Technologies Ain't no Architectures

Previously established approaches, concepts and procedures have been partially revised or even jeopardized by the new Java Enterprise Edition. Too complicated here, too monolithic there, and hardly ever tested. New ways emerging with Spring, Hibernate, Maven and company - tools and technologies. But where is the architecture?

Architecture

- Do you know your architecture?
  - Do you mean system architecture?
  - Do you mean software architecture?
  - Do you mean …

Somewhere, lost in a meeting…

Question: What is your architecture?

Answer: Well…

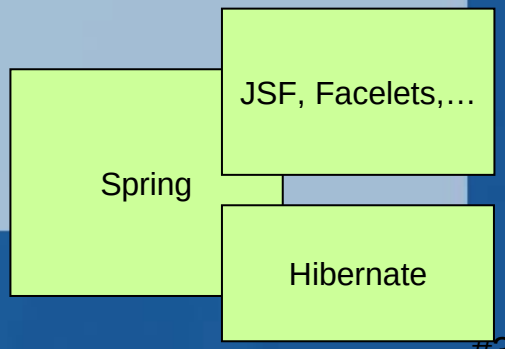
we use Java…

…

we use Spring, JSF and Hibernate…

…

Does this answer your question?



Sorry…

- The real answer to the question was Jackrabbit.
- Does not make really better…

Wikipedia: Architecture (disambiguation)

- Architecture is the art and science of design and structure.

Wikipedia: Architecture (disambiguation)

- Architecture may also refer to:
  - Technical architecture, the technical definition of an engineered system
    - **Systems architecture**, the representation of an engineered system
    - **Computer architecture**, the systems architecture of a computer
    - **Information architecture**, the systems architecture for structuring the information flows in a knowledge-based system
    - **Software architecture**, the systems architecture of a software system
    - **Hardware architecture**, the systems architecture of a hardware system

Wikipedia: Software architecture

- The software architecture of a program or computing system is the structure or structures of the system, which comprise software **components**, the externally visible properties of those components, and the **relationships** between them. The term also refers to **documentation** of a system's software architecture. Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects.

Wikipedia: System architecture

- A system architecture or systems architecture is the conceptual design or set of **relations** between the **parts** of a **system**. There is no strict definition of which aspects constitutes a system architecture, and various organizations define it in different ways, including…

So…

- Relationship
- System
- Components/ composing parts
- Structure
- Documentation

- Technologies???

Well…

Technologies

Ain‘t

no

Architecture!

Well…

Technologies

Ain‘t

no

Architecture!

Well…

Technologies

Ain‘t

no

Architecture!

Well…

Technologies

Ain‘t

no

Architecture!

or

- In other words …

Example given: the XML analogy

- The **Extensible Markup Language** (XML) is a general-purpose specification for **creating custom markup languages**. It is classified as an extensible language because it allows its users to define their own elements..

XML meta language

You can do **anything** with XML!!!

XML is **Fun**

XMI: one possible XML usage

```
<xmi:XMI xmi:version="2.1" timestamp="Sat May 27 19:48:23 CEST 2006" xmlns:uml="http://schema.omg.org/spec/UML/2.0"
        xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">
    <xmi:Documentation xmi:Exporter="MagicDraw UML" xmi:ExporterVersion="11.0"/>
    <uml:Model xmi:id="eee_1045467100313_135436_1" name="Data" visibility="public">
            <xmi:Extension xmi:Extender="MagicDraw UML 11.0" xmi:ExtenderID="MagicDraw UML 11.0">
                    <moduleExtension ignoredInModule="true"/>
            </xmi:Extension>
            <ownedComment xmi:type="uml:Comment" xmi:id="_11_0_1_8b70279_1148722236786_123286_1" body="Author:pabick&#10;Created:27.05.06
11:28.&#10;http://&#10;Comment:.|#10;" annotatedElement="eee_1045467100313_135436_1"/>
            <ownedMember xmi:type="uml:Package" xmi:id="_11_0_1_8b70279_1148722111694_682770_2" name="com.clickandbuy" visibility="public"/>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148722161245_471481_7" name="subsystem-core" visibility="public">
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148722190807_781764_30" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148722395292_63694_157" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148722358809_513951_140" supplier="_11_0_1_8b70279_1148722358809_513951_140"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148733508912_898457_546" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148733508912_180621_544" supplier="_11_0_1_8b70279_1148733508912_180621_544"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148733540788_592306_586" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148733540788_228842_584" supplier="_11_0_1_8b70279_1148733540788_228842_584"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <nestedClassifier xmi:type="uml:Interface" xmi:id="_11_0_1_8b70279_1148722190807_12161_28" name="PublishedService"
visibility="public"/>
                    <nestedClassifier xmi:type="uml:Class" xmi:id="_11_0_1_8b70279_1148733508912_180621_544" visibility="public"/>
                    <nestedClassifier xmi:type="uml:Class" xmi:id="_11_0_1_8b70279_1148733540788_228842_584" name="DOM" visibility="public"/>
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148722217446_942875_60" visibility="public"
contract="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
            </ownedMember>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148722264834_475474_73" name="subsystem-impl" visibility="public">
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148733557612_907435_606" visibility="public"
contract="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148722264834_475474_73"/>
            </ownedMember>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148722312903_528399_94" visibility="public"/>
            <ownedMember xmi:type="uml:Dependency" xmi:id="_11_0_1_8b70279_1148722312923_409077_96" visibility="public"
supplier="_11_0_1_8b70279_1148722312903_528399_94" client="_11_0_1_8b70279_1148722264834_475474_73"/>
            <ownedMember xmi:type="uml:Artifact" xmi:id="_11_0_1_8b70279_1148722358809_513951_140" visibility="public"/>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148732049524_447560_277" name="subsystem-serviceExporter"
visibility="public">
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148736855494_61906_1426" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148736855494_900019_1424" supplier="_11_0_1_8b70279_1148736855494_900019_1424"
client="_11_0_1_8b70279_1148732049524_447560_277"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148736866450_127377_1447" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148736866440_37201_1445" supplier="_11_0_1_8b70279_1148736866440_37201_1445"
client="_11_0_1_8b70279_1148732049524_447560_277"/>
                    <nestedClassifier xmi:type="uml:Class" xmi:id="_11_0_1_8b70279_1148736855494_900019_1424" name="DTO" visibility="public"/>
                    <nestedClassifier xmi:type="uml:Interface" xmi:id="_11_0_1_8b70279_1148736866440_37201_1445"
name="PublishedRemoteDTOService" visibility="public"/>
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148732140745_117398_312" visibility="public"
contract="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148732049524_447560_277"/>
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148732270101_889829_352" visibility="public"
```

XMI XML…

- It is not really extensible: it was designed to describe UML diagrams
  - Preferable not in a text editor…
- Definitively no fun
- Creativity?

Again... XMI XML

```
<xmi:XMI xmi:version="2.1" timestamp="Sat May 27 19:48:23 CEST 2006" xmlns:uml="http://schema.omg.org/spec/UML/2.0"
        xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">
    <xmi:Documentation xmi:Exporter="MagicDraw UML" xmi:ExporterVersion="11.0"/>
    <uml:Model xmi:id="eee_1045467100313_135436_1" name="Data" visibility="public">
            <xmi:Extension xmi:Extender="MagicDraw UML 11.0" xmi:ExtenderID="MagicDraw UML 11.0">
                    <moduleExtensionIgnoredModules value="true"/>
            </xmi:Extension>
            <ownedComment xmi:type="uml:Comment" xmi:id="_11_0_1_8b70279_1148722106536_123286_1" body="Author:papick.&#10;Created:27.05.06
11:28.&#10;Title:.&#10;Comment:.&#10;" annotatedElement="eee_1045467100313_135436_1"/>
            <ownedMember xmi:type="uml:Package" xmi:id="_11_0_1_8b70279_1148722111694_682770_2" name="com.clickandbuy" visibility="public"/>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148722161245_471481_7" name="subsystem-core" visibility="public">
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148722190807_781764_30" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148722395292_63694_157" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148722358809_513951_140" supplier="_11_0_1_8b70279_1148722358809_513951_140"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148733508912_898457_546" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148733508912_180621_544" supplier="_11_0_1_8b70279_1148733508912_180621_544"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148733540788_592306_586" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148733540788_228842_584" supplier="_11_0_1_8b70279_1148733540788_228842_584"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
                    <nestedClassifier xmi:type="uml:Interface" xmi:id="_11_0_1_8b70279_1148722190807_12161_28" name="PublishedService"
visibility="public"/>
                    <nestedClassifier xmi:type="uml:Class" xmi:id="_11_0_1_8b70279_1148733508912_180621_544" visibility="public"/>
                    <nestedClassifier xmi:type="uml:Class" xmi:id="_11_0_1_8b70279_1148733540788_228842_584" name="DOM" visibility="public"/>
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148722217446_942875_60" visibility="public"
contract="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148722161245_471481_7"/>
            </ownedMember>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148722264834_475474_73" name="subsystem-impl" visibility="public">
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148733557612_907435_606" visibility="public"
contract="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148722264834_475474_73"/>
            </ownedMember>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148722312903_528399_94" visibility="public"/>
            <ownedMember xmi:type="uml:Dependency" xmi:id="_11_0_1_8b70279_1148722312923_409077_96" visibility="public"
supplier="_11_0_1_8b70279_1148722312903_528399_94" client="_11_0_1_8b70279_1148722264834_475474_73"/>
            <ownedMember xmi:type="uml:Artifact" xmi:id="_11_0_1_8b70279_1148722358809_513951_140" visibility="public"/>
            <ownedMember xmi:type="uml:Component" xmi:id="_11_0_1_8b70279_1148732049524_447560_277" name="subsystem-serviceExporter"
visibility="public">
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148736855494_61906_1426" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148736855494_900019_1424" supplier="_11_0_1_8b70279_1148736855494_900019_1424"
client="_11_0_1_8b70279_1148732049524_447560_277"/>
                    <realization xmi:type="uml:Realization" xmi:id="_11_0_1_8b70279_1148736866450_127377_1447" visibility="public"
realizingClassifier="_11_0_1_8b70279_1148736866440_37201_1445" supplier="_11_0_1_8b70279_1148736866440_37201_1445"
client="_11_0_1_8b70279_1148732049524_447560_277"/>
                    <nestedClassifier xmi:type="uml:Class" xmi:id="_11_0_1_8b70279_1148736855494_900019_1424" name="DTO" visibility="public"/>
                    <nestedClassifier xmi:type="uml:Interface" xmi:id="_11_0_1_8b70279_1148736866440_37201_1445"
name="PublishedRemoteDTOService" visibility="public"/>
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148732140745_117398_312" visibility="public"
contract="_11_0_1_8b70279_1148722190807_12161_28" supplier="_11_0_1_8b70279_1148722190807_12161_28"
client="_11_0_1_8b70279_1148732049524_447560_277"/>
                    <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="_11_0_1_8b70279_1148732270101_889829_352" visibility="public"
```

XMI XML

- no…
  - Creativity
  - No fun

So where is the analogy?

- Technologies are meta languages
  - With (new) technologies you can do anything
  - Technologies are fun (well, at least new ones)
  - You can (must) be creative when applying new technologies
- The architecture is where it stops being fun…

J two EE

- Architecture follows technology
- „One size fits all“
- „All in one“
  - Architecture
  - Programming model
  - Infrastructure / runtime
  - Technologies used

J five EEEEE

- Architecture still follows technology
- Still „One size fits all“
- Still „All in one“
  - Architecture
  - Programming model
  - Infrastructure / runtime
  - Technologies used

Ei-Jay-En-Gee [what?]

