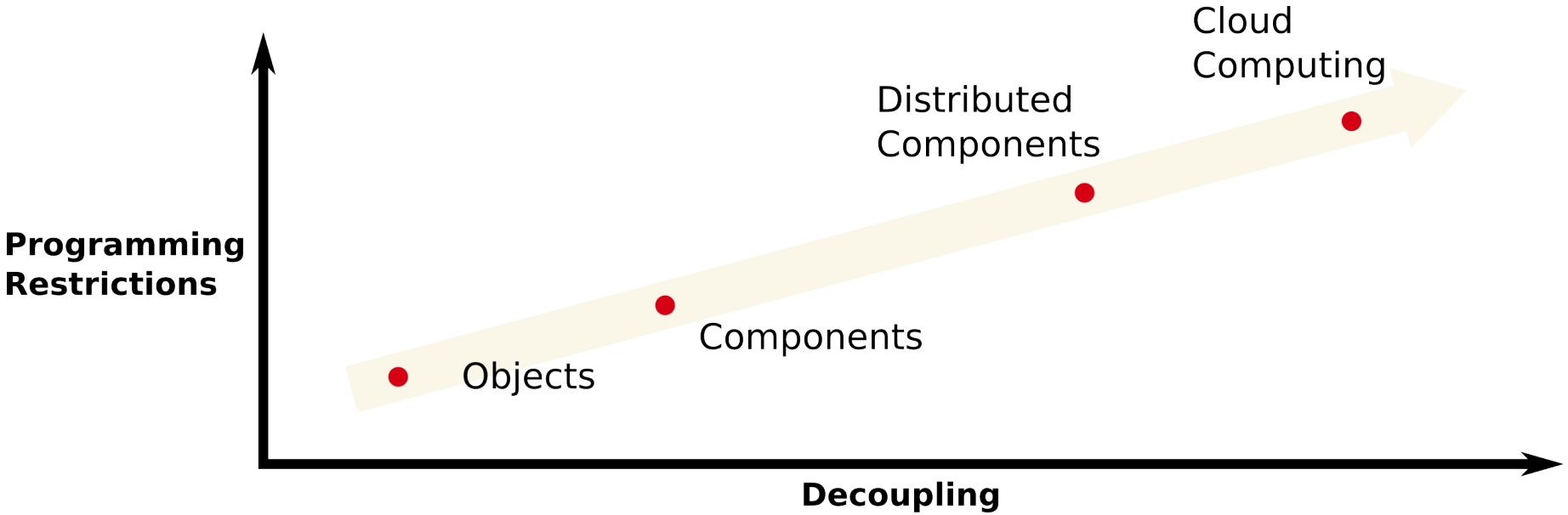


Raccoon [1997] Revisited

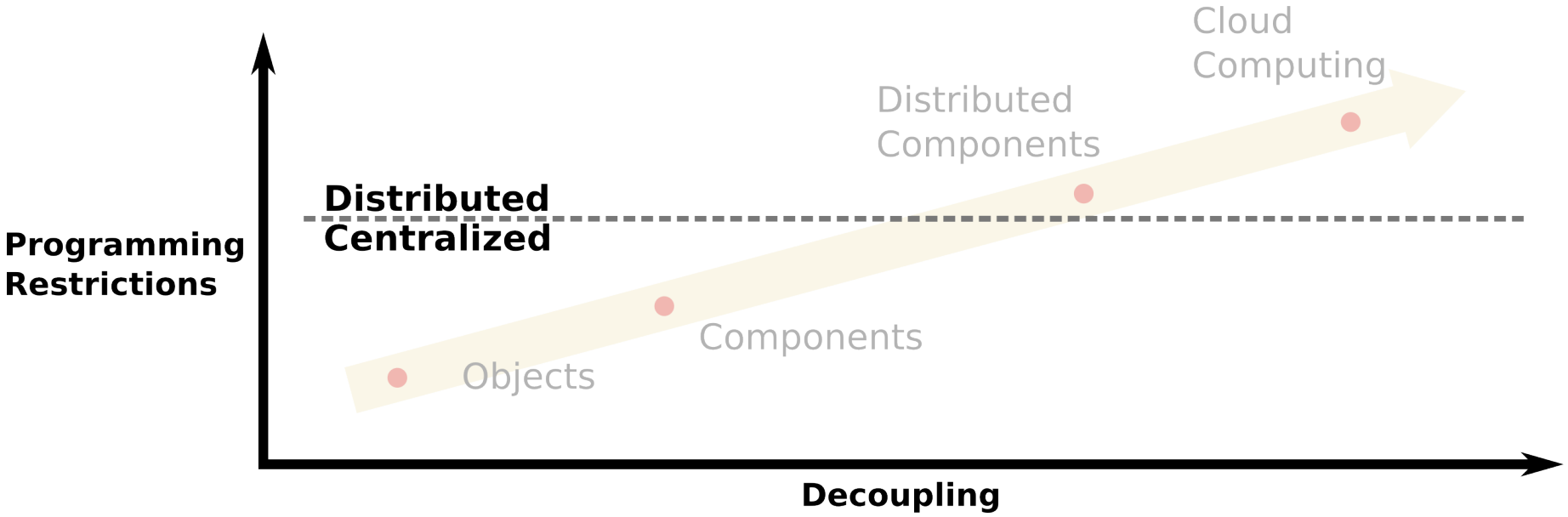
Restrictions vs. decoupling



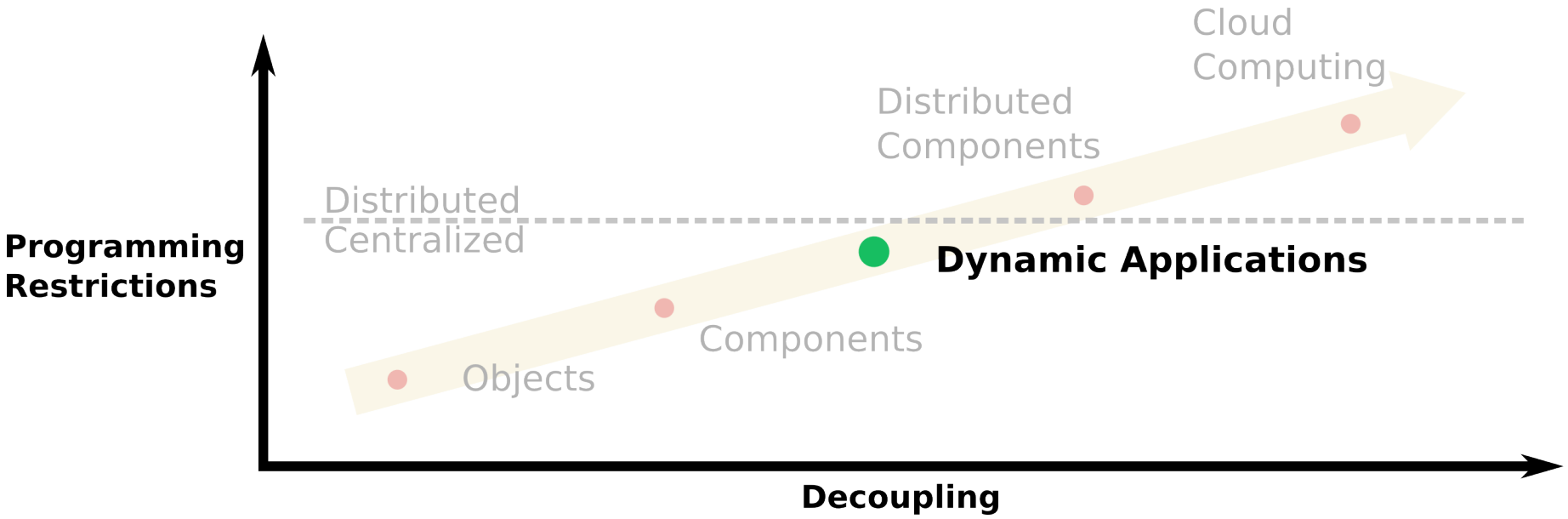
Restrictions vs. decoupling



Restrictions vs. decoupling



Restrictions vs. decoupling



We address

Centralized

Multi-threaded

Component-based

Object Oriented implementations

Dynamic Applications

To build dynamic applications

Design dynamic applications

Write dynamic components

Understand component coupling

Manage the impact of dynamism

Objective

Write robust centralized dynamic applications

Outline

Context & Challenges

State of the Art

Proposition

Implementation & Validation

Conclusion & Perspectives

A few

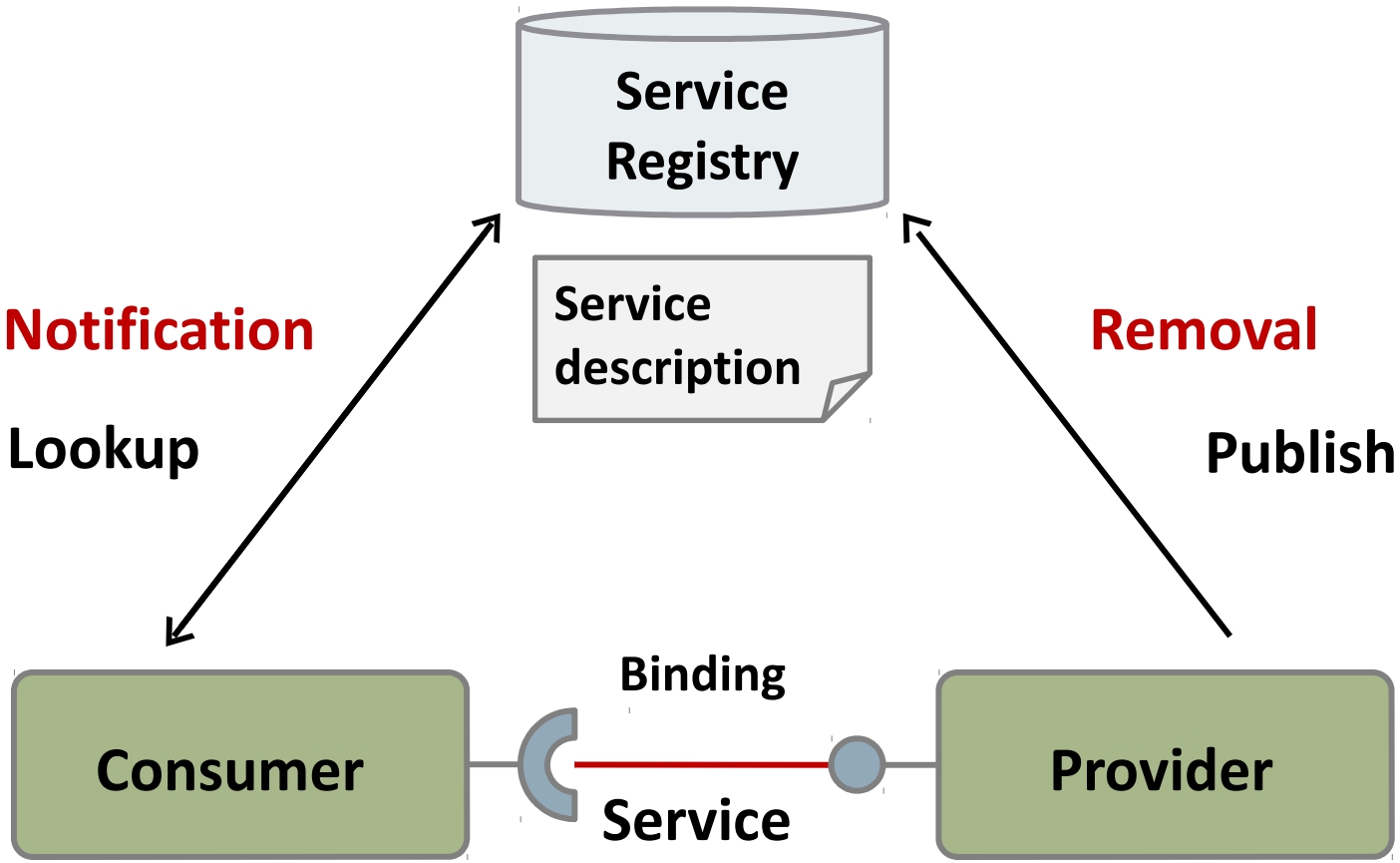
definitions

Service Oriented Architecture

Services are **self-describing,** platform-agnostic computational elements that support rapid, **low-cost composition** of distributed applications.

M. Papazoglou [2003]

SOA Component Interaction



Benefits of SOA

Reduced coupling

Dynamic resilience

Substitutability

Implementation transparency

Software architecture

> Abstractly, software architecture involves the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns.

M. Shaw and D. Garlan [1996]

Dynamic software architecture

> represent **systems** that do not simply consist of a fixed, static structure, but can **react** to certain requirements or events by **run-time reconfiguration** of its components and connections.

Baresi et al. [2004]

Advantages of Architecture

Programming-in-the-large

[DeRemer and Kron 1975]

High-level design & integration concerns

[Favre 1997]

Issues with

dynamism

1. Safe-stopping components

2. Handling stateful artifacts

3. Dynamism in software architectures

1

How to **safely stop** components and remove them from a running system?

Requirements to safe-stopping

Passivate components

Find safe-state

Remove old components

Instantiate new components

Avoid or recover from corruption

Ensure consistency.

Minimize disruption.

Transactional approaches

Quiescence

J. Kramer and J. Magee [1990]

Tranquility

Yves Vandewoude et al. [2007]

Version Consistency

Ma et al. [2011]

Component models	Fractal iPOJO OpenCom

Limits to current approaches

- Mostly proactive solutions
  - Lack of reactive approaches
- Too optimistic for many uses
  - Lack of recovery
- Unclear programming restrictions
  - What leads to coupling?
- Lack of centralized solutions
  - Distributed solutions impose higher decoupling

How to handle state?

2

No State Transfer.