- EJNG: Enterprise Java Next Generation
  - DI/ IOC/ KIS(S)/ DRY/ POJOs
  - Lightweight container
  - Minimal invasive frameworks
  - Ajaxified web UIs
  - J2EE without EJB
  - Testing, continuous integration

Ei-Jay-En-Gee is soooo cooool

- Many cool technologies and cool frameworks
- Few restrictions
- Just do it
- Plenty room for new ideas, creativity, fun

Ei-Jay-En-Gee in practice

- No technology line up
- Best practices restricted to layers
- No big picture, no architecture definitions

Responsabiliy++

- Know and understand reference architectures
  - Why and Why nots
  - Dos and Donts
- Know your requirements
- Architecture must be
  - Viable, communicable, transferrable, placeable

Architectures should provide solutions to the presented problems, should not create new ones!

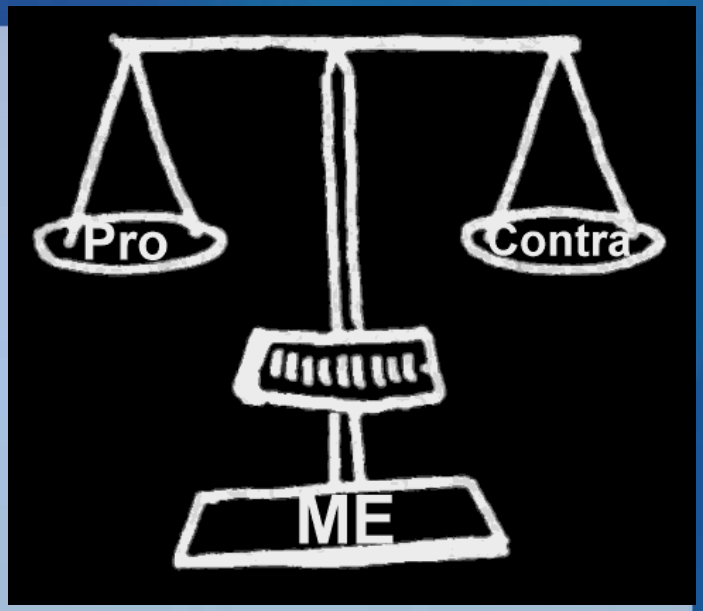
Architectures must be documented

- 5 boxes in powerpoint looks good, but it not even close do something that could be called a documentation
- No one reads 500 pages in Word
- Wiki vs. Word?
  - Good: no structure restrictions, history, easy publishing, multi user
  - Bad: wild grown wiki spaces, old pages

Common ingredients?

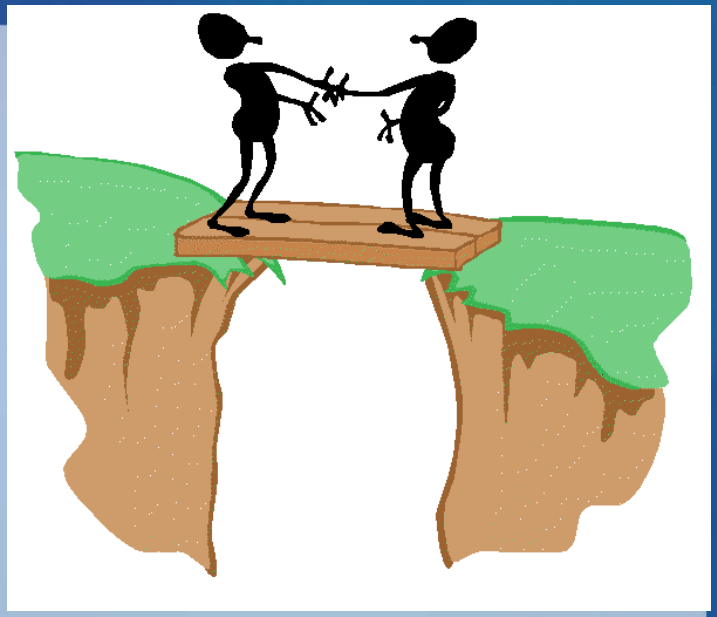
- Are there commons in reference architectures?
  - CBD
  - SOA
  - EDA
- Is performance an ingredient?

Neutrality?



- Should technologies be neutral to technologies?
- How do we know which technologies will fit the architecture?

Who fills the gap?



- The not knowing technologies architect?
- The not understanding the architecture developers?

Dead end?

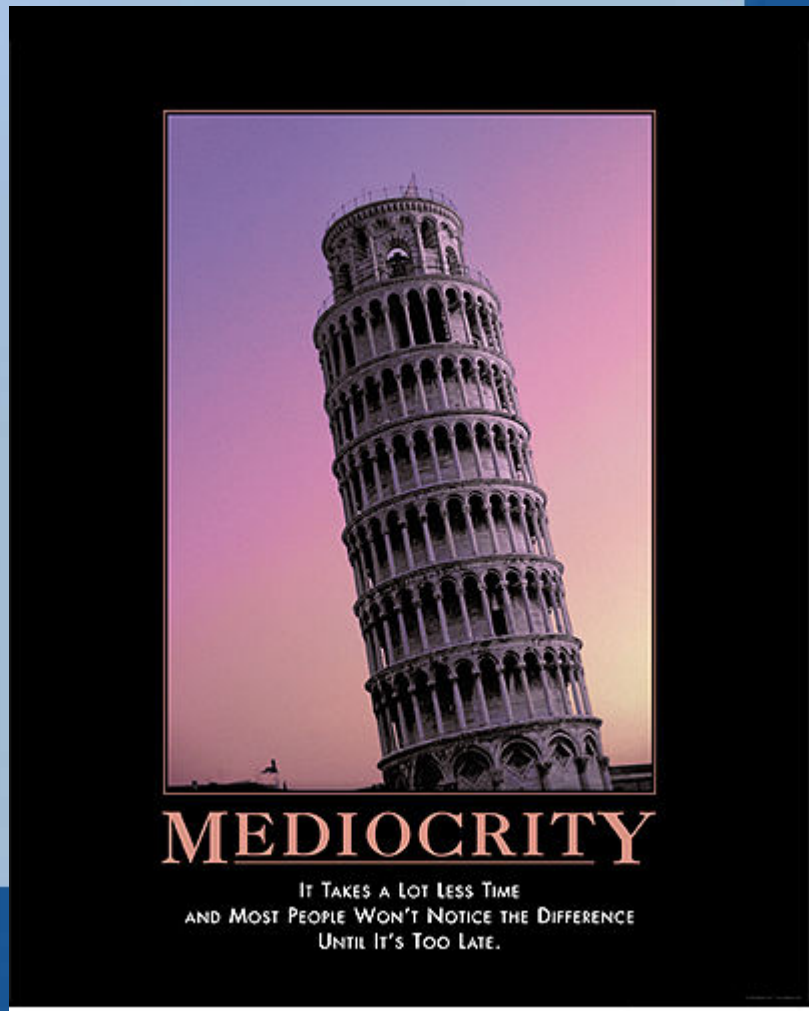
- Abstract architectures must be implemented (technology spike)

- Not a common skill for „plain vanilla“ software architect

Software development two-dot-oh?

Project Danger!

It takes a lot less time and most people won‘t notice the difference until it‘s too late…



Source: http://www.despair.com

Super software architect needed

Finally, something to do!

- Define the architecture
- Line up technologies and concepts
- Define process
- Introduce tooling
- Execute coaching and training
- Inspire team and establish best practices

Image: http://www.kotaku.com/gaming/new-superman-suit_lg.jpg

Brew my own architecture?

What ingredients do I need to brew my own architecture?

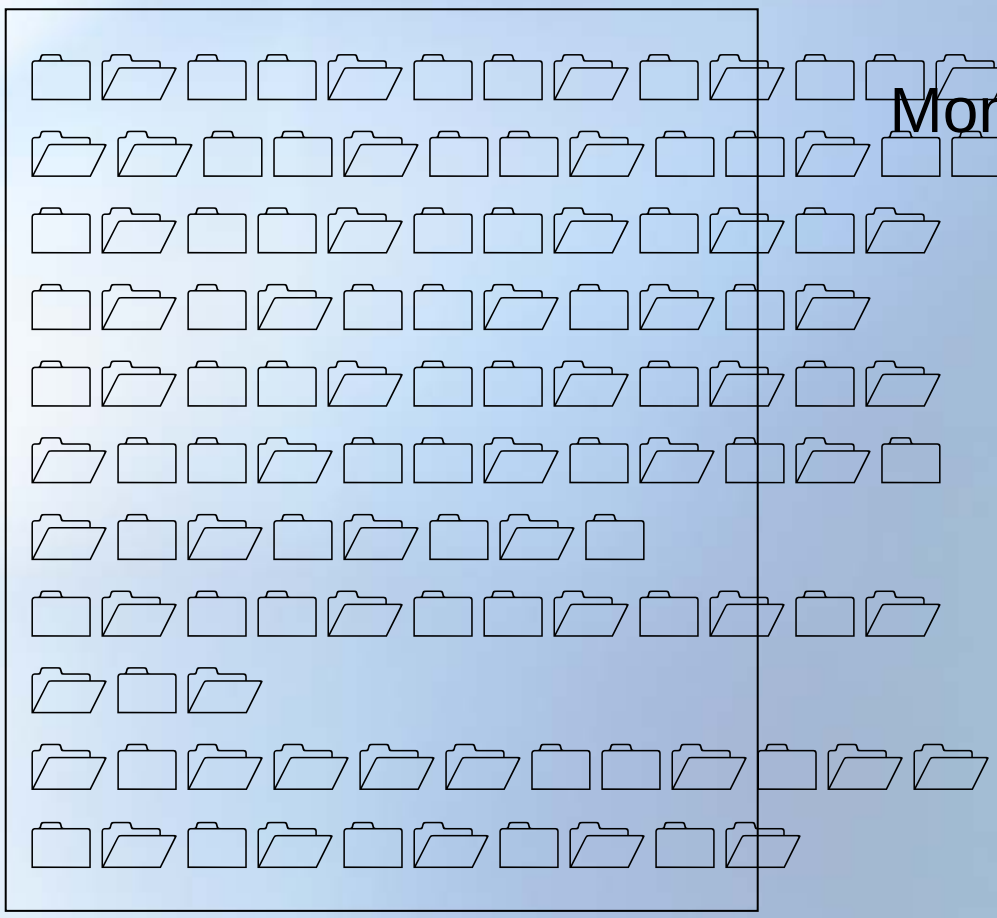
Divide and conquer

- Components
- Layers
- Modules
- Aspects?

Components…

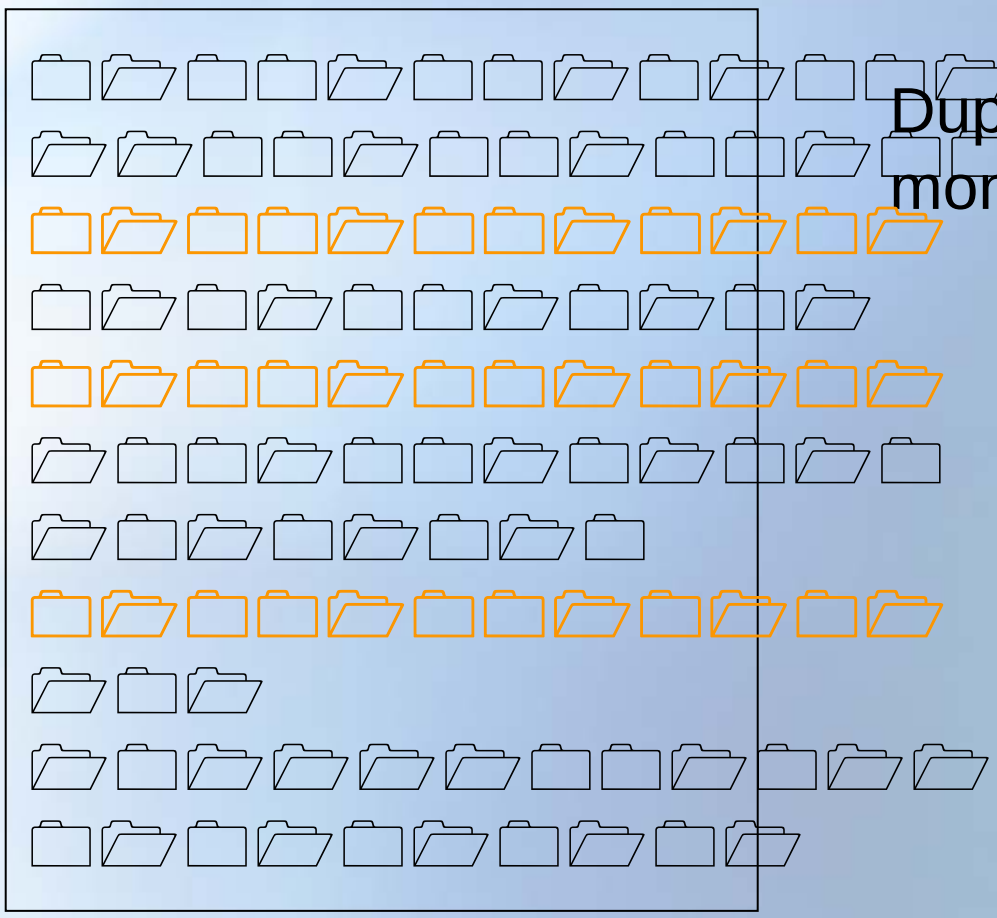
What are components?

Creating components…



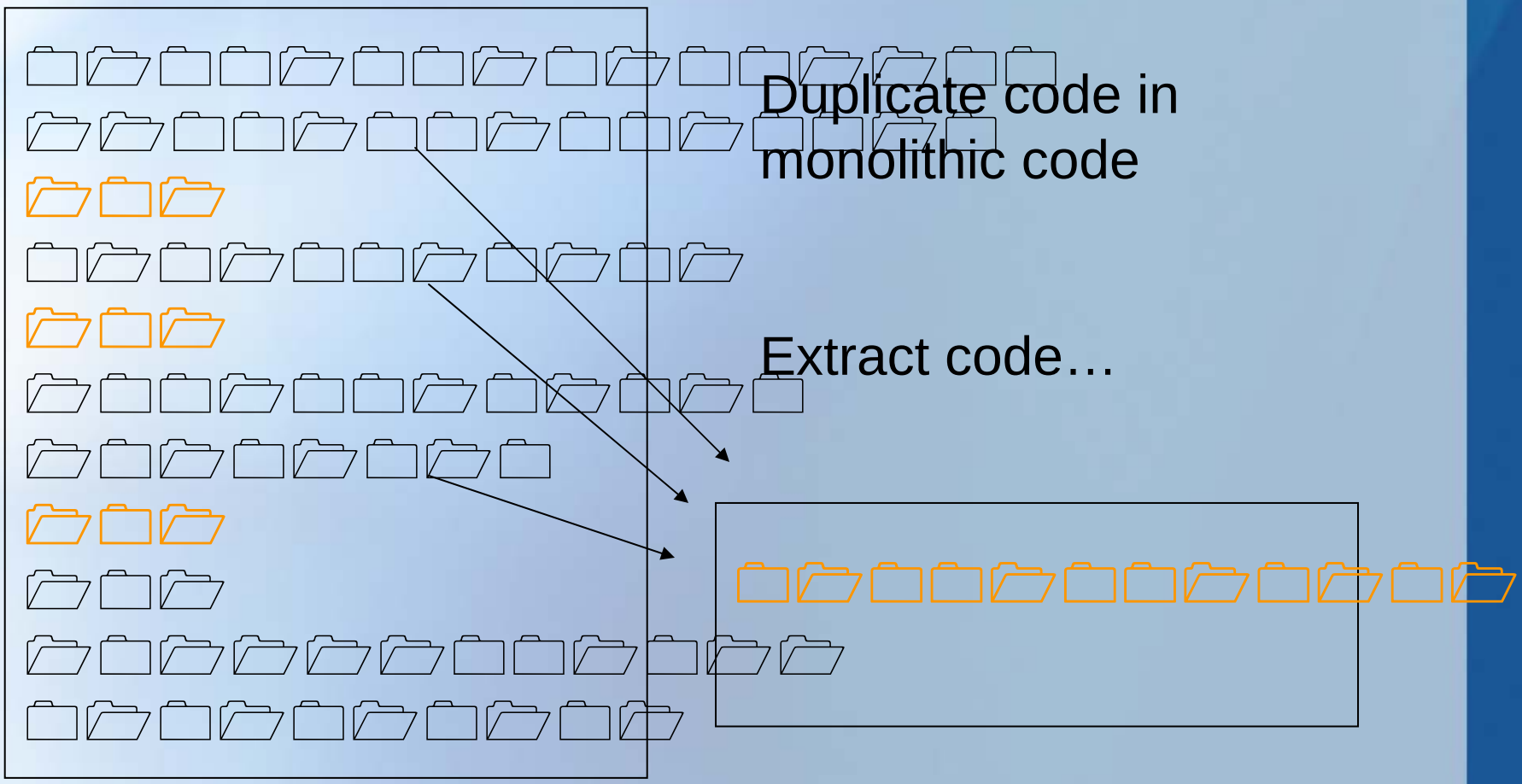
Monolithic code

Creating components…

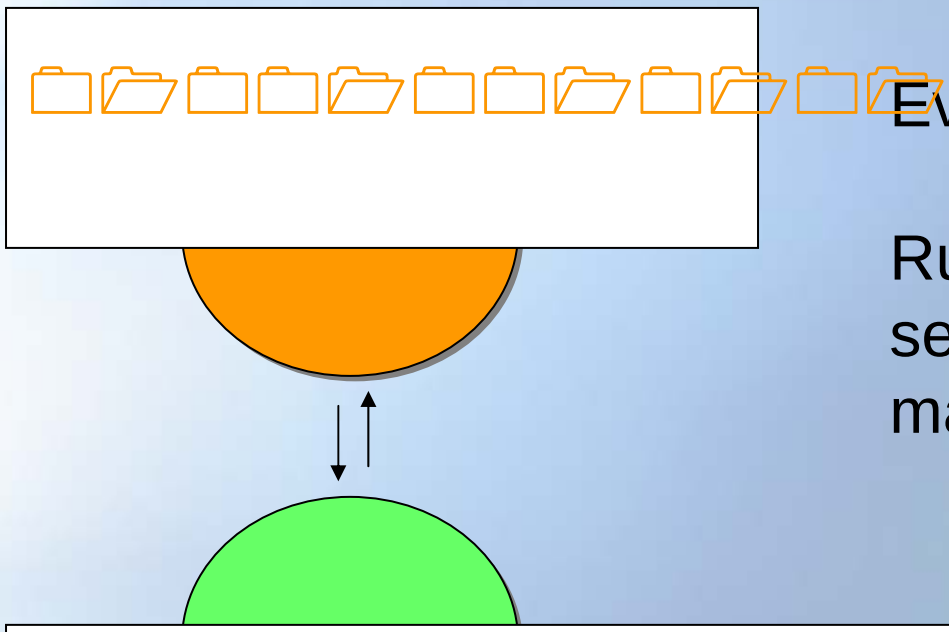


Duplicate code in monolithic code

Komponentenbildung…



Runtime

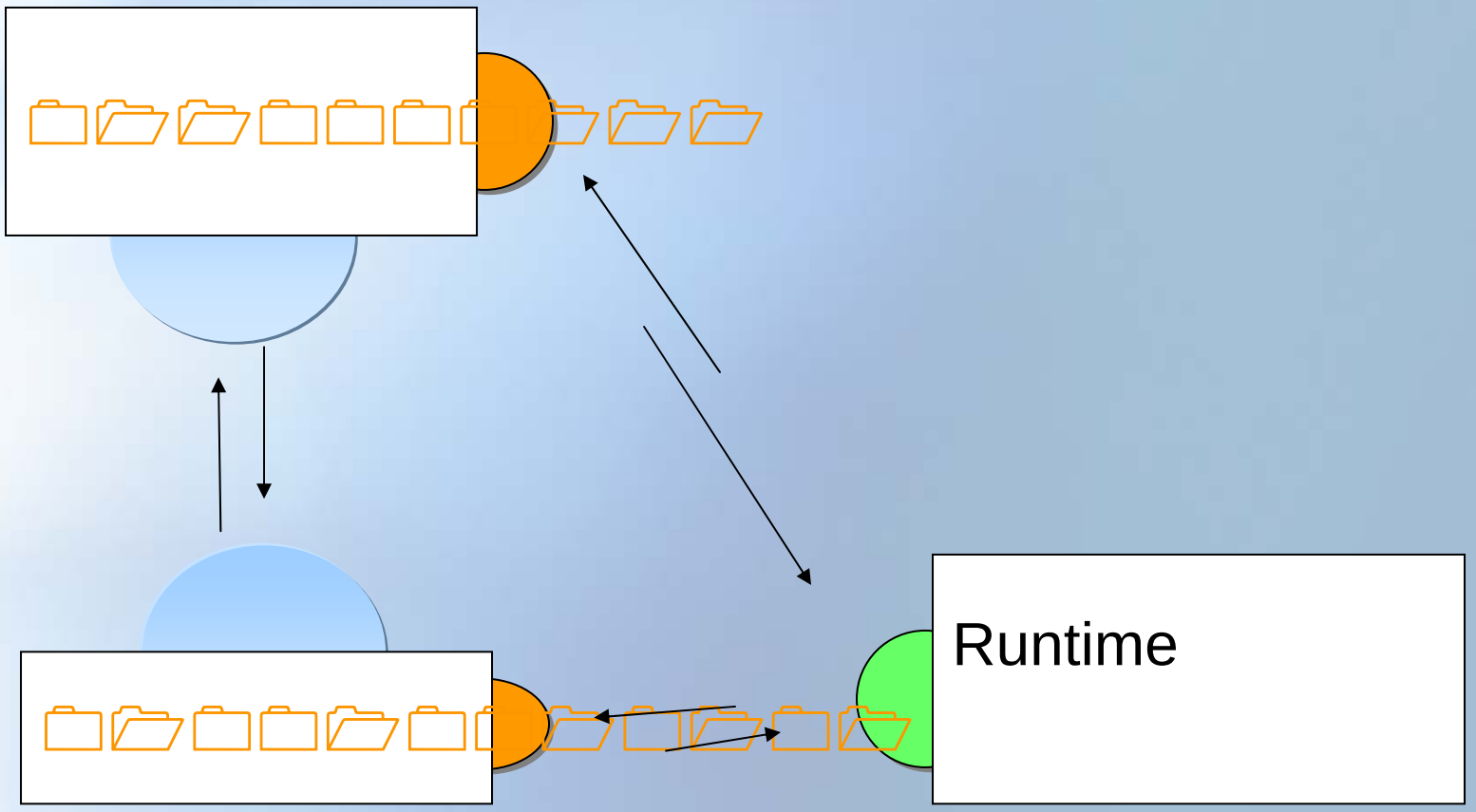


Evolution to a component:

Runtime provides common services and lifecycle management

Runtime
Common services, component lifecycle management

Component based systems



It‘s just the beginning…

- Does a component have state?
  - Activation/ passivation
- Complexityof lifecycle?
- Dependency
  - Mediation
  - Resolution
- Common services
- …

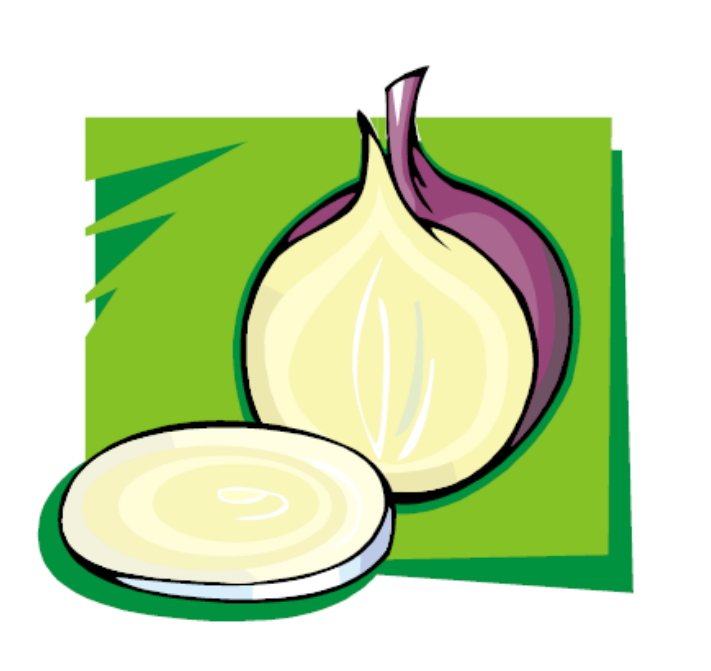
Layers

Layers in the system architecture

It-What-Must-Not-Be-Named

The evil has
many names.
But there is evil that
is so feared in the
development
world that it is
considered
dangerous even
to speak its name.
Most developers
in the world refer
to it as "You-Know-What"
or „It-What-Must-Not-Be-
Named" rather than
saying its name aloud.