Delegated State Transfer.

Automated State Transfer.

State transfer limitations

Hard to automate

Hybrid approaches show promise

Still no generic solution

Use simple ad-hoc state transfer.

3

Dynamism in software architectures?

Current approaches

Explicit vs. Constrained

Proactive vs. Reactive

Support unexpected change

devices, remote services, failure...

Reactive & Constrained

What's missing in
existing solutions?

What's missing?

Selectively enable dynamism

Manage the impact of dynamism

Development guidelines

Ensure consistency at runtime despite unexpected change

What does the **runtime** need to support **unexpected** dynamism?

Runtime requirements
for unexpected dynamism

Correctness & Consistency

Proactive & Reactive

Recovery

Additional runtime requirements

Change impact

Minimal disruption

Timeliness

Outline

Context & Challenges

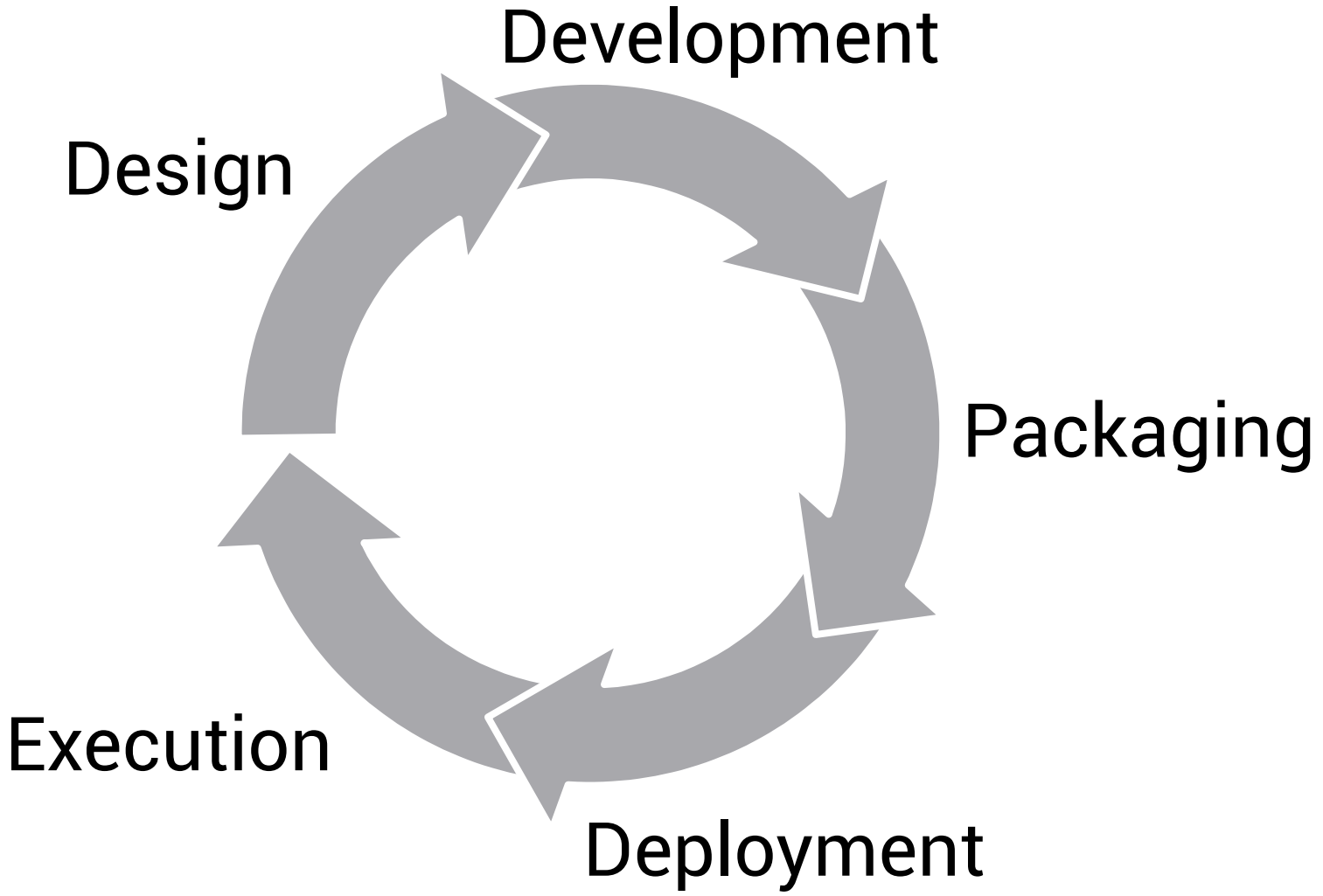
State of the Art

Robusta

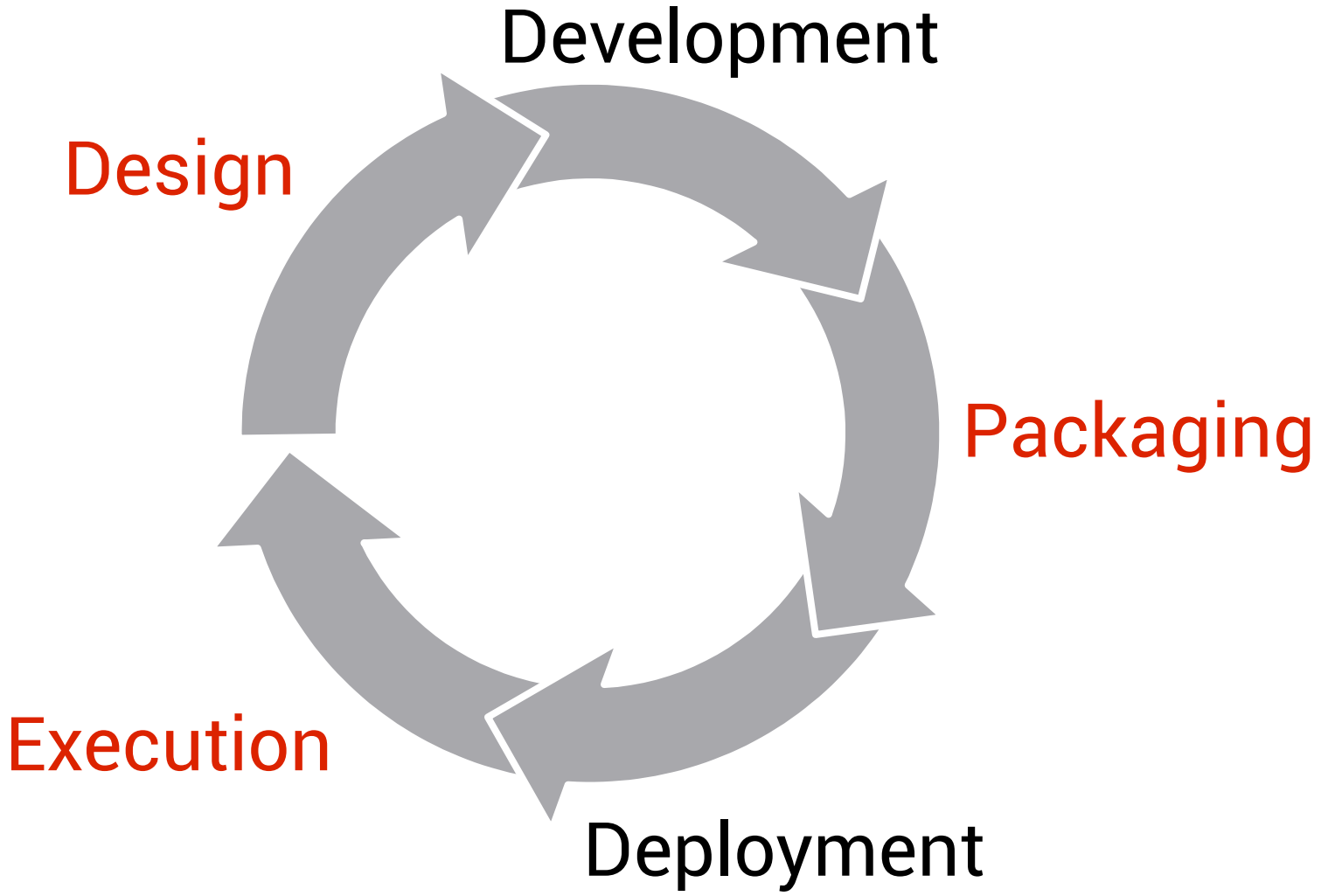
Implementation & Validation

Conclusion & Perspectives

The software cycle



The software cycle



Dynamism is …

Cross-cutting

- Design, deploy, develop, package, execute

Invasive

- Cannot be fully transparent

Difficult to understand

- Lack of tools and guidelines

Easy to get wrong

- Subtle mistakes cause disasters

Robusta is

An architectural approach to managing dynamism

Manages dynamism at multiple levels

Service, component, module and class, object

Particularly focused on design, packaging and execution

Robusta principles

Dynamism is not needed everywhere

Manage dynamism early

Architecture centric

- Services → component architecture
- Components → module architecture
- Modules → packaging architecture

Component behavior

Robusta component behavior

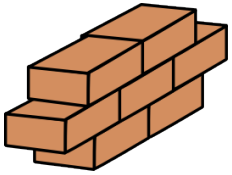
@Stable

@Detachable

@Volatile

Robusta component behavior

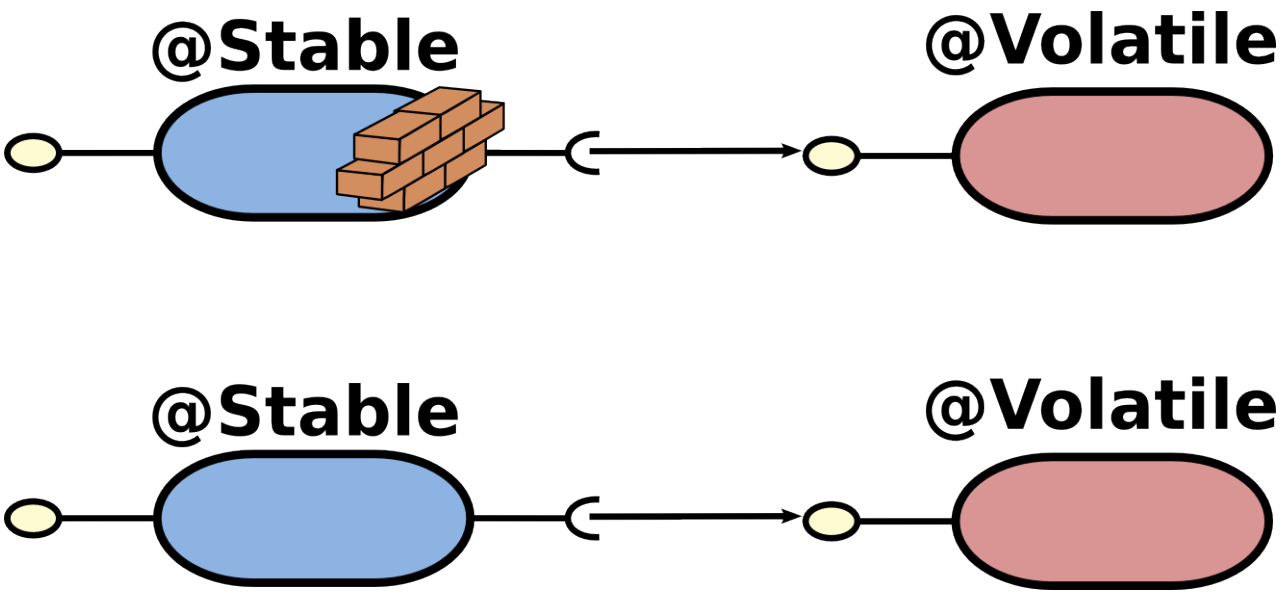
Determine decoupling and resilience required by dependencies