Onion-software

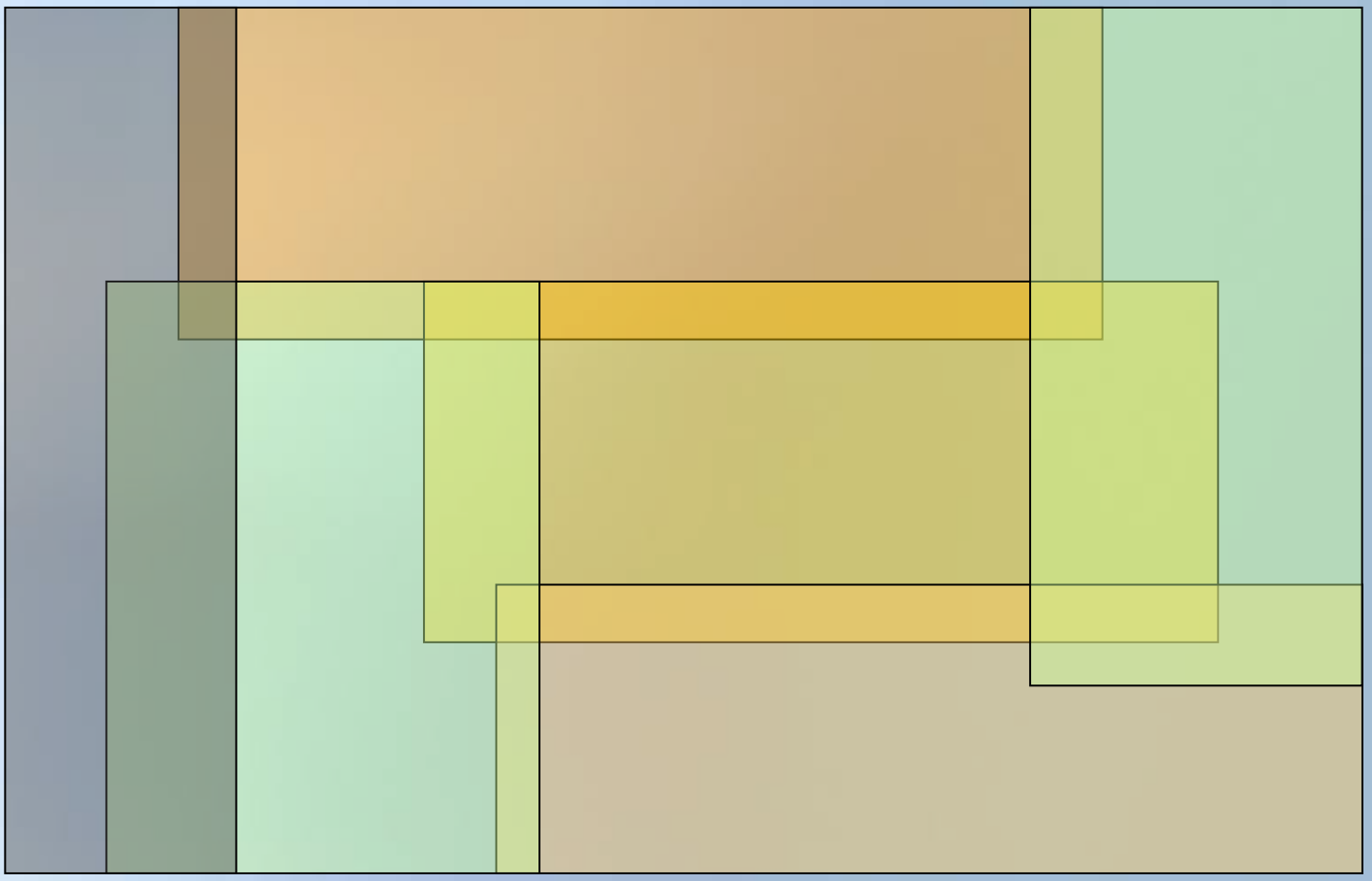


Presentation Layer

Business Logic Layer

Persistence Layer

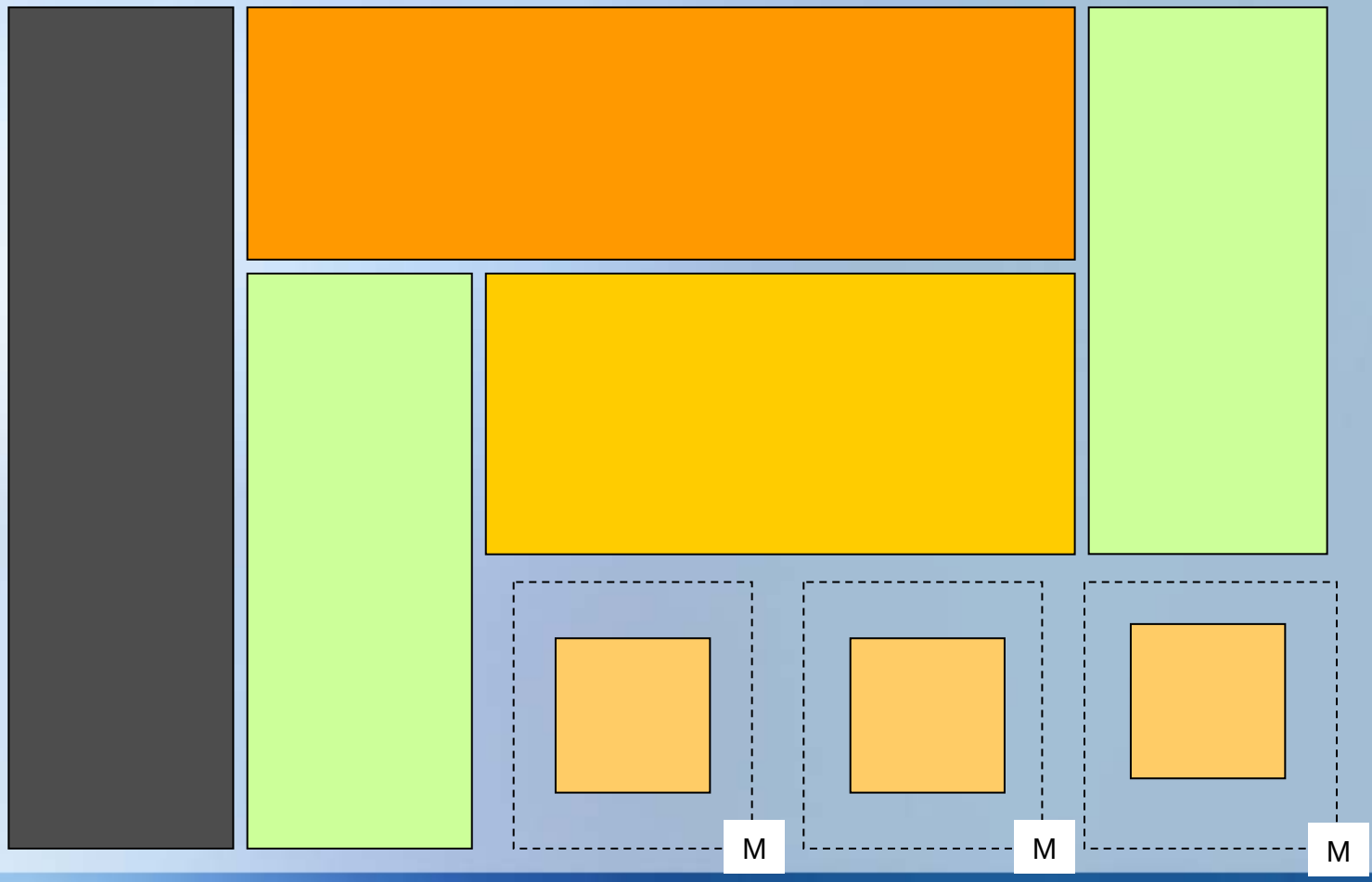
Layers in practice…



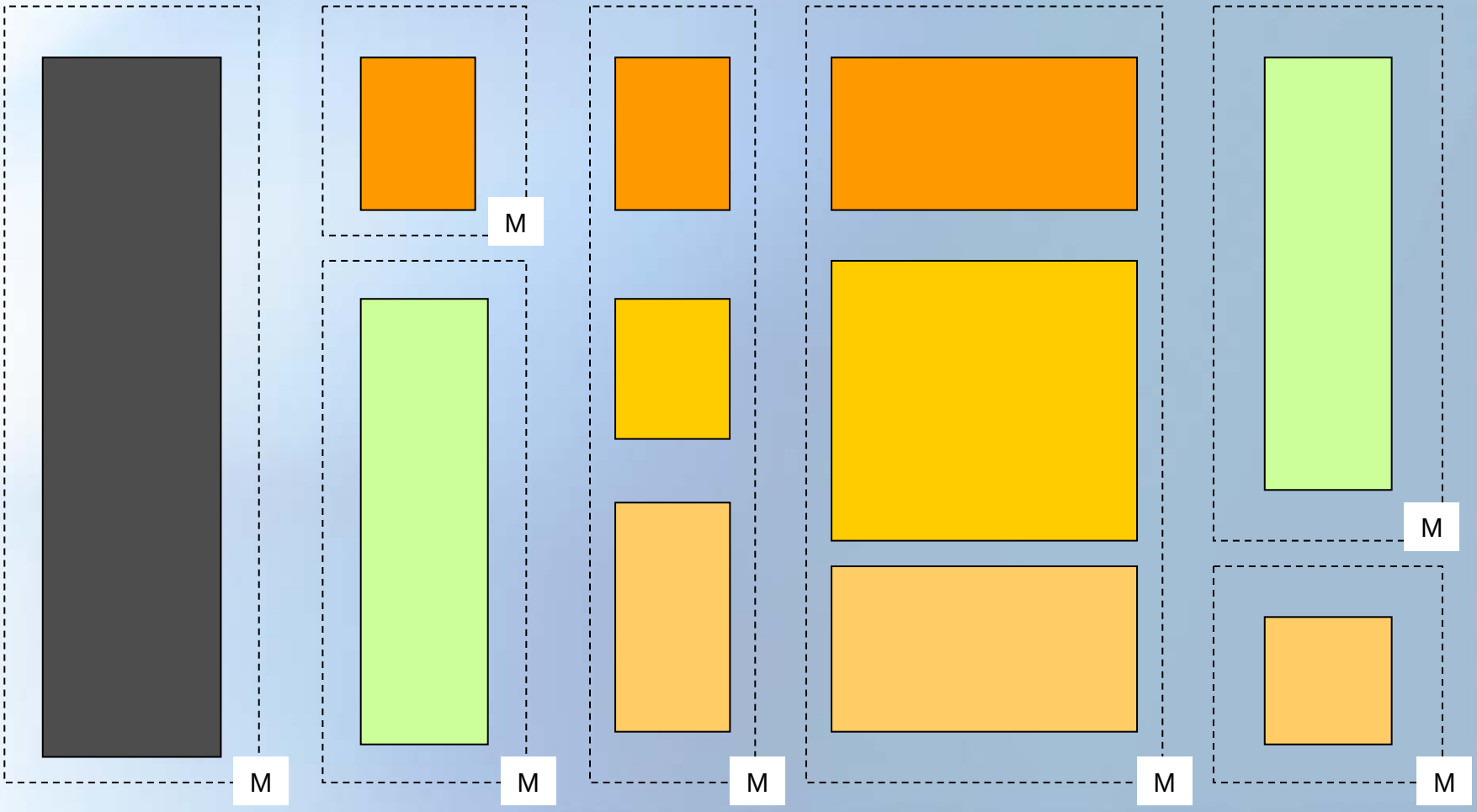
Modules

A module is a self-contained component of a system, which has a well-defined interface to the other components; something is modular if it includes or uses modules which can be interchanged as units without disassembly of the module. Design, manufacture, repair, etc. of the modules may be complex, but this is not relevant; once the module exists, it can easily be connected to or disconnected from the system.

Modules and layers, #1



Modules and layers, #2



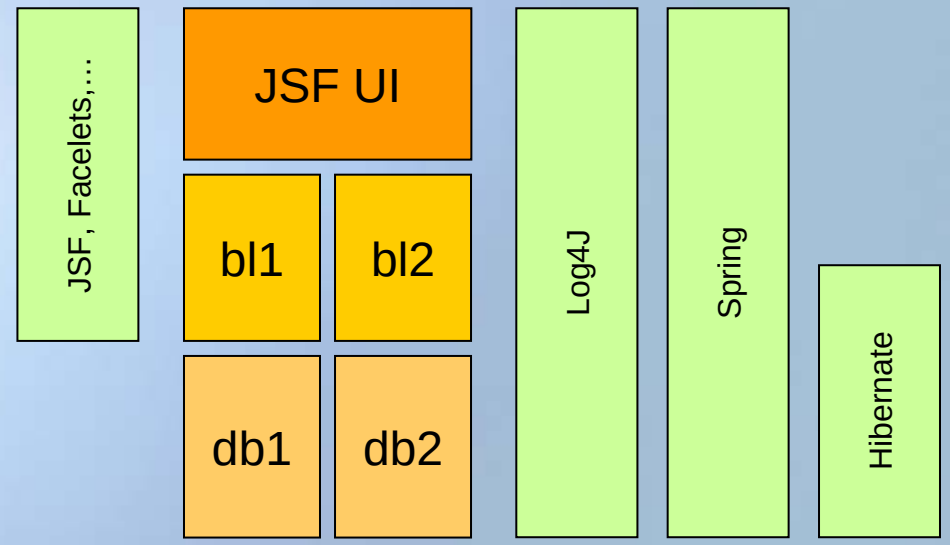
Chaos

Only the true genius masters the chaos – Albert Einstein

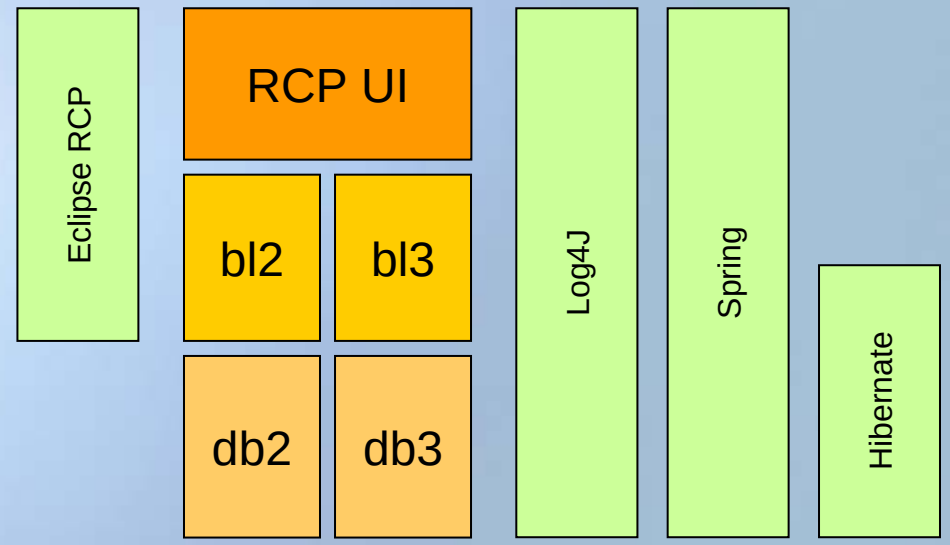
A little example project…

- A „simple“ WebApp „A“
  - JSF/ Facelets UI
  - Business logic „bl1“ and „bl2“
  - Persistence „db1“ and „db2“
- A little more complex rich client „B“
  - RCP Client
  - Business logic „bl2“ and „bl3“
  - Persistence „db2“ and „db3“

Simple outline – WebApp A



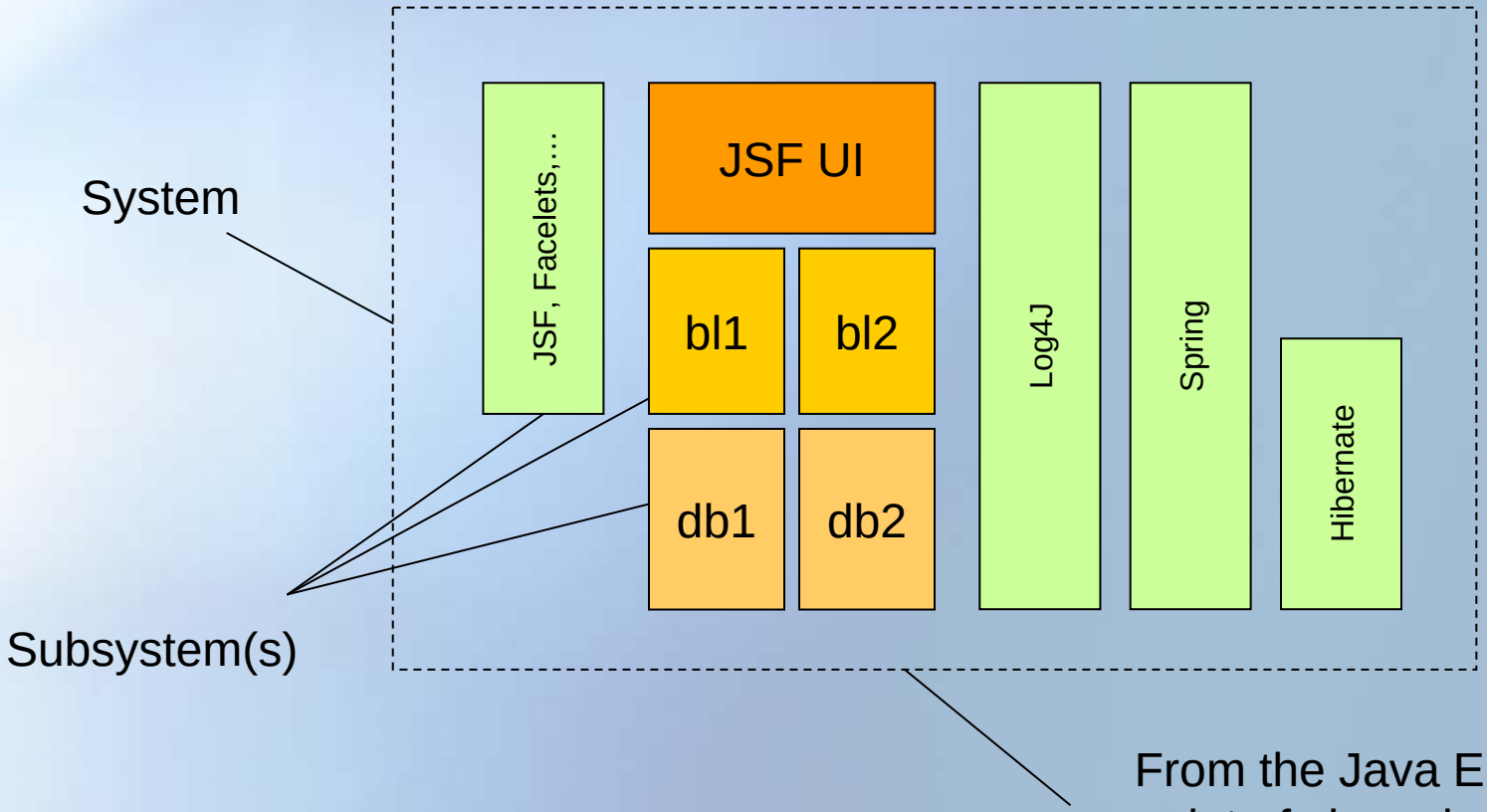
Simple outline – Rich client B



Would it hurt to know more?

- Too imprecise?
- Too few information?