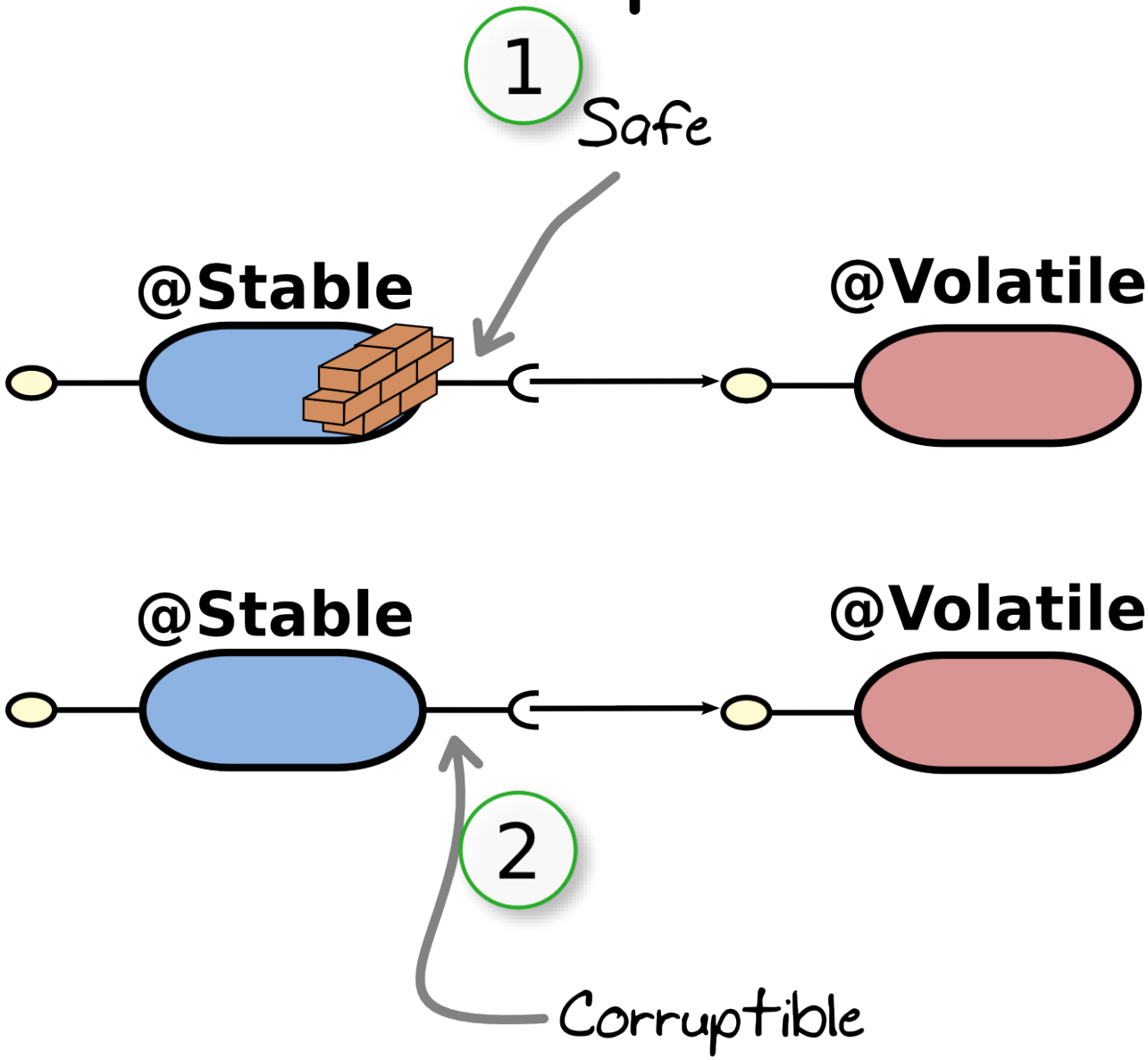
Resilience to volatility

Coupled dependency

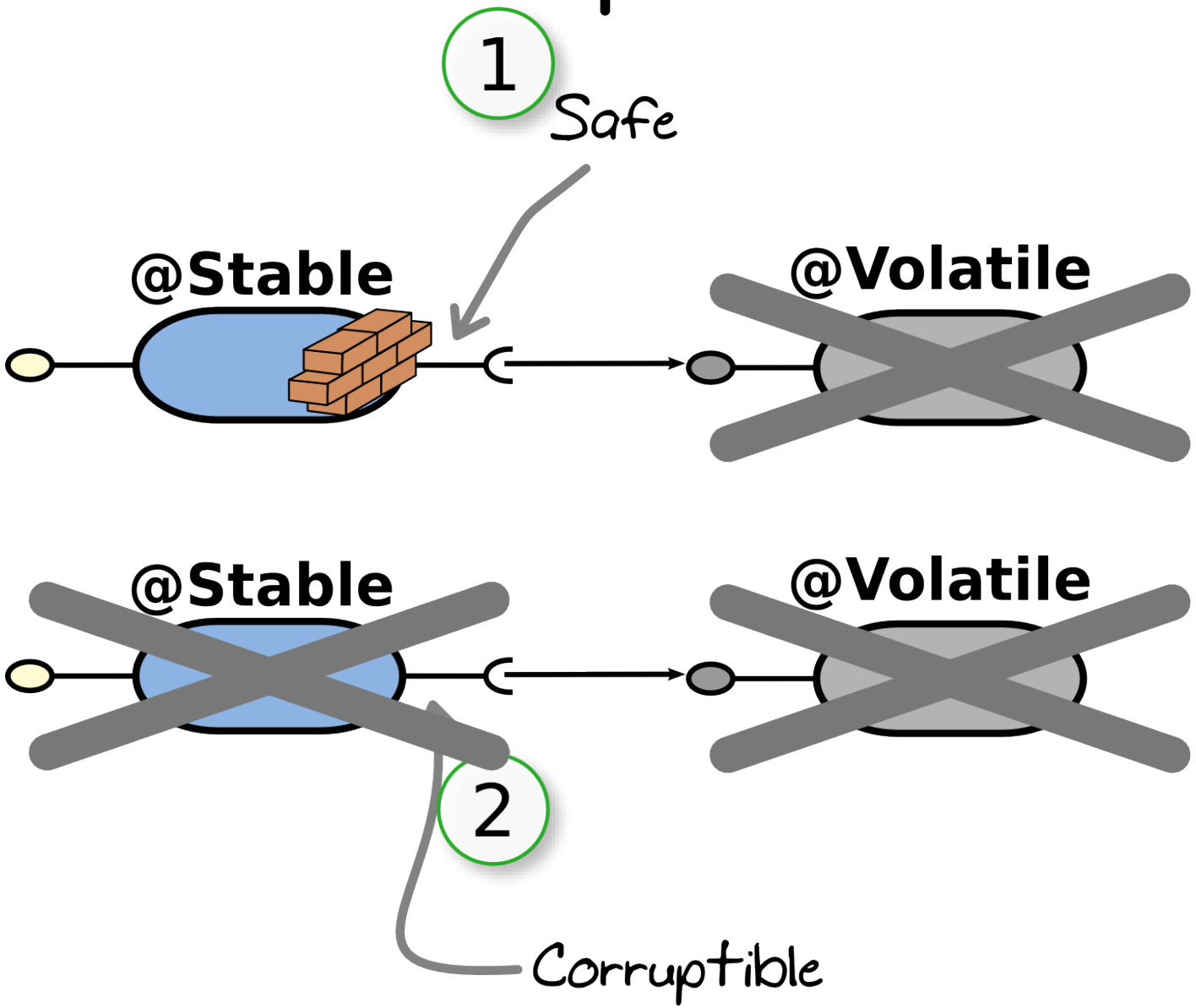
Example



Example



Example



Robusta component behavior

Protect `@stable` components

Decouple `@detachable` components

Isolate `@volatile` components

Component
decoupling

Decoupling requirements

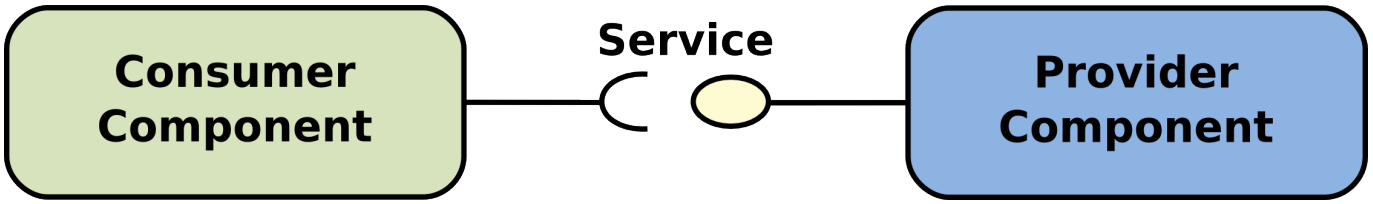
Multiple consumers & providers

Multiple versions of the same class

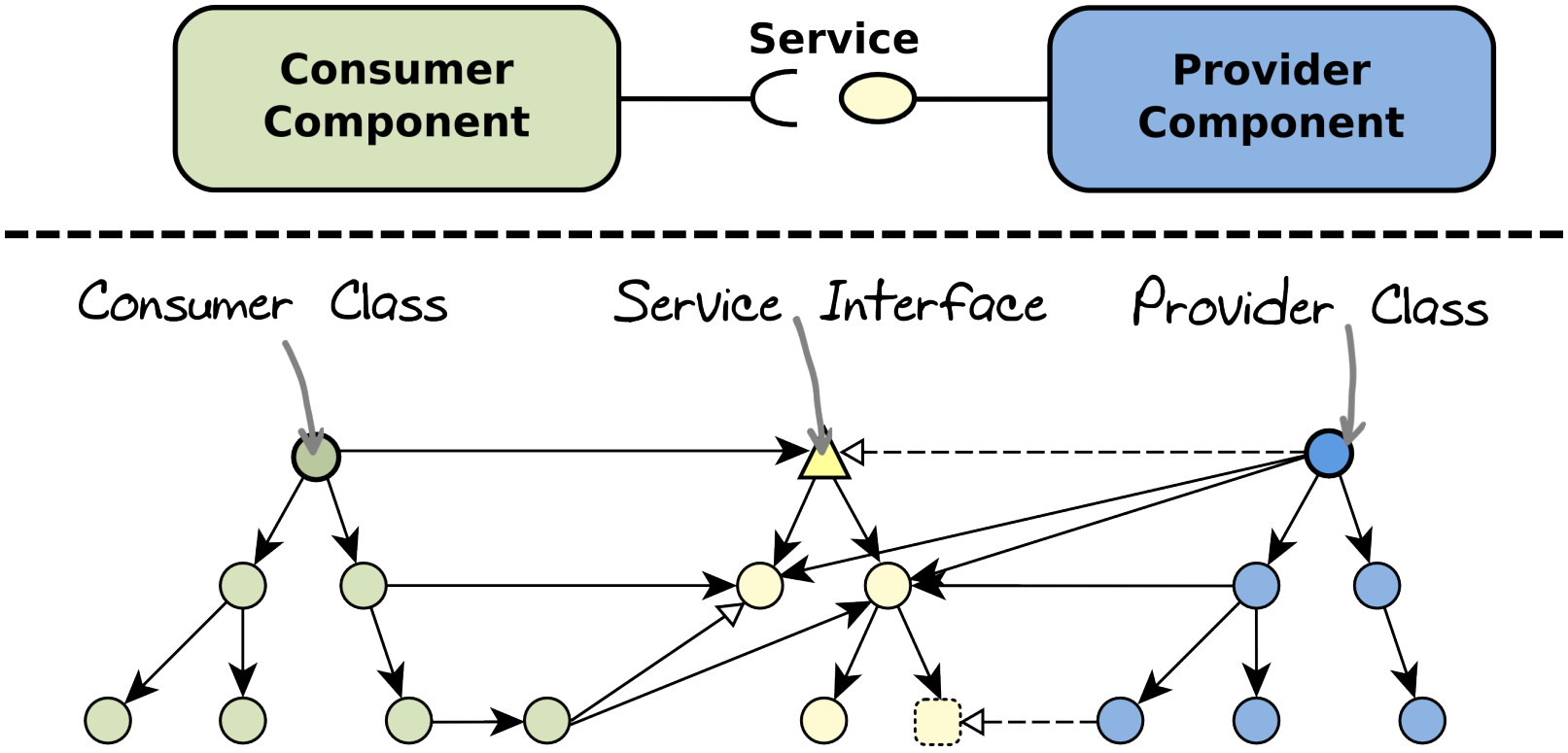
Complex objects

Service specialization

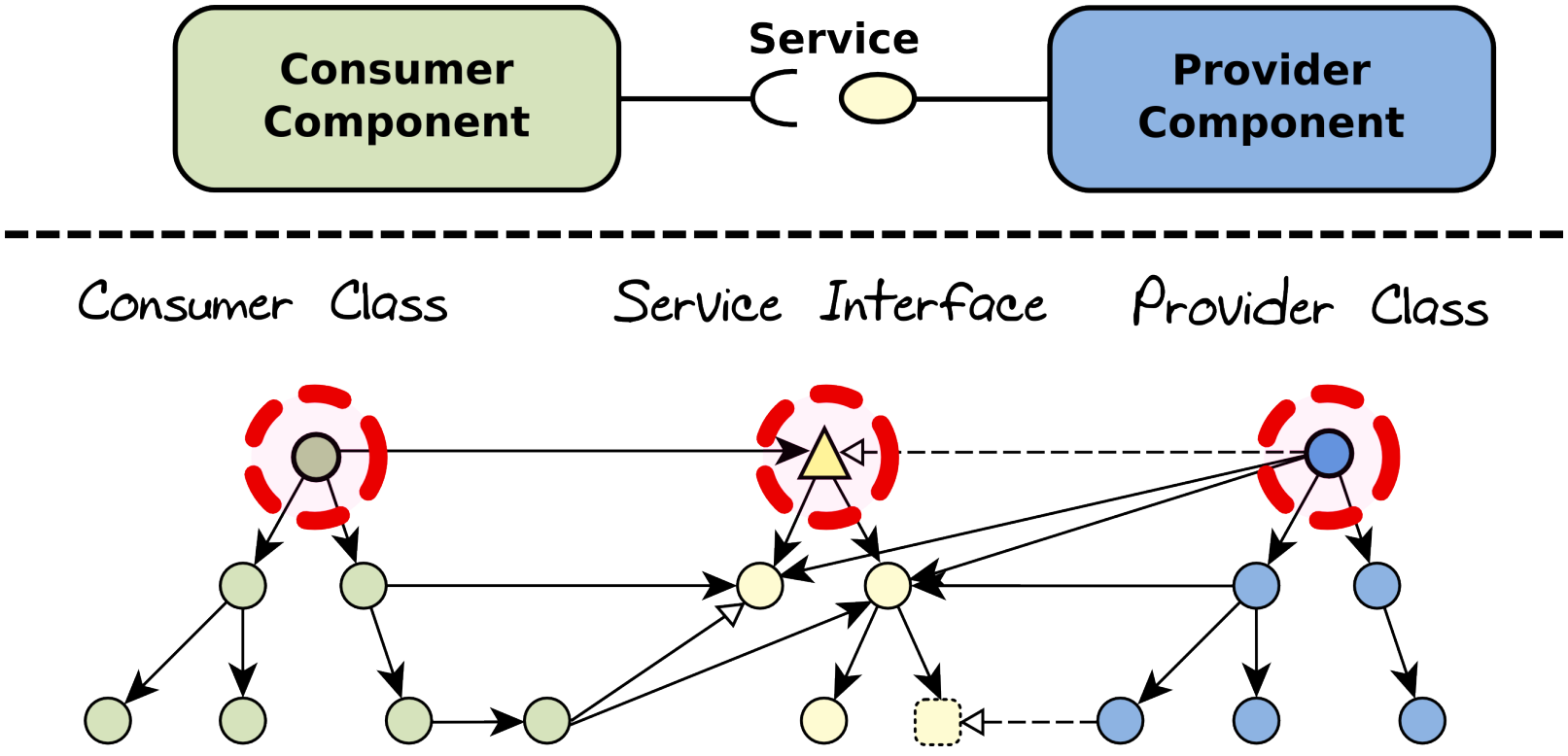
Decoupling example



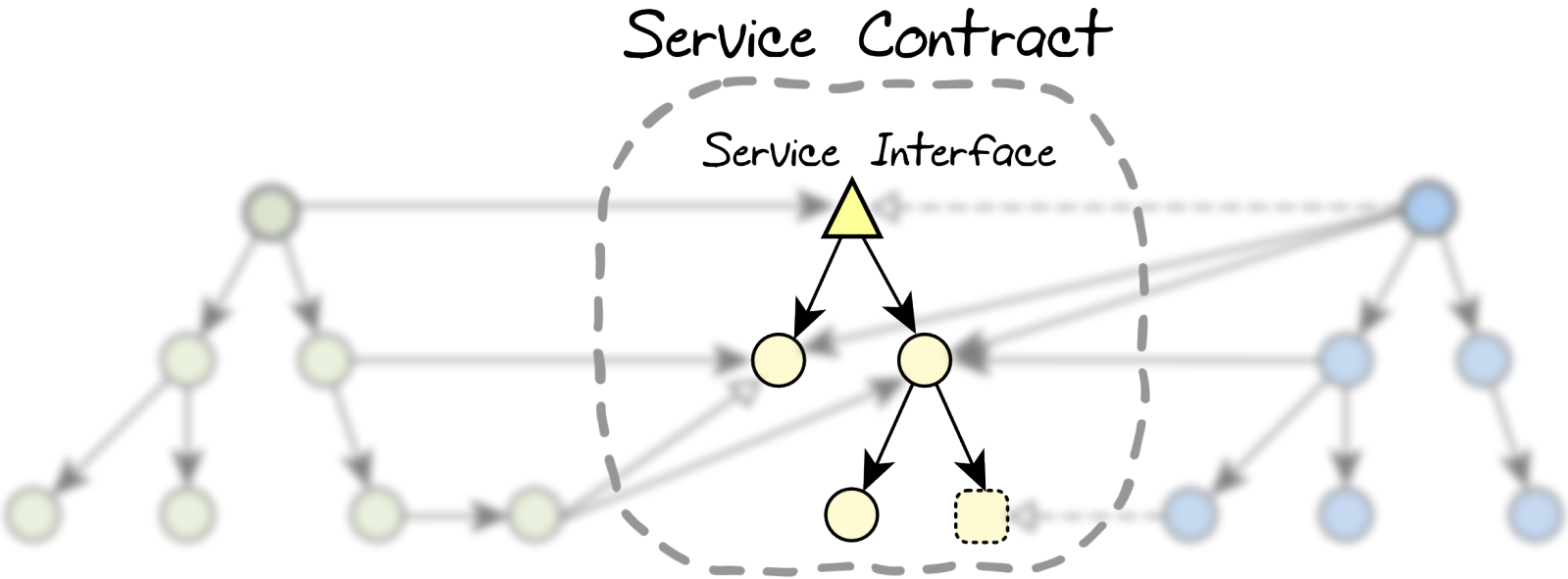
Decoupling example



Decoupling example



Decoupling example

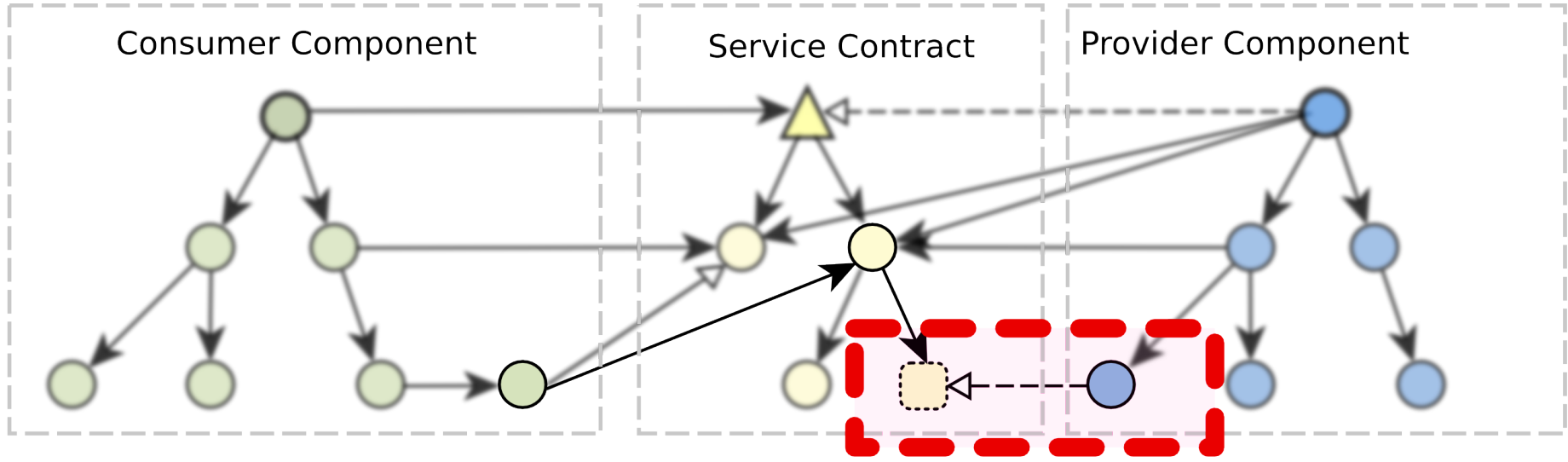


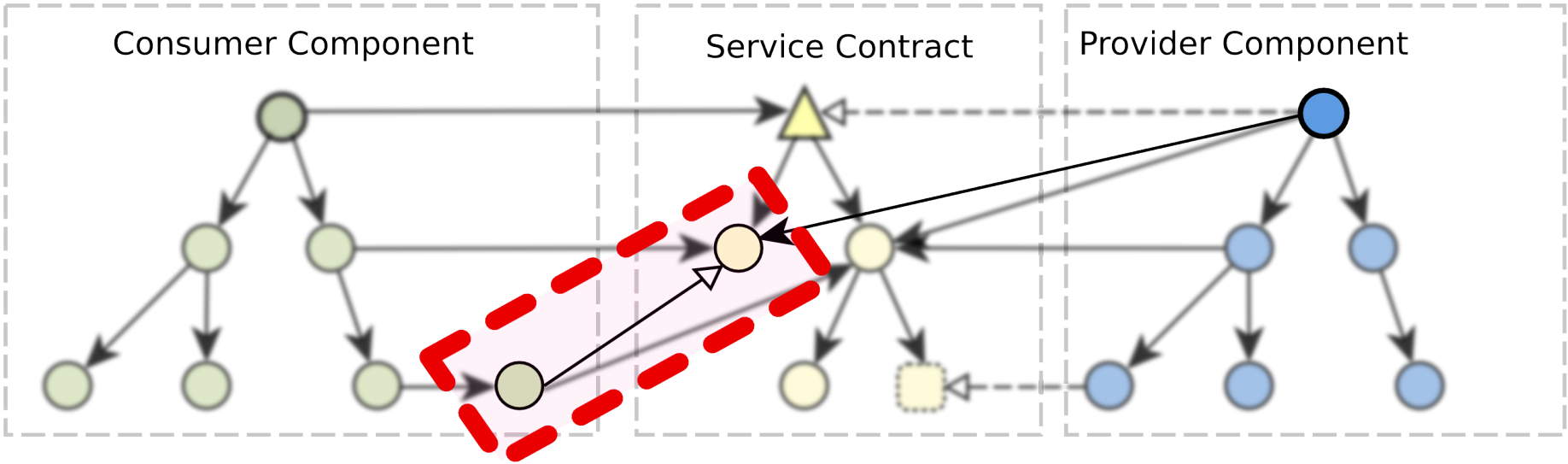
The service contract

The Service Interface and the types that it directly depends on.

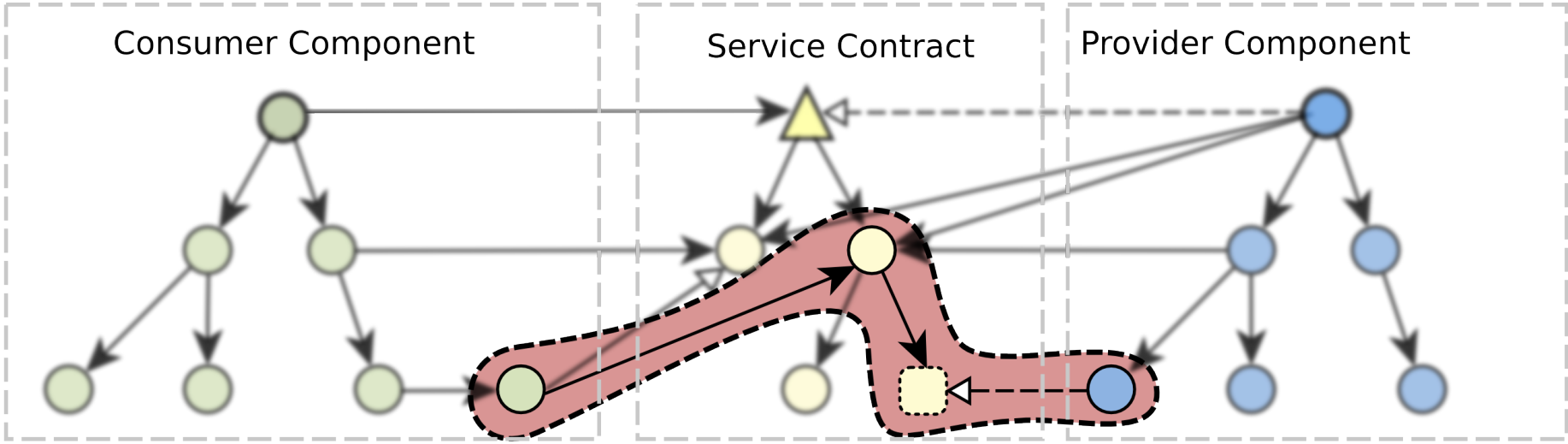
However, there's still
indirect (hidden) coupling!