Web A: boundaries and names



Naming the pieces

System,
Subsystem,
Module

Versions

Public vs. published interfaces

Communication

Fine and coarse grained components

- System
- Subsystem
- Module
  - Smallest composing part
  - Will be to mapped
    - Maven artifacts
    - IDE project

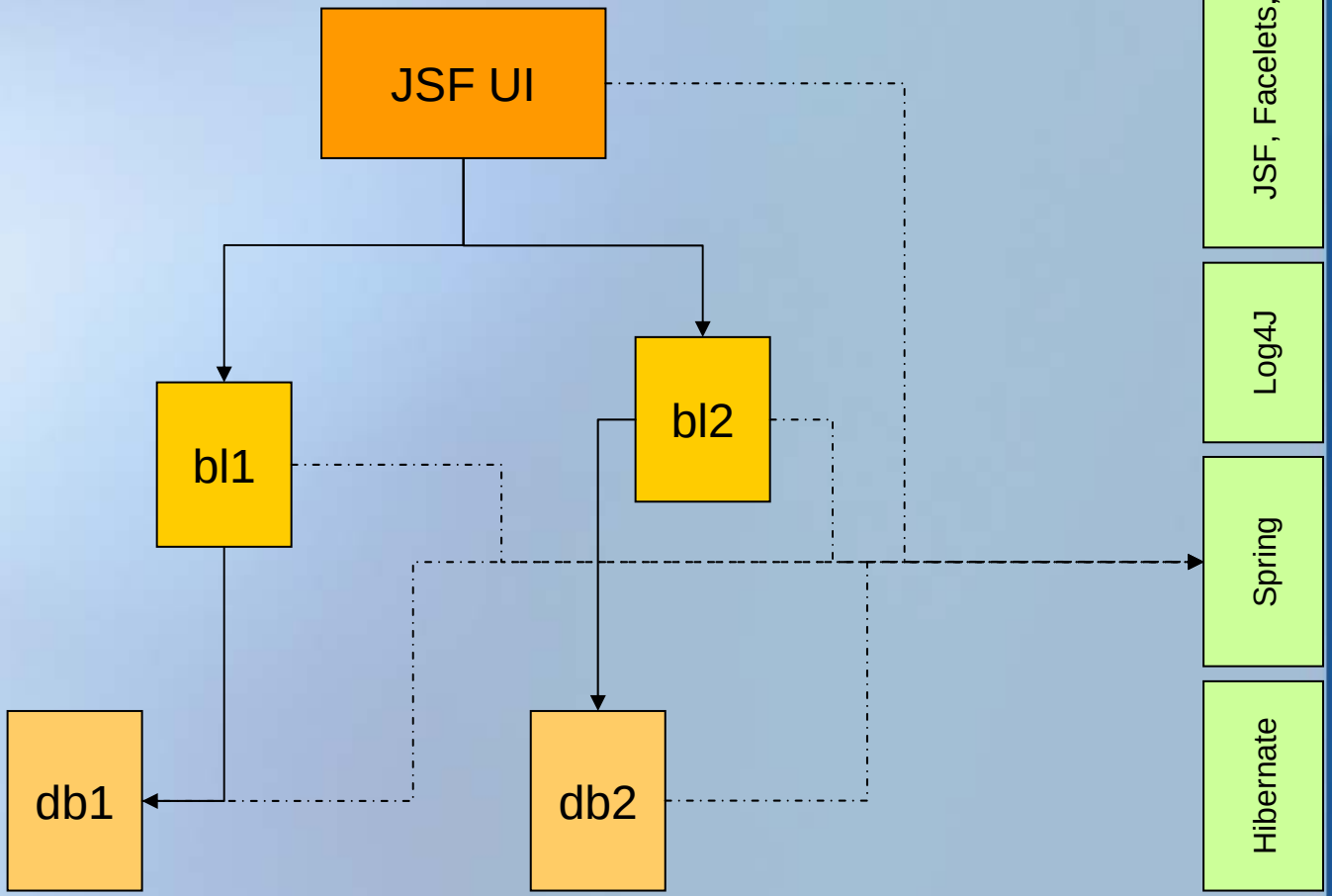
The term „component“ is used by many technologies. To avoid misinterpretation I do not use it here…

Web A: we can do better!

Dependencies!

Between modules

Between modules and required technologies



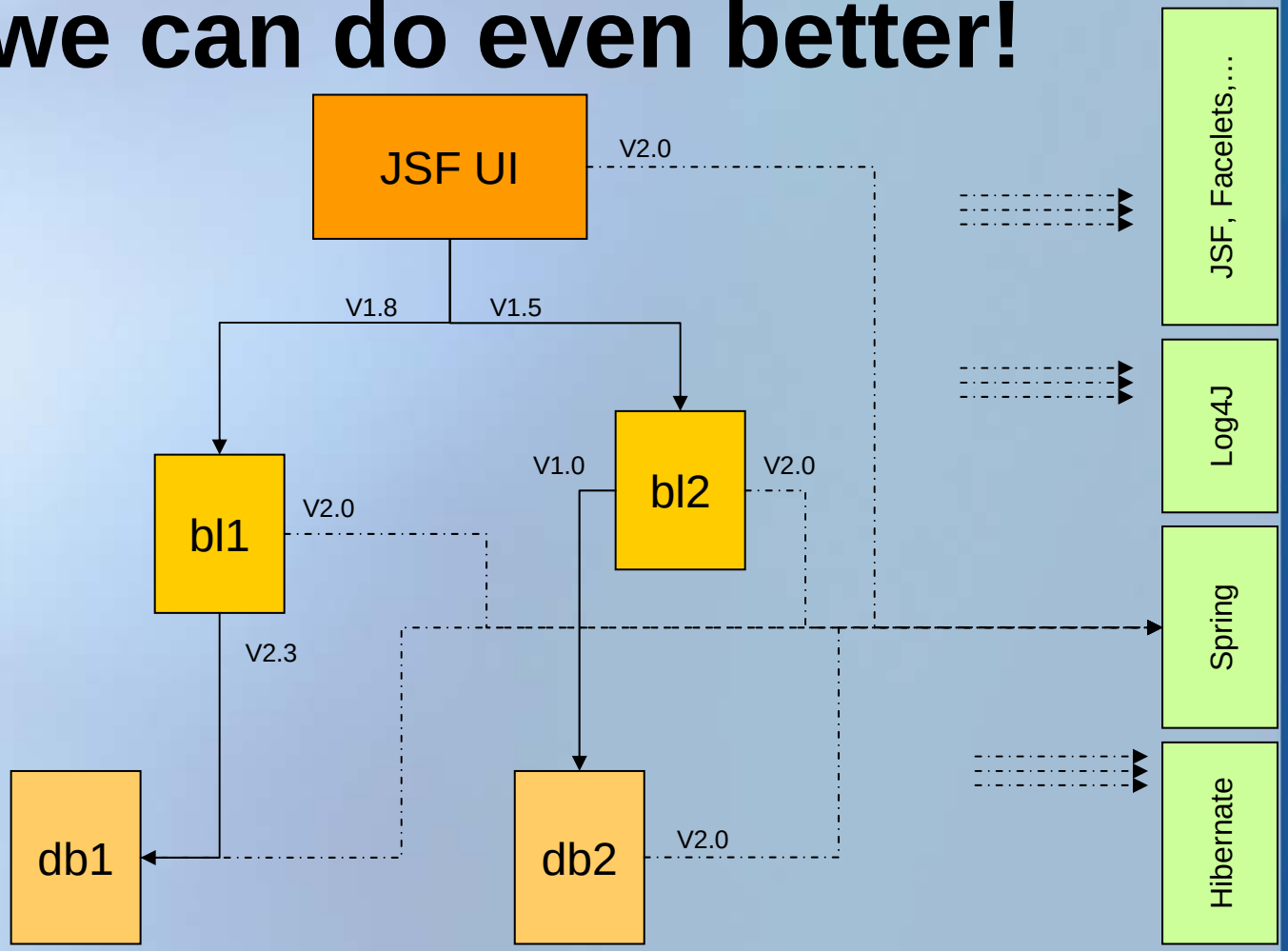
Web A: we can do even better!

Dependencies!

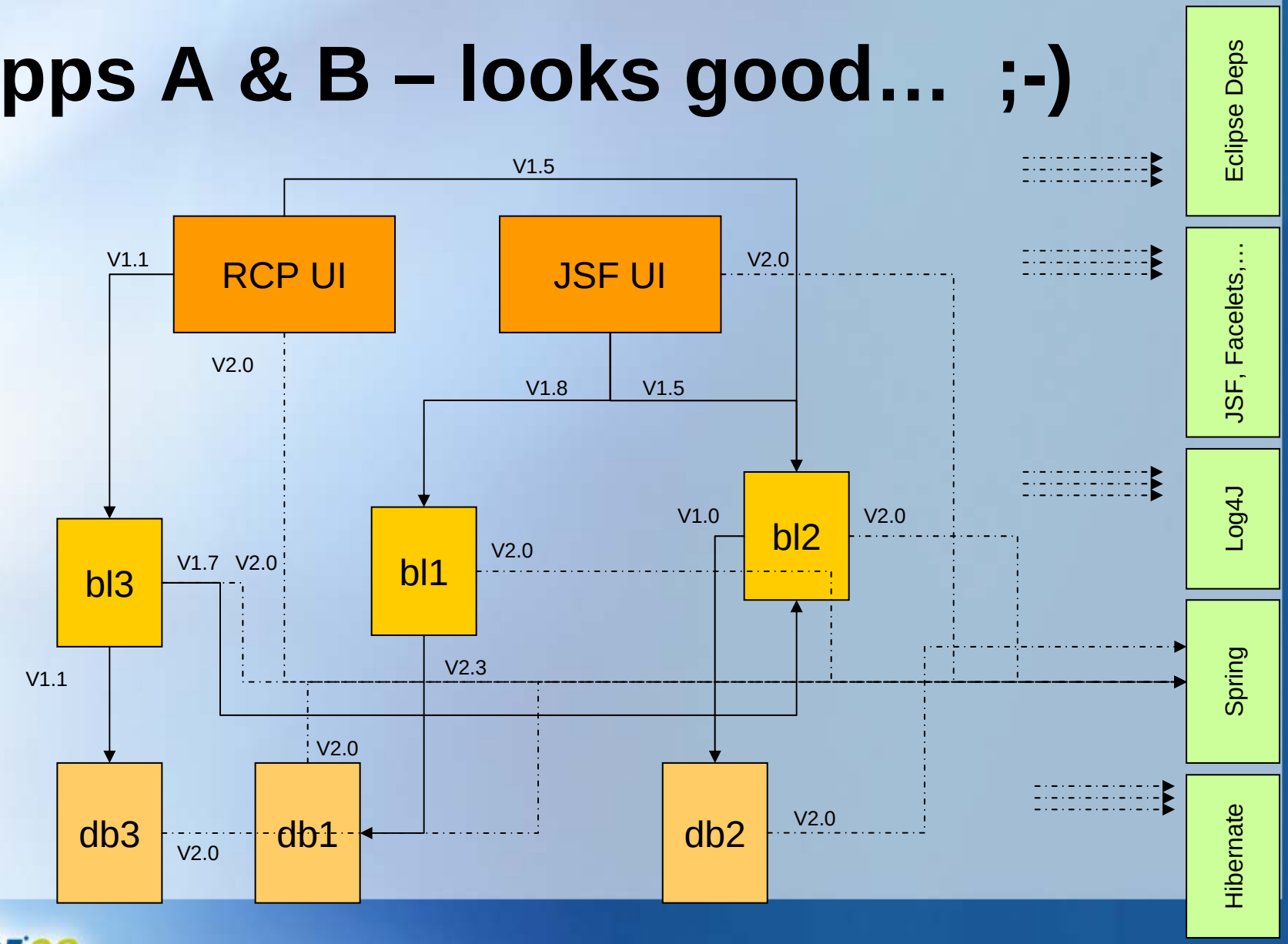
Between modules

Between modules and required technologies

Versions!!!