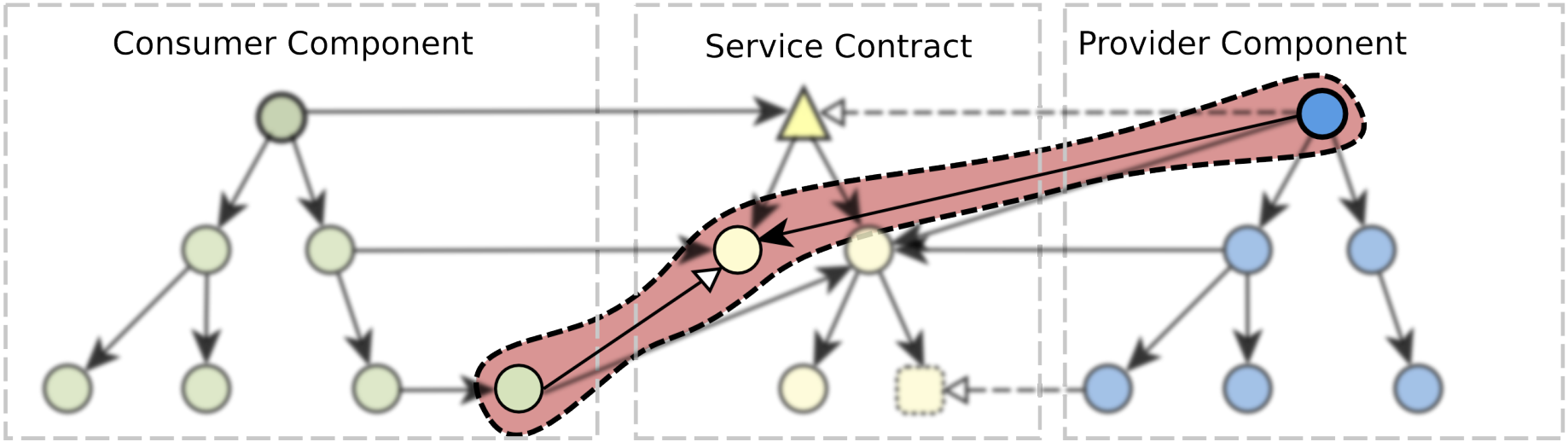
Decoupling example



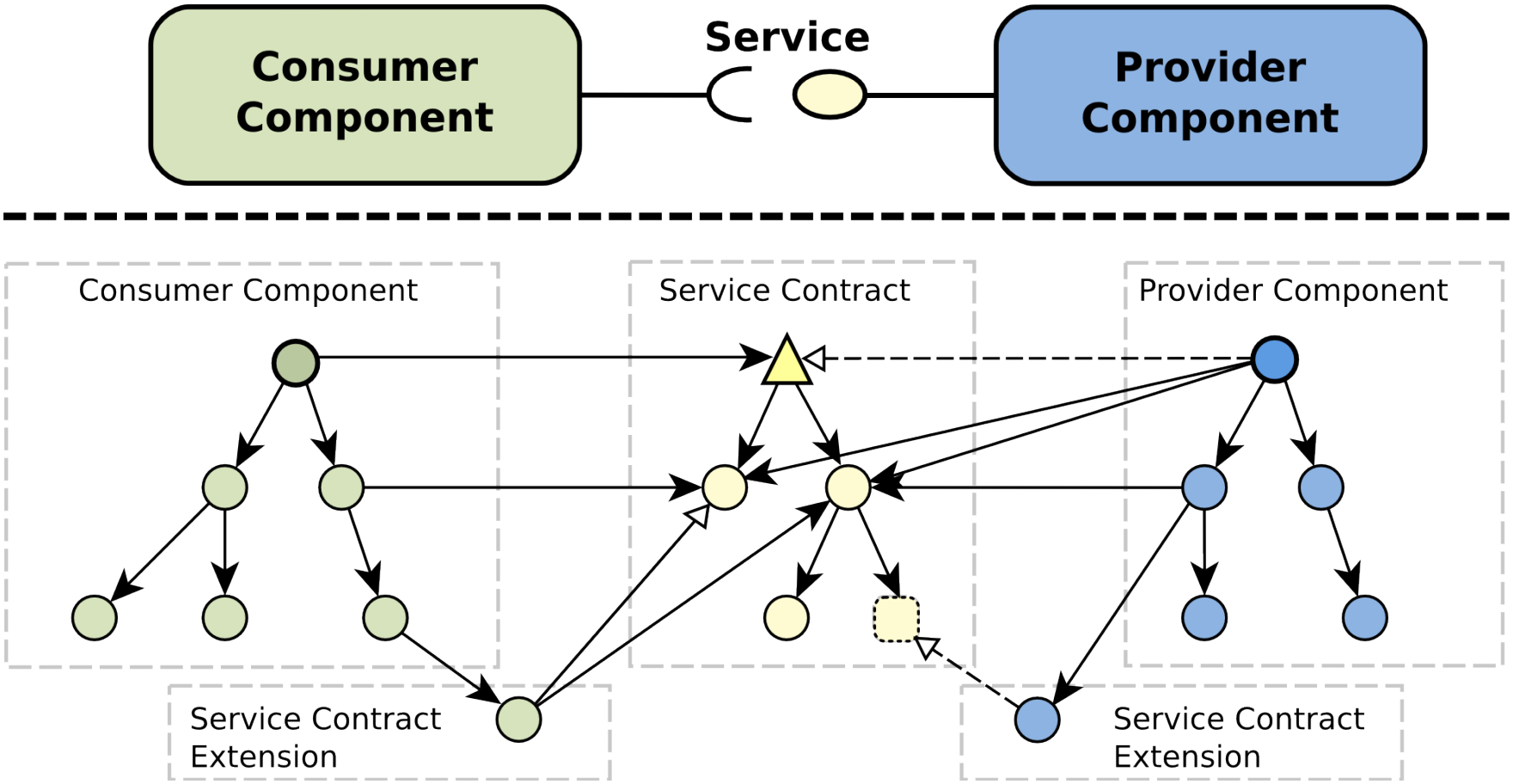


Decoupling example





Decoupling example



The extended service contract

The Service Interface and the types that it directly and indirectly depends on.

Component

packaging

Packaging

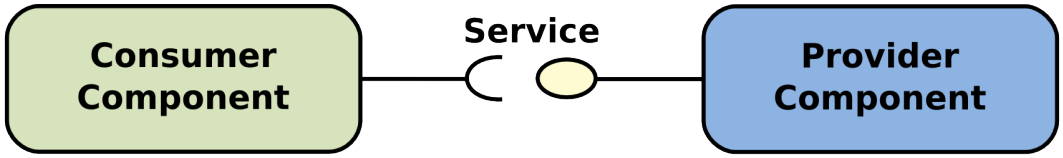
Defines class→modules

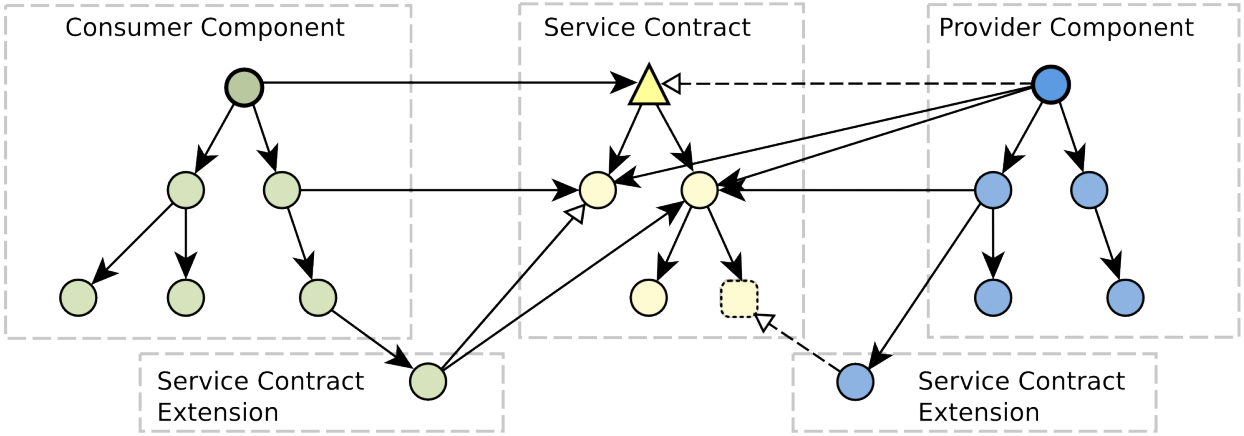
Modules are units of deployment

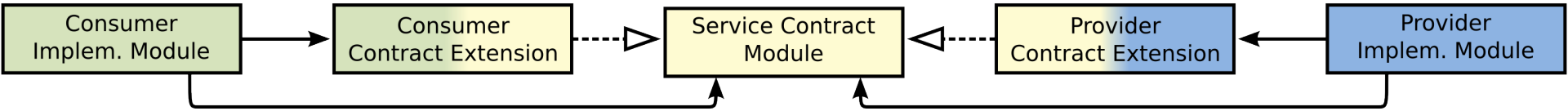
It's guided by

- Service Contract
- Contract Extensions
- Component Implementations

Packaging example







Packaging results

Components evolve independently

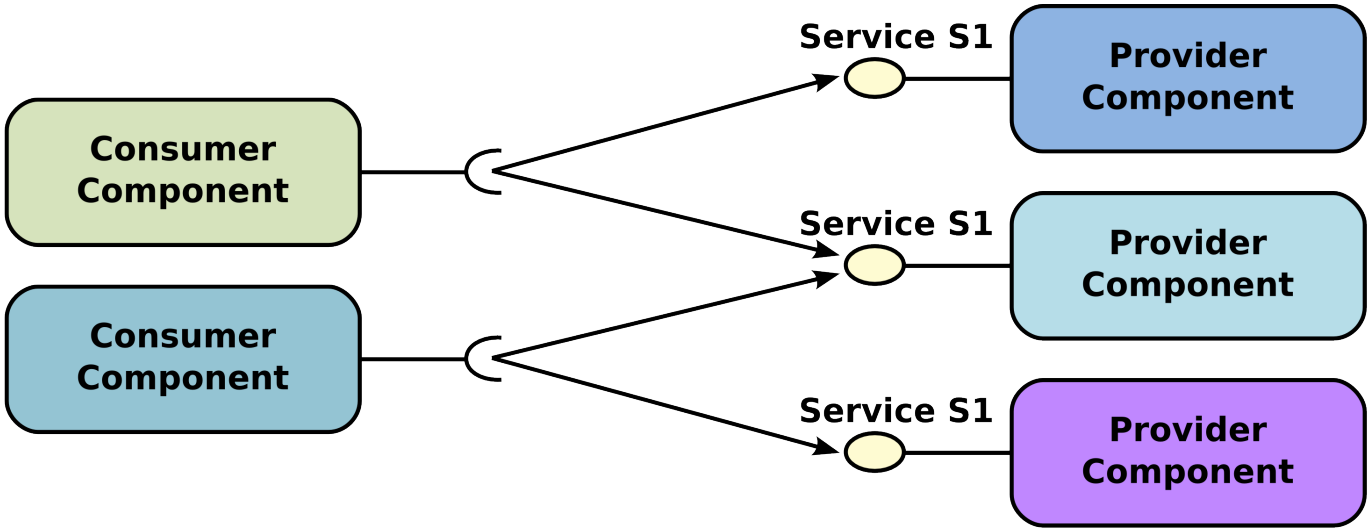
Isolate Service Contract

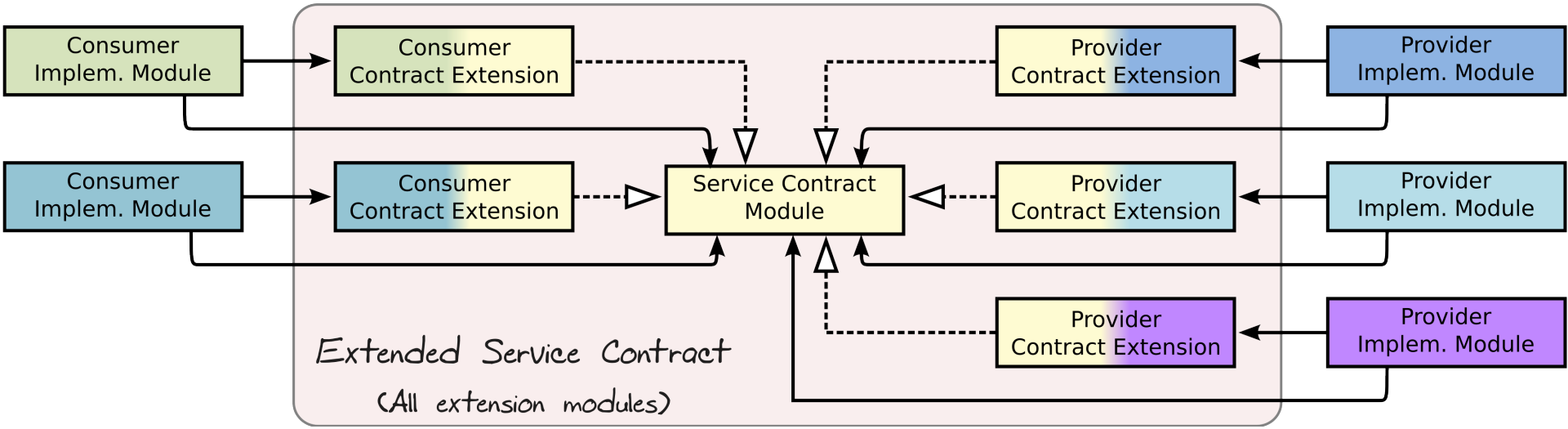
Specialize Service Contract

Avoid service incompatibilities

Multiple consumer→provider

Packaging example





Architecture

analysis

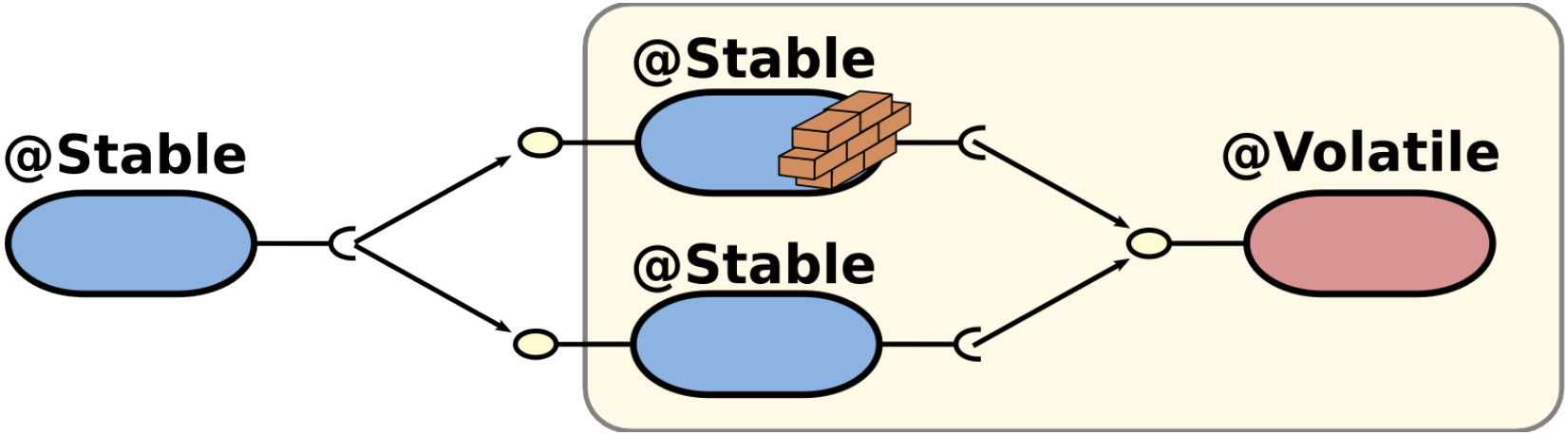
Architecture analysis requirements

Selective dynamism

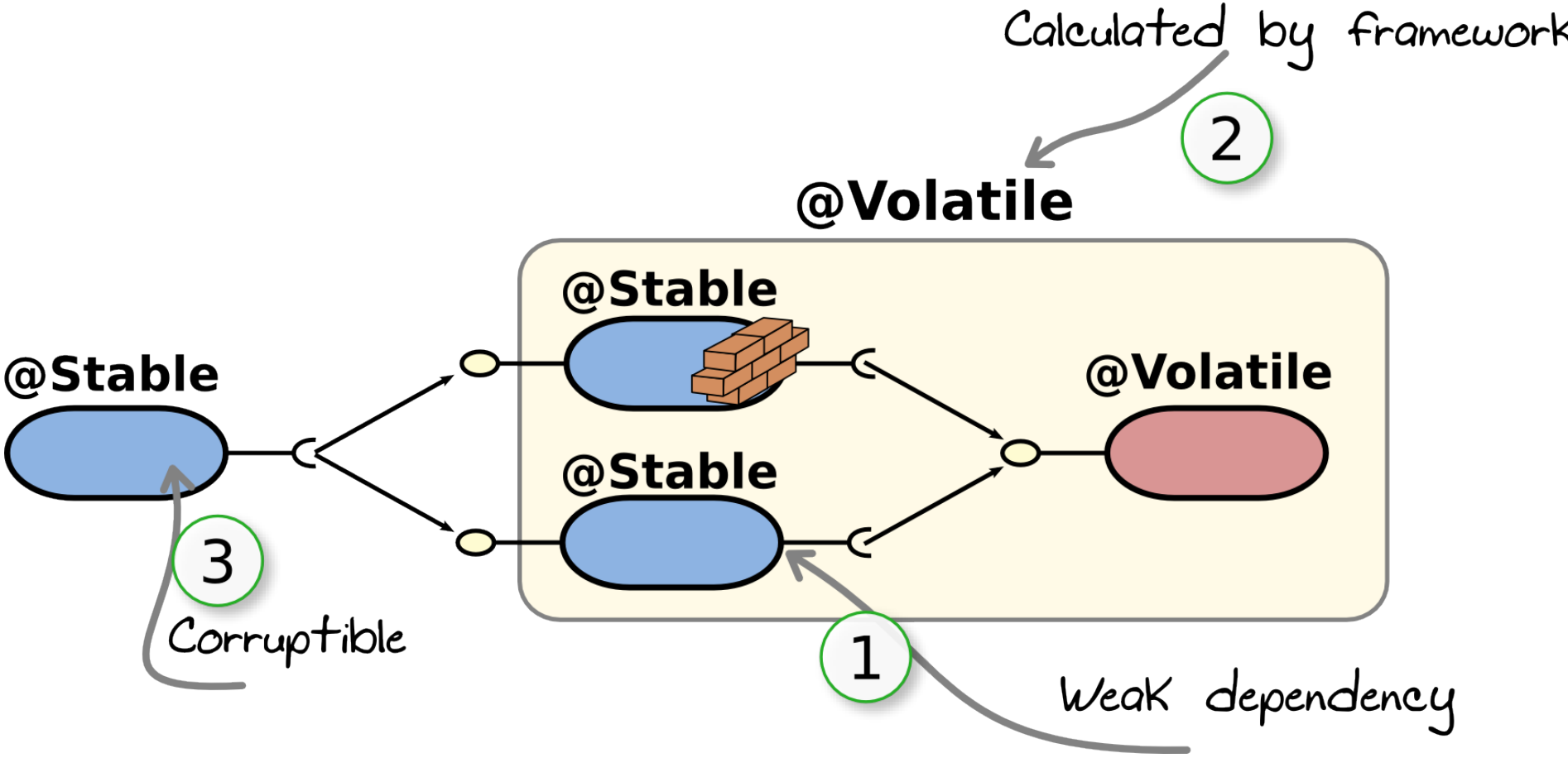
Zone-ification

Property composition

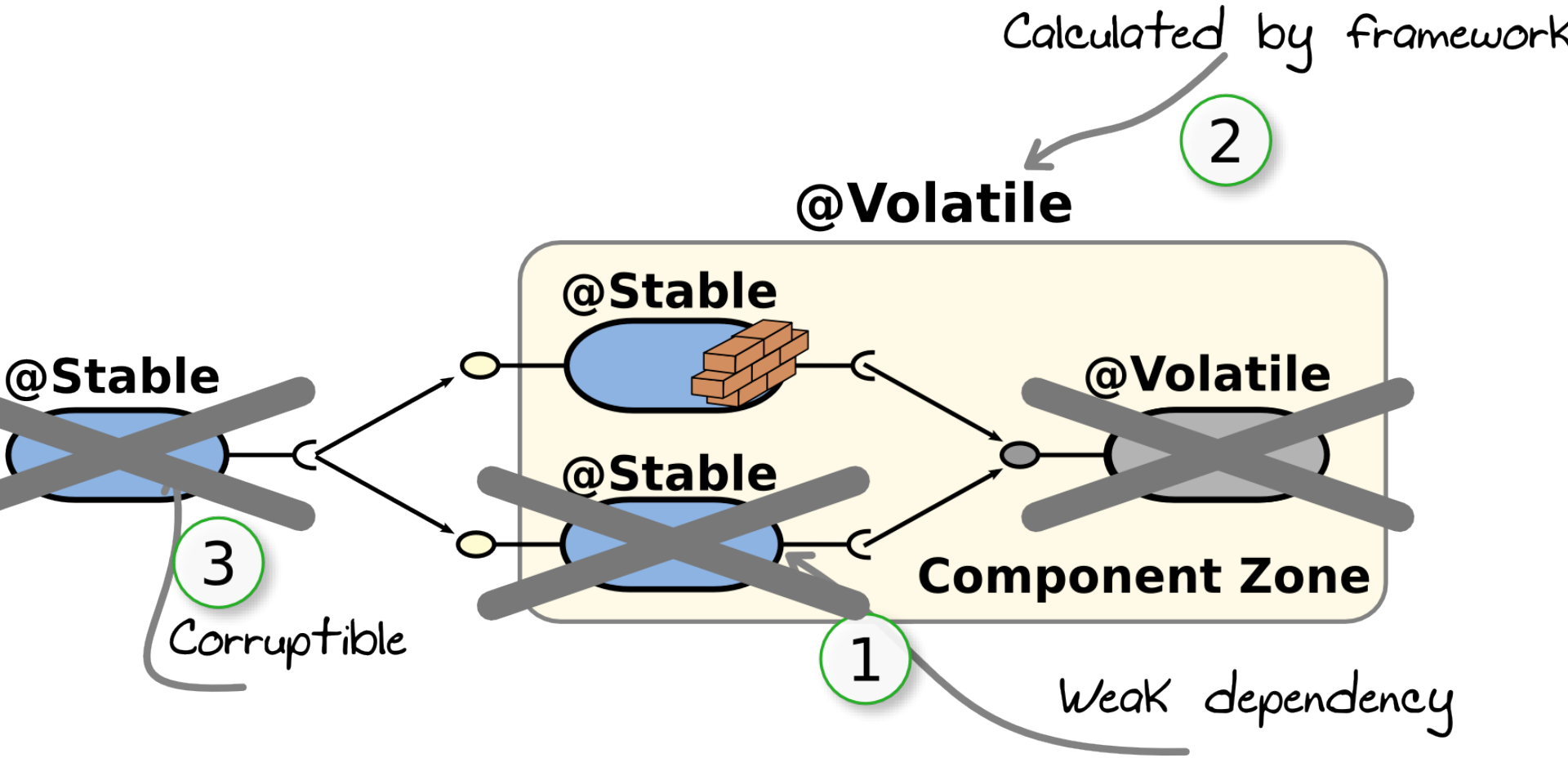
Volatile example



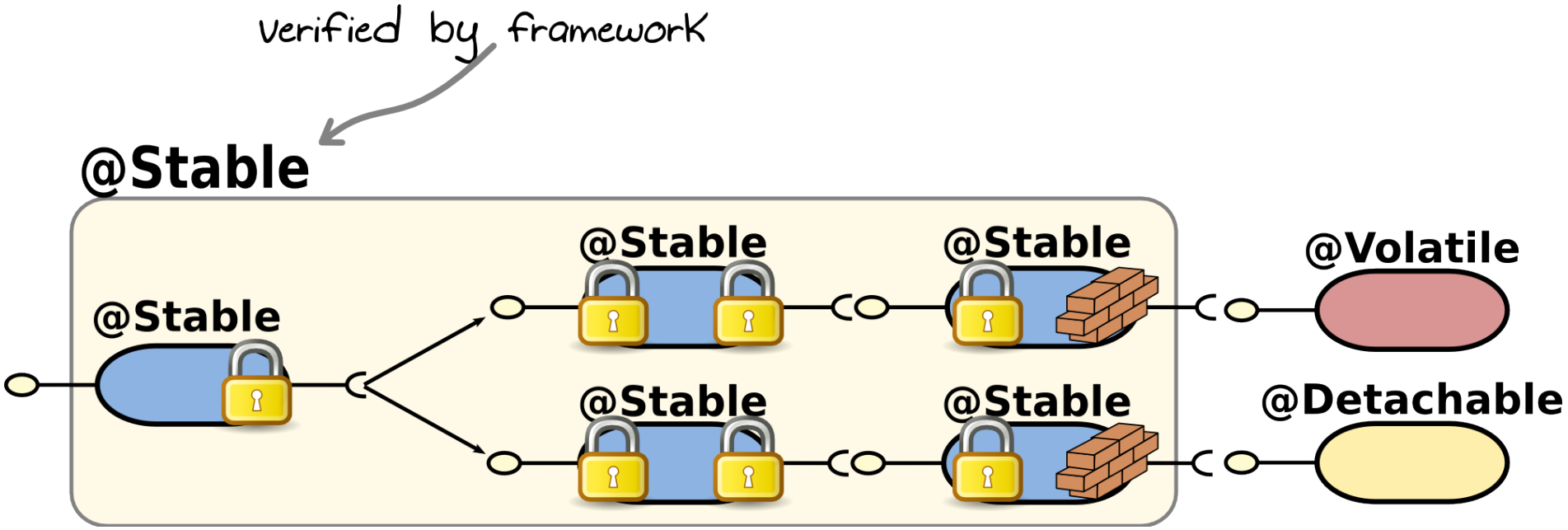
Volatile example



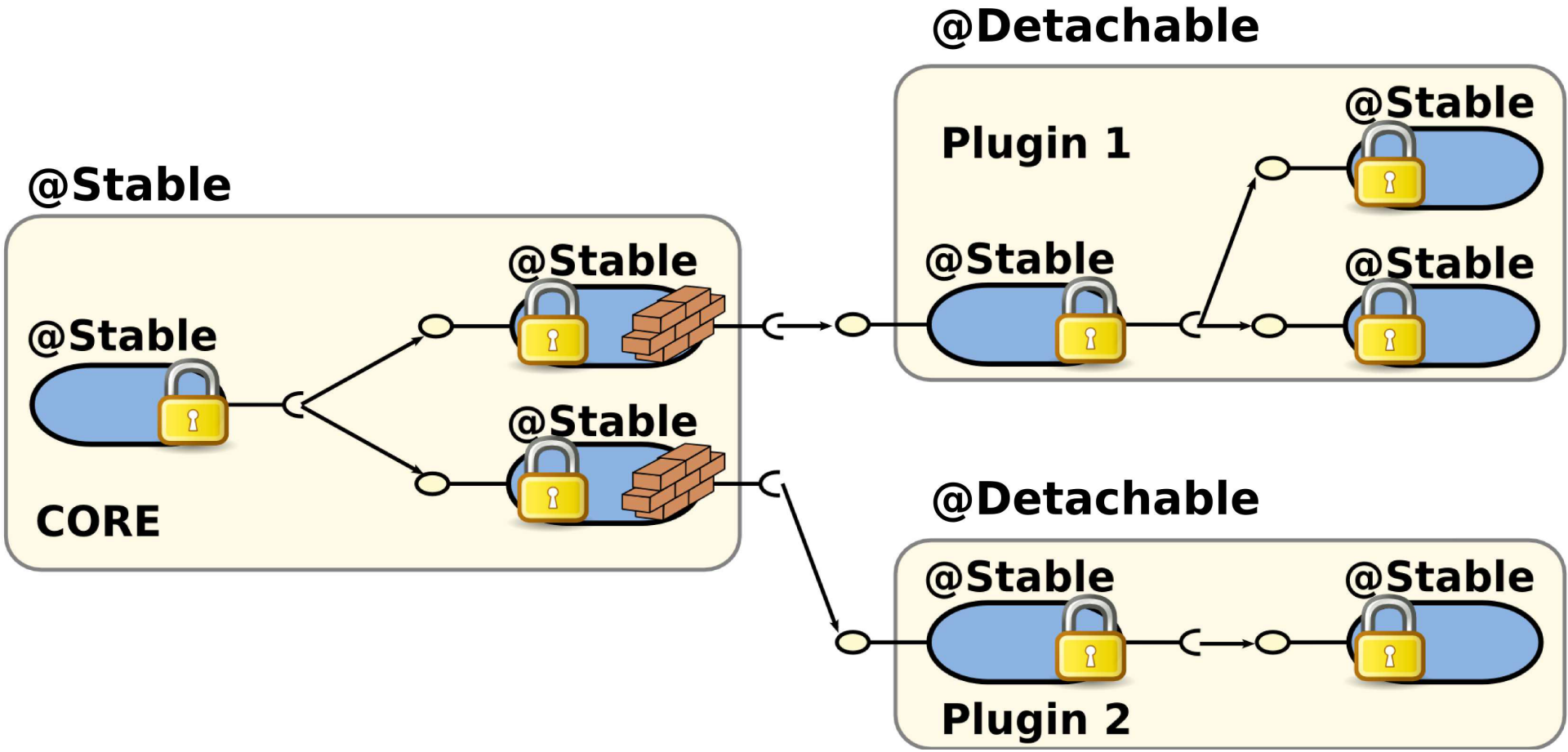
Volatile example



Stable example



Plugin example



Outline

Context & Challenges

State of the Art

Proposition

Implementation & Validation

Conclusion & Perspectives

Requirements

Build class dependency graph

- Analyze all Classes

Identify Class ↔ Module relations

Calculate Service Contract

- Simple graph algorithms

Open-world solution

Solution comparison

Design-time	***versus***	Runtime
Source code	***versus***	Bytecode
Automated Analysis	***versus***	Interactive diagnostics

Hardest case possible

Technical solution

Java agent for Instrumentation

Get and Instrument ALL Classes

Code injection (ASM)