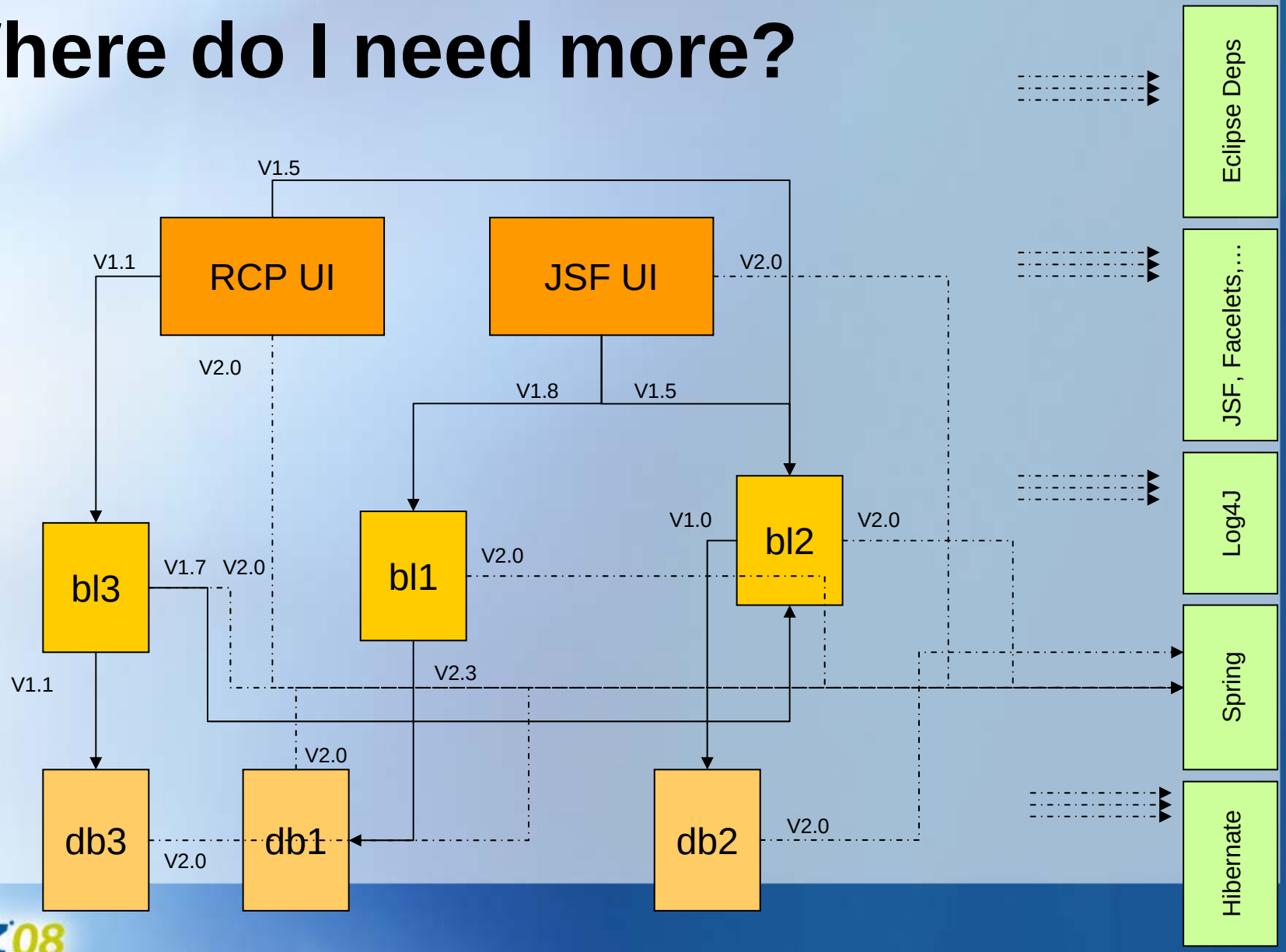
Apps A & B – looks good… ;-)



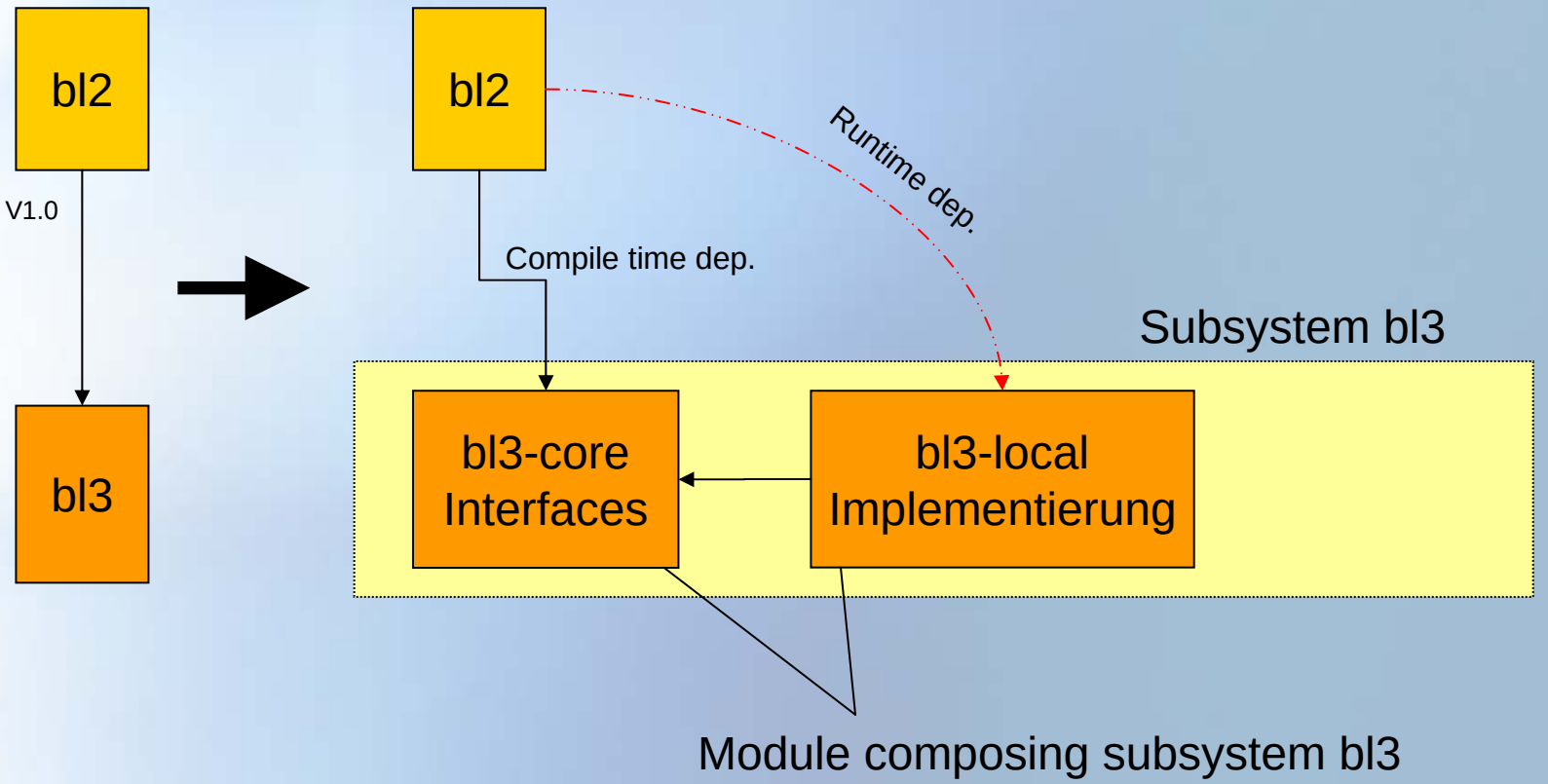
Would it hurt do have even more?

- YES!
- But…
- Still too imprecise?
- Still too few information?

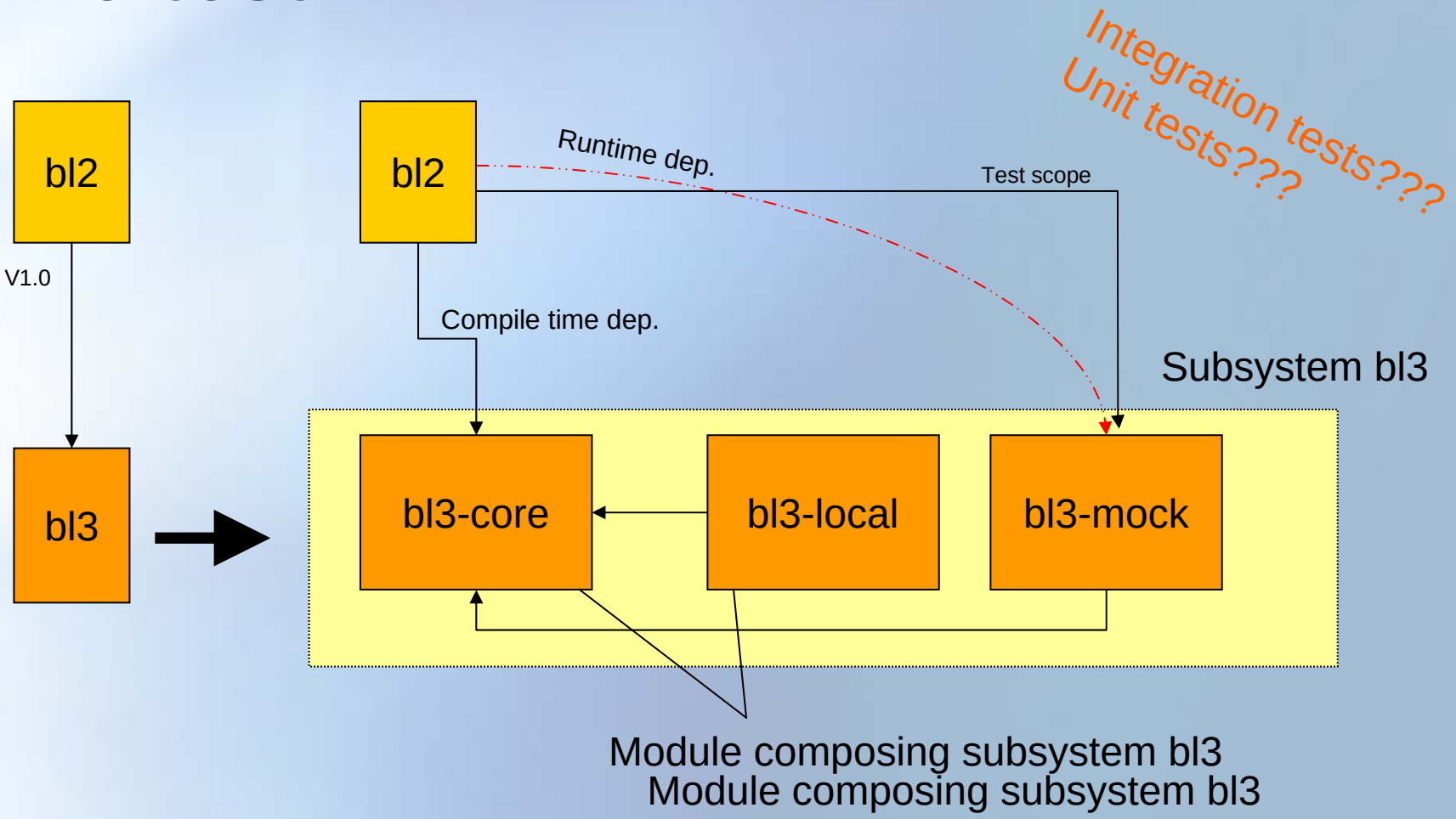
Where do I need more?