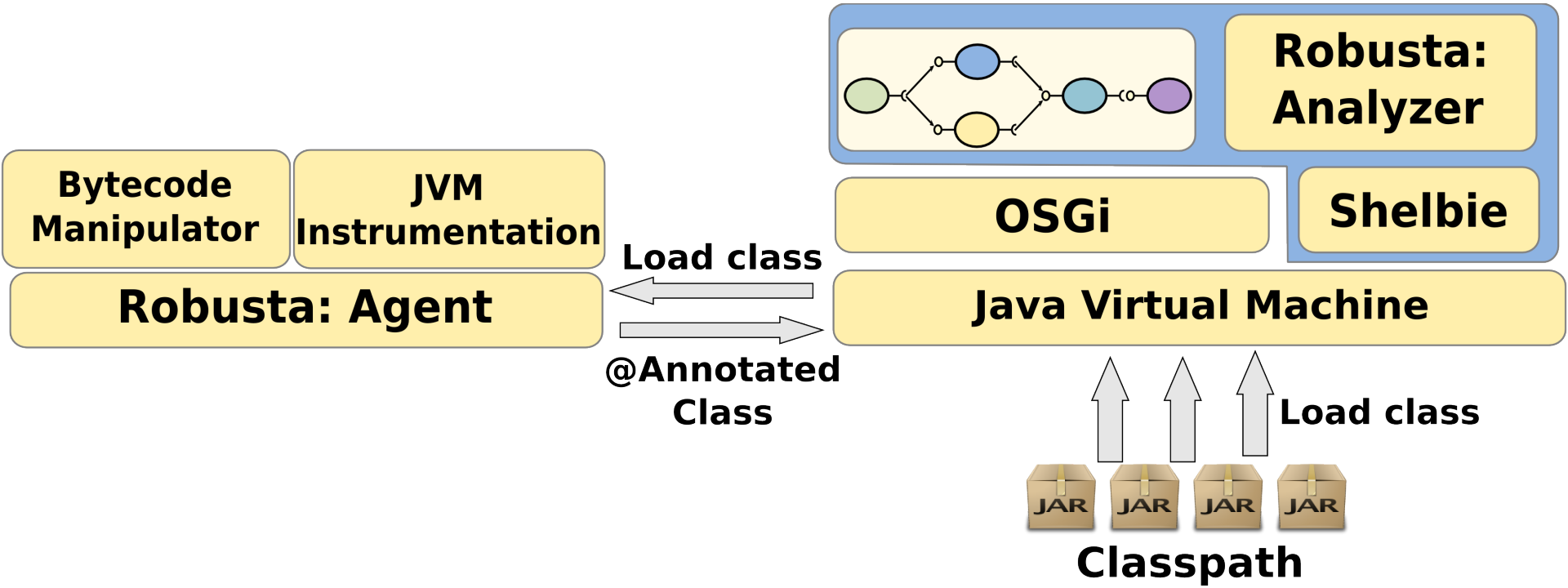
`@Robusta` & `@ClassDependency`

Includes filtering options

Interactive Commands with Shelbie

An OSGi Shell

Technical solution



Interactive command-line

Classloaders (i.e., Modules)

Classes

Duplicates

Service Contract calculation

- Transitive dependency graph
- Extensions too

Graphs represent current state unambiguously.

Experimentation

OW2 JonAS Java EE Application Server

+ 300 modules

+ 120 composants

+ 400k Lines of Code

Console based output

```
**********************************************************************************************
******        ClassLoader List showing Parent and Loading classloaders        ******
**********************************************************************************************
  1: bundle:     loader: bootstrap (NULL)                                                    parent: null-parent
  2: bundle:     loader: sun.reflect.DelegatingClassLoader@766d65fd                          parent: org.apache.felix.
  3: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@3b1aed57  parent: null-parent
  4: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@3ec7d45e  parent: null-parent
  5: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@80f252    parent: null-parent
  6: bundle: 37  loader: org.apache.felix.framework.BundleWiringImpl@2d14a694                parent: null-parent
  7: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@7dd9578a  parent: null-parent
  8: bundle:     loader: sun.reflect.DelegatingClassLoader@a563d79                           parent: org.apache.felix.
  9: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@5373b318  parent: null-parent
 10: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@e3f6d     parent: null-parent
 11: bundle: 291 loader: org.apache.felix.framework.BundleWiringImpl@5e536b73                parent: null-parent
 12: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@7f8e1a98  parent: null-parent
 13: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@5eb2c603  parent: null-parent
 14: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$NullableClassLoader@2829306c parent: sun.misc.Launche
 15: bundle:     loader: sun.reflect.DelegatingClassLoader@db5eaed                           parent: org.apache.felix.
 16: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@1a06c11d  parent: null-parent
 17: bundle: 336 loader: org.apache.felix.framework.BundleWiringImpl@14ea0724                parent: null-parent
 18: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$NullableClassLoader@57b5b346 parent: sun.misc.Launche
 19: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$NullableClassLoader@4f9380c1 parent: sun.misc.Launche
 20: bundle: 9   loader: org.apache.felix.framework.BundleWiringImpl@7157c76a                parent: null-parent
 21: bundle: 139 loader: org.apache.felix.framework.BundleWiringImpl@2b3cfcf1                parent: null-parent
 22: bundle: 336 loader: org.apache.felix.framework.BundleWiringImpl@107ad736                parent: null-parent
 23: bundle: 219 loader: org.apache.felix.framework.BundleWiringImpl@f2c03ac                 parent: null-parent
 24: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$NullableClassLoader@10e07658 parent: sun.misc.Launche
 25: bundle:     loader: sun.reflect.DelegatingClassLoader@3bc79148                          parent: org.apache.felix.
 26: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$NullableClassLoader@1ece988a parent: sun.misc.Launche
 27: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@202d0e1c  parent: null-parent
 28: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$NullableClassLoader@5cdc6180 parent: sun.misc.Launche
 29: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@34bf7baa  parent: null-parent
 30: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@2919d975  parent: null-parent
 31: bundle:     loader: sun.reflect.DelegatingClassLoader@60d861b7                          parent: org.apache.felix.
 32: bundle:     loader: sun.reflect.DelegatingClassLoader@187b2d93                          parent: org.apache.felix.
 33: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@3e1dfb2   parent: null-parent
 34: bundle: 249 loader: org.apache.felix.framework.BundleWiringImpl@4083633f                parent: null-parent
 35: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@642ff0de  parent: null-parent
 36: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@34780af5  parent: null-parent
 37: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@217c26cc  parent: null-parent
 38: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@147cd80e  parent: null-parent
 39: bundle:     loader: org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@5e279c88  parent: null-parent
 40: bundle: 336 loader: org.apache.felix.framework.BundleWiringImpl@443104bc6               parent: null-parent
```

Dependency trees

```
**********************************************************************************************
*****        Printing classloader loader tree (how the classloaders were loaded)        *****
**********************************************************************************************
[  1] Bootstrap (NULL): System Classloader
[  2]         |---sun.reflect.DelegatingClassLoader@766d65fd
[  3]         |---sun.reflect.DelegatingClassLoader@a563d79
[  4]         |---sun.reflect.DelegatingClassLoader@db5eaed
[  5]         |---sun.reflect.DelegatingClassLoader@3bc79148
[  6]         |---sun.reflect.DelegatingClassLoader@60d861b7
[  7]         |---sun.reflect.DelegatingClassLoader@187b2d93
[  8]         |---sun.reflect.DelegatingClassLoader@a2c6f70
[  9]         |---sun.reflect.DelegatingClassLoader@2b988802
[ 10]         |---sun.reflect.DelegatingClassLoader@1d618248
[ 11]         |---java.net.URLClassLoader@61a116c9
[ 12]                 |---org.apache.felix.framework.BundleWiringImpl@2d14a694
[ 13]                 |---org.apache.felix.framework.BundleWiringImpl@5e536b73
[ 14]                 |---org.apache.felix.framework.BundleWiringImpl@14ea0724
[ 15]                 |---org.apache.felix.framework.BundleWiringImpl@7157c76a
[ 16]                 |---org.apache.felix.framework.BundleWiringImpl@2b3cfcf1
[ 17]                 |---org.apache.felix.framework.BundleWiringImpl@107ad736
[ 18]                 |---org.apache.felix.framework.BundleWiringImpl@f2c03ac
[ 19]                 |---org.apache.felix.framework.BundleWiringImpl@4083633f
[ 20]                 |---org.apache.felix.framework.BundleWiringImpl@44319bc6
[ 21]                 |---org.apache.felix.framework.BundleWiringImpl@36b37b66
[ 22]                 |---org.apache.felix.framework.BundleWiringImpl@72d876d9
[ 23]                 |---org.apache.felix.framework.BundleWiringImpl@4915a928
[ 24]                 |---org.apache.felix.framework.BundleWiringImpl@39e53a48
[ 25]                 |---org.apache.felix.framework.BundleWiringImpl@78556aa9
[ 26]                 |---org.apache.felix.framework.BundleWiringImpl@53642565
[ 27]                 |---org.apache.felix.framework.BundleWiringImpl@2f19f33d
[ 28]                 |---org.apache.felix.framework.BundleWiringImpl@2fcdbaf7
[ 29]                 |---org.apache.felix.framework.BundleWiringImpl@1858c80c
[ 30]                 |---org.apache.felix.framework.BundleWiringImpl@2f8891c4
[ 31]                 |---org.apache.felix.framework.BundleWiringImpl@53d334a8
[ 32]                 |---org.apache.felix.framework.BundleWiringImpl@cdeb65f
[ 33]                         |---org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@3b1aed57
[ 34]                         |---org.apache.felix.ipojo.handlers.dependency.Dependency$SmartProxyFactory@3ec7d45e
```

Duplicated classes

```
rudametw@jonas$ robusta:class -sort -c -cl -v -n | grep robust | grep -v Already | grep -v Added
264:org.apache.felix.framework.BundleWiringImpl@70c722ad:fr.adele.robusta.commands.TestAction
276:org.apache.felix.framework.BundleWiringImpl@70c722ad:fr.adele.robusta.dependencygraph.ClassUt
326:org.apache.felix.framework.BundleWiringImpl@44319bc6:fr.adele.robusta.dependencygraph.ClassLo
559:org.apache.felix.framework.BundleWiringImpl@571f0759:fr.adele.robusta.dependencygraph.ClassLo
571:org.apache.felix.framework.BundleWiringImpl@571f0759:fr.adele.robusta.internal.util.AnsiPrint
662:org.apache.felix.framework.BundleWiringImpl@21b5c5b3:fr.adele.robusta.dependencygraph.ClassLo
663:org.apache.felix.framework.BundleWiringImpl@107ad736:fr.adele.robusta.dependencygraph.ClassLo
884:org.apache.felix.framework.BundleWiringImpl@44319bc6:fr.adele.robusta.internal.util.GraphWrit
911:org.apache.felix.framework.BundleWiringImpl@44319bc6:fr.adele.robusta.commands.ClassAction
965:org.apache.felix.framework.BundleWiringImpl@44319bc6:fr.adele.robusta.internal.util.AnsiPrint
1003:sun.misc.Launcher$AppClassLoader@12360be0:fr.adele.robusta.agent.RobustaJavaAgent
1014:sun.misc.Launcher$AppClassLoader@12360be0:fr.adele.robusta.agent.manipulator.Dependency
1076:org.apache.felix.framework.BundleWiringImpl@21b5c5b3:fr.adele.robusta.dependencygraph.ClassL
1218:org.apache.felix.framework.BundleWiringImpl@44319bc6:fr.adele.robusta.commands.DumpAction
1268:org.apache.felix.framework.BundleWiringImpl@14ea0724:fr.adele.robusta.internal.util.AnsiPrin
1566:org.apache.felix.framework.BundleWiringImpl@571f0759:fr.adele.robusta.dependencygraph.ClassU
1669:org.apache.felix.framework.BundleWiringImpl@70c722ad:fr.adele.robusta.dependencygraph.ClassL
1716:org.apache.felix.framework.BundleWiringImpl@14ea0724:fr.adele.robusta.commands.ClassAction
1765:org.apache.felix.framework.BundleWiringImpl@14ea0724:fr.adele.robusta.dependencygraph.Classl
1776:org.apache.felix.framework.BundleWiringImpl@571f0759:fr.adele.robusta.dependencygraph.ClassL
1891:org.apache.felix.framework.BundleWiringImpl@70c722ad:fr.adele.robusta.dependencygraph.ClassT
2034:org.apache.felix.framework.BundleWiringImpl@571f0759:fr.adele.robusta.dependencygraph.ClassL
2083:org.apache.felix.framework.BundleWiringImpl@571f0759:fr.adele.robusta.dependencygraph.Classl
2088:org.apache.felix.framework.BundleWiringImpl@107ad736:fr.adele.robusta.commands.TestAction
2111:org.apache.felix.framework.BundleWiringImpl@14ea0724:fr.adele.robusta.internal.util.GraphWri
2149:sun.misc.Launcher$AppClassLoader@12360be0:fr.adele.robusta.annotations.Robusta
```

Statistics

```
*****************************************
*****          Statistics           *****
*****************************************

    Total number of classes: 5125
    Total number of duplicated classes: 223
    Total number of classloaders (including hidden): 416
    Total number of classes intercepted: 4295
    Total number of non_modified_classes: 0
    Total number of modified_classes: 4295
    Total number of redefined_classes: 0

***  Total execution time: 23 miliseconds  ***

rudametw@jonas$
```

Overhead

Memory

- 1 X annotation per class-dependency
- 4 X attributes per annotation

Execution

- Graph calculation times: 20ms – **160ms**
- Agent manipulation times: **~9ms**
  - Between 0ms – 220ms

No execution overhead when not used.

Lessons

Loading all classes

Java is lazy, Garbage Collection is lazy

Duplicate classes do happen

Root hierarchy

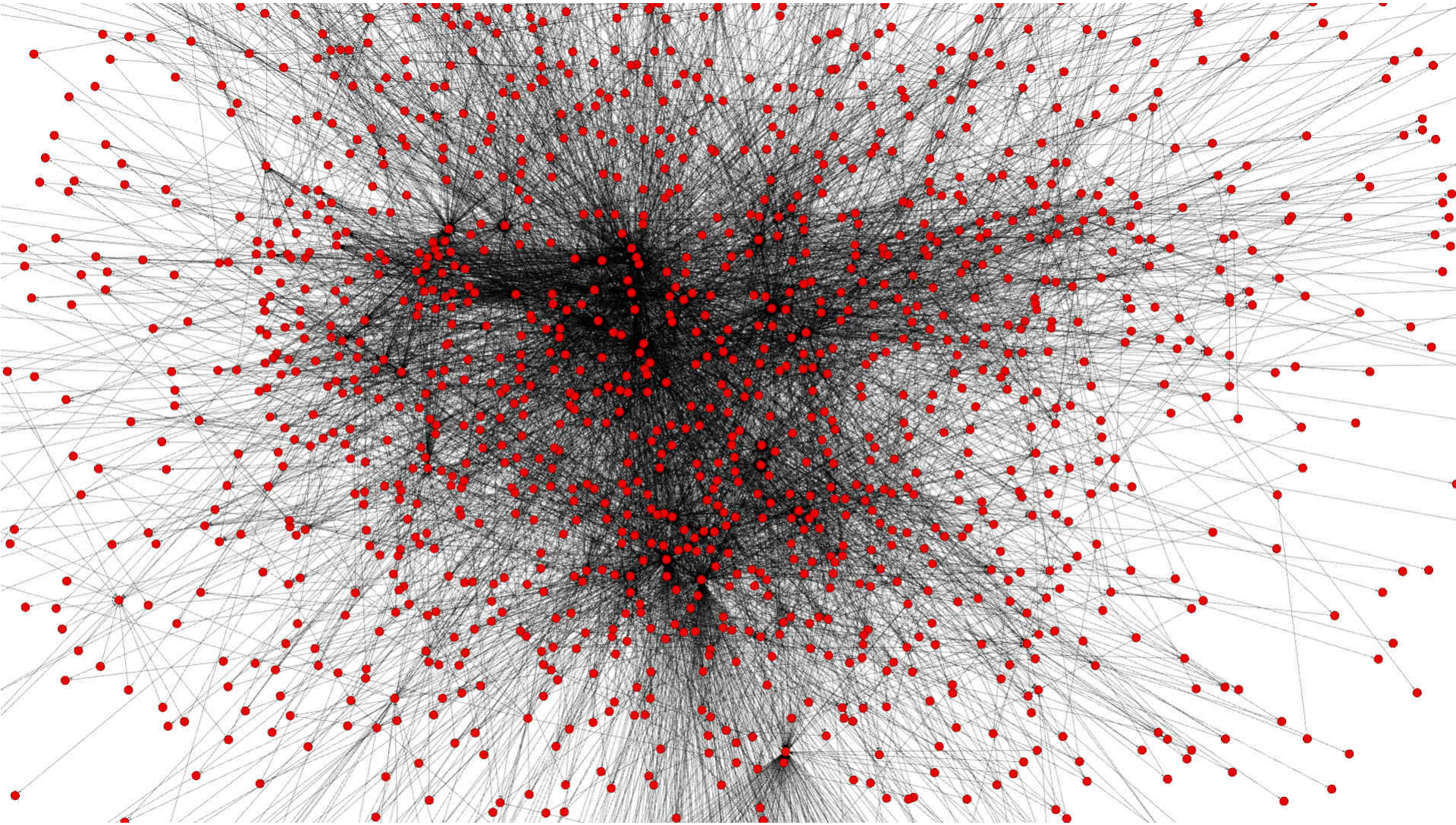
Object class causes hidden coupling

Complexity

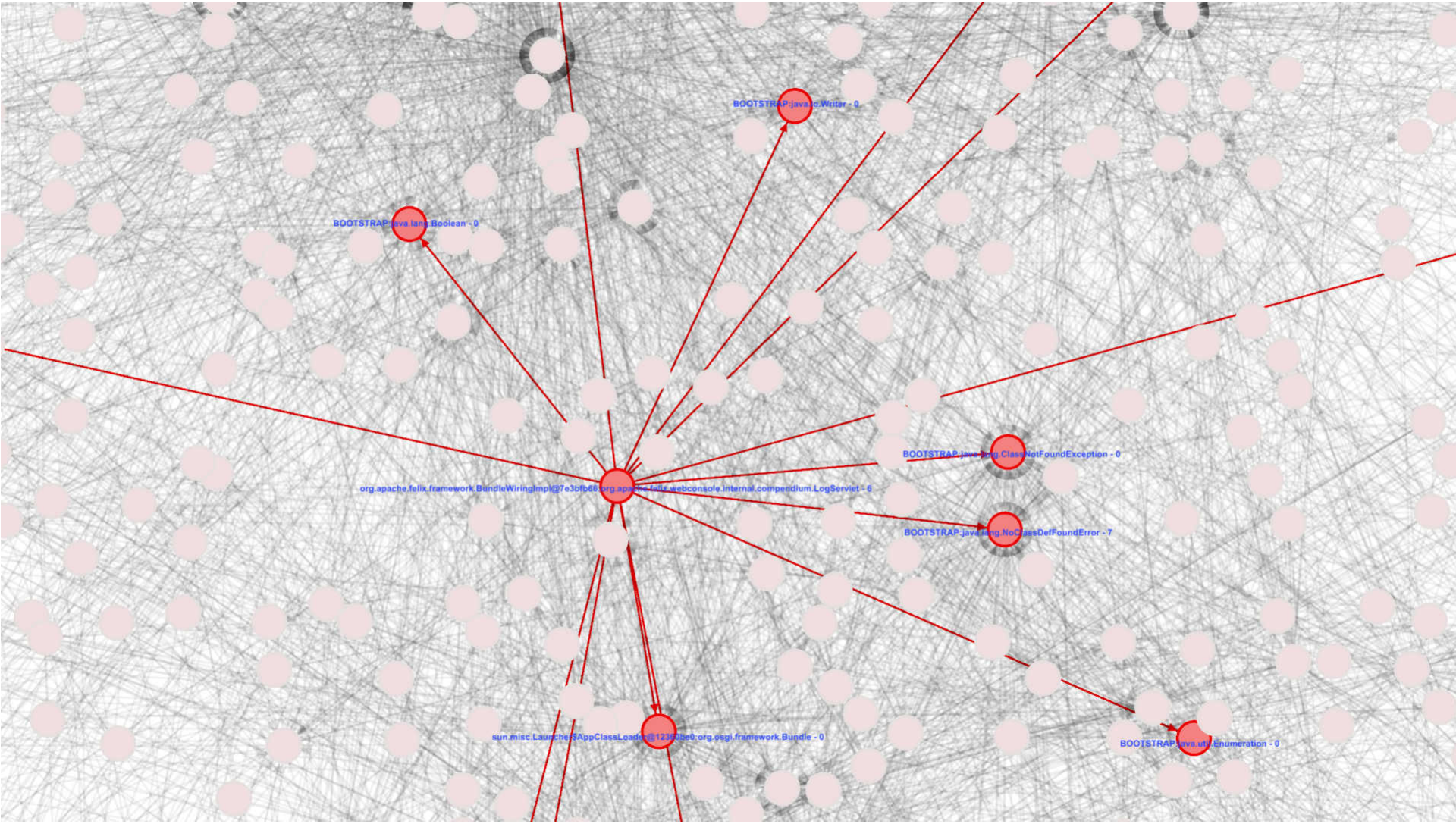
Services can be quite complex

Requires tooling

Simple application



Simple application



Outline

Context & Challenges

State of the Art

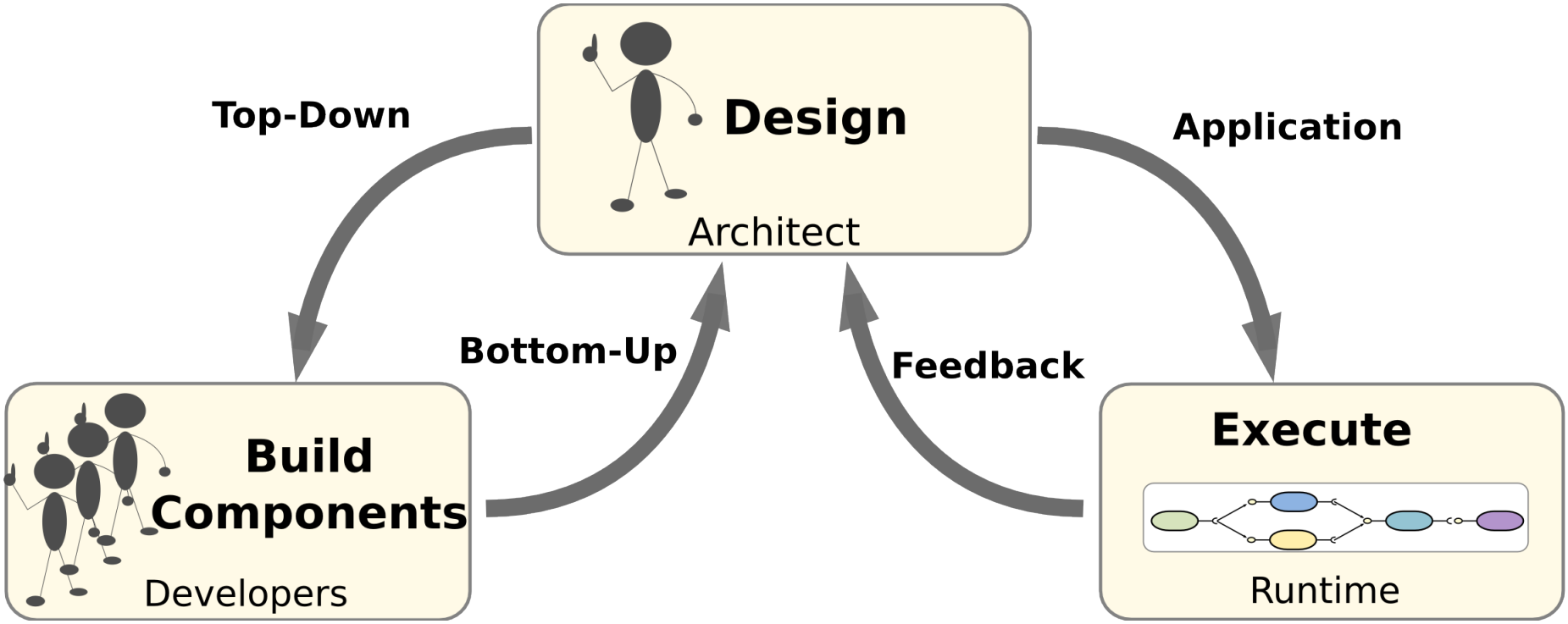
Proposition

Implementation & Validation

Conclusion & Perspectives

Robusta as an
approach to dynamism

Developing Dynamic Applications



Final remarks

Problem

Write robust centralized dynamic applications

Design

Write

Understand

Manage

Contributions

Architecture-centric approach to dynamism

Improved understanding

Coupling & Dynamism

Safe dynamism through decoupling

Proof-of-Concept (open-world)

https://github.com/rudametw/Robusta

Perspectives

Integrate into

- Felix WebConsole
- M@RT & Monitoring
- IDE
- Javascript

Impact

New methodology for dynamism?

Generalizable solution or niche?

Components ↔ Objects

Future adoption?

Thanks.

Questions?

Main Publications

Book Chapters

2011	Lionel Touseau, Kiev Gama, Didier Donsez, **Walter Rudametkin**, Adaptive and Dynamic Service Compositions in the OSGi Service Platform, chapter in book “Service Life Cycle Tools and Technologies: Methods, Trends and Advances”, Ed J. Lee, S. Ma, and Alan Liu, Publ. IGI Global, 2011.

International Conferences

2012	Joao Americo, **Walter Rudametkin** and Didier Donsez. Managing the Dynamism of Real-Time Java Applications on the OSGi Platform. In Proceedings of the 27th ACM Symposium on Applied Computing (SAC' 2012), 2012-03-26, Riva del Garda, Italy.
2011	Anthony Gelibert, **Walter Rudametkin**, Didier Donsez and Sebastien Jean, Clustering OSGi Applications using Distributed Shared Memory. In Proceedings of the 11th annual International Conference on New Technologies of Distributed Systems (NOTERE 2011), Paris, France, 9–12 May.
2010	**Walter Rudametkin**, Lionel Touseau, Didier Donsez and François Exertier. A framework for managing dynamic service-oriented component architectures. In Proceedings of the IEEE 2010 Asia-Pacific Services Computing Conference, 2010-12-06, Hangzhou, China.
2010	**Walter Rudametkin**, Kiev Gama, Lionel Touseau and Didier Donsez. Towards a Dynamic and Extensible Middleware for Enhancing Exhibits. In Proceedings of the 7th IEEE Consumer Communications & Networking Conference (CCNC'10), 2010-01-10, Las Vegas, Nevada.

Workshops

2012	Kiev Gama, **Walter Rudametkin** e Didier Donsez, "Resilience in dynamic component-based applications", III Congresso Brasileiro de Software: Teoria e Prática. SBES'2012.
2008	Kiev Gama, **Walter Rudametkin** and Didier Donsez. Using Fail-stop Proxies for Enhancing Services Isolation in the OSGi Service Platform. In Proceedings of the Workshop of the 9th International Middleware Conference 2008, 2008-12-01, Leuven, Belgium.

POT À LA MEXICAINE

Bâtiment C, Salle 005