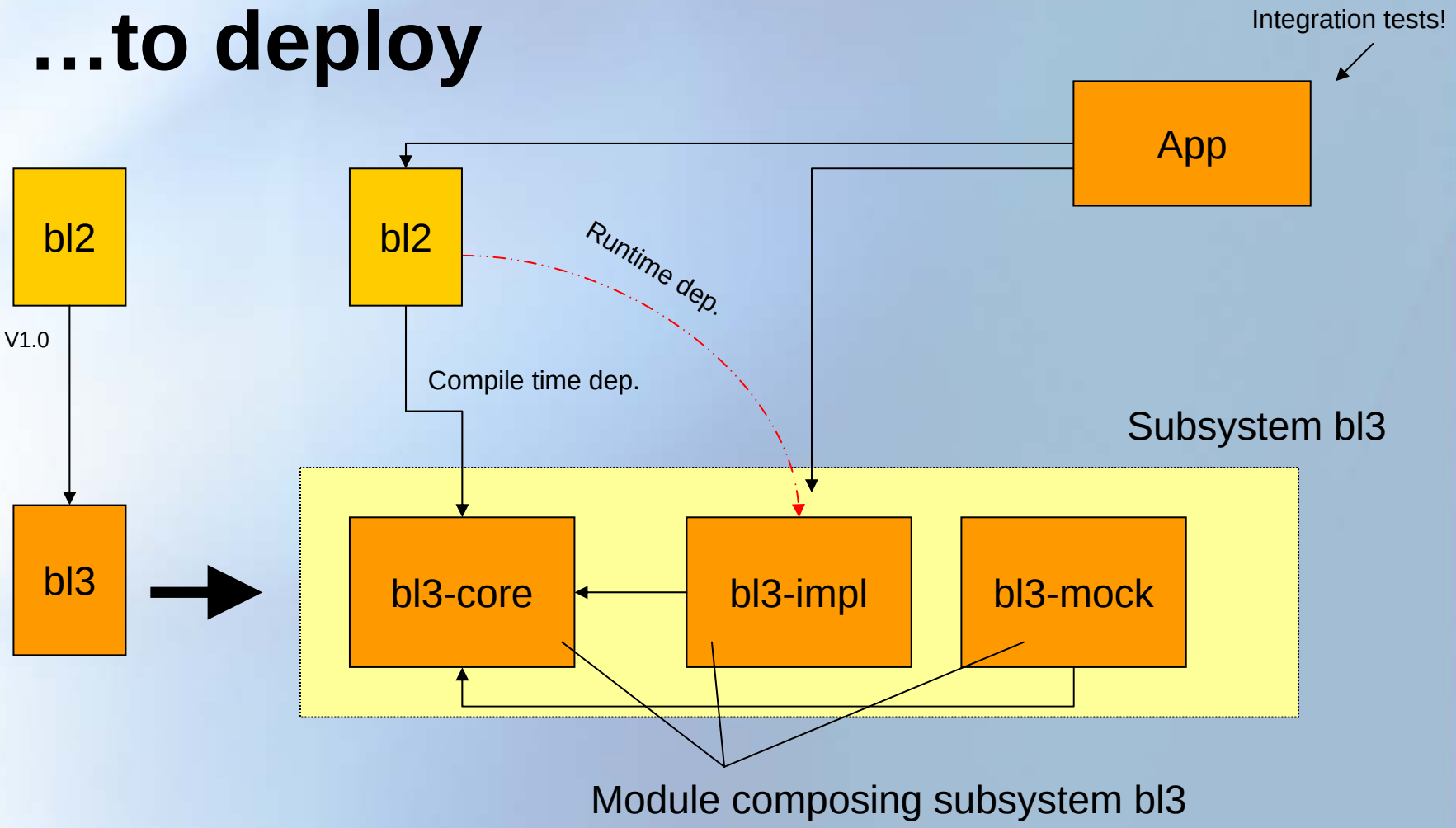
Contract first?



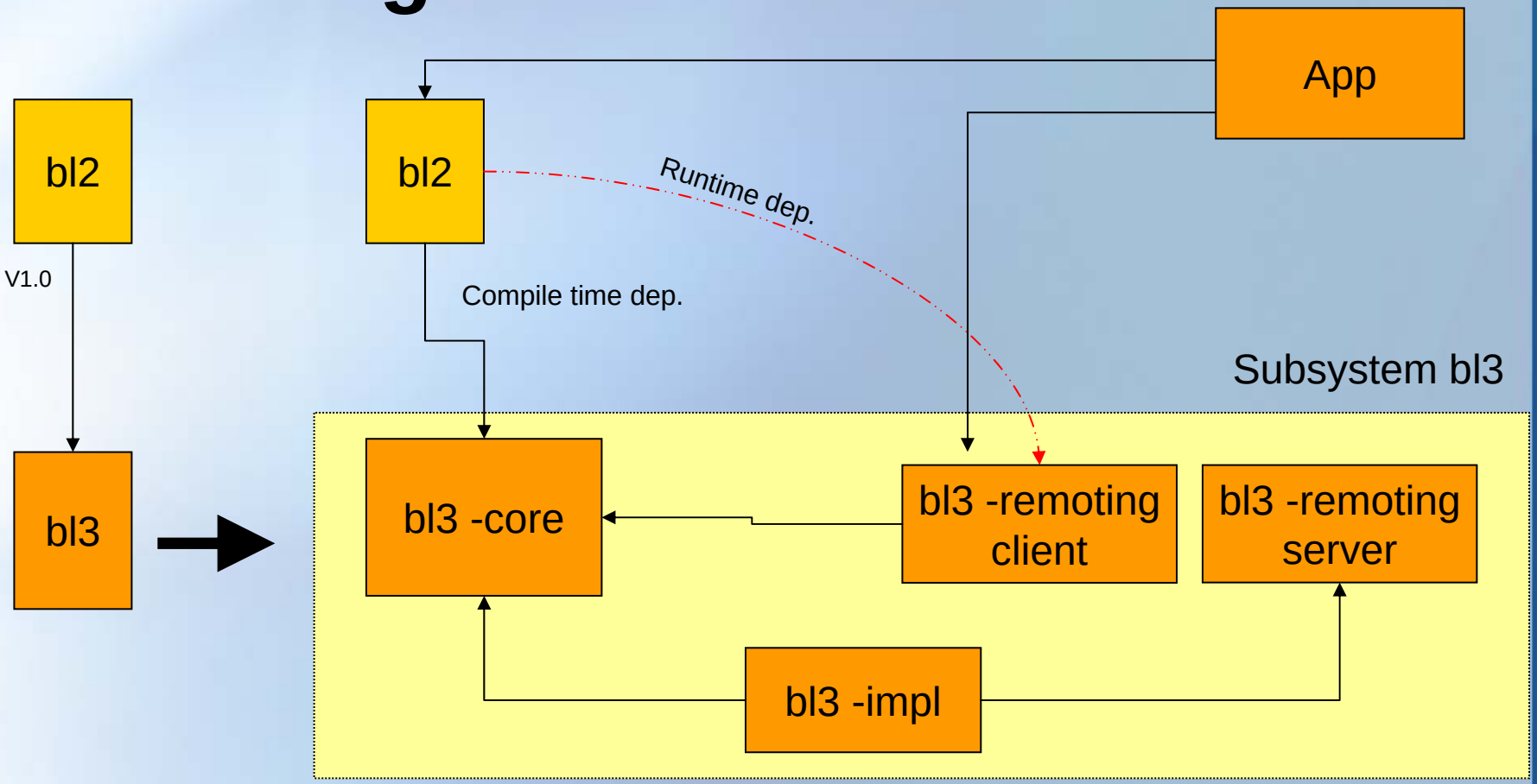
To test...


Integration tests???
Unit tests???
bl2
V1.0
bl3
bl2
Runtime dep.
Test scope
Compile time dep.
Subsystem bl3
bl3-core
bl3-local
bl3-mock
Module composing subsystem bl3
Module composing subsystem bl3

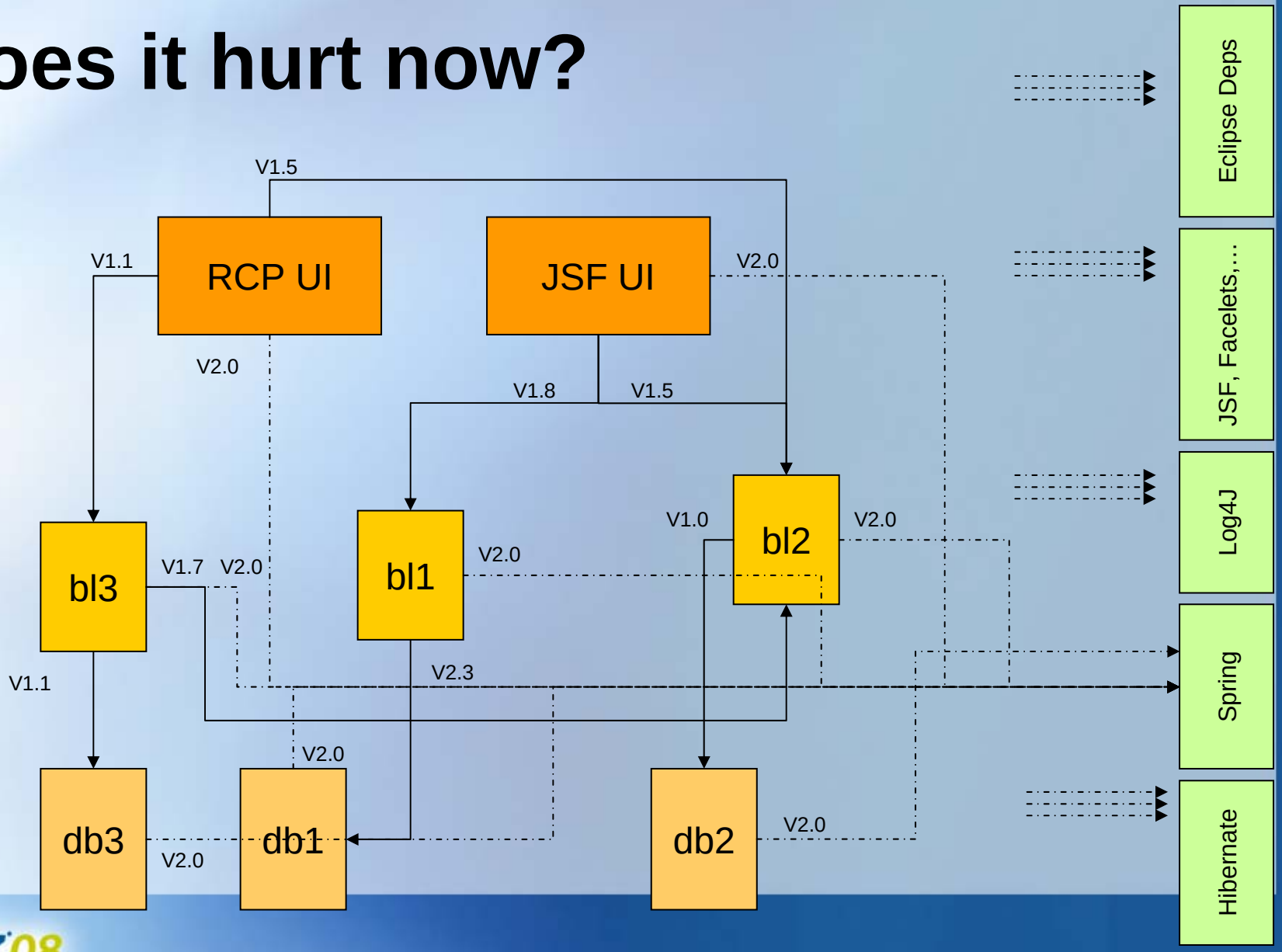
...to deploy


Integration tests!
App
bl2
V1.0
bl2
Runtime dep.
Compile time dep.
Subsystem bl3
bl3
bl3-core
bl3-impl
bl3-mock
Module composing subsystem bl3

Remoting?


App
bl2
bl2
Runtime dep.
V1.0
Compile time dep.
Subsystem bl3
bl3
bl3 -core
bl3 -remoting client
bl3 -remoting server
bl3 -impl

Does it hurt now?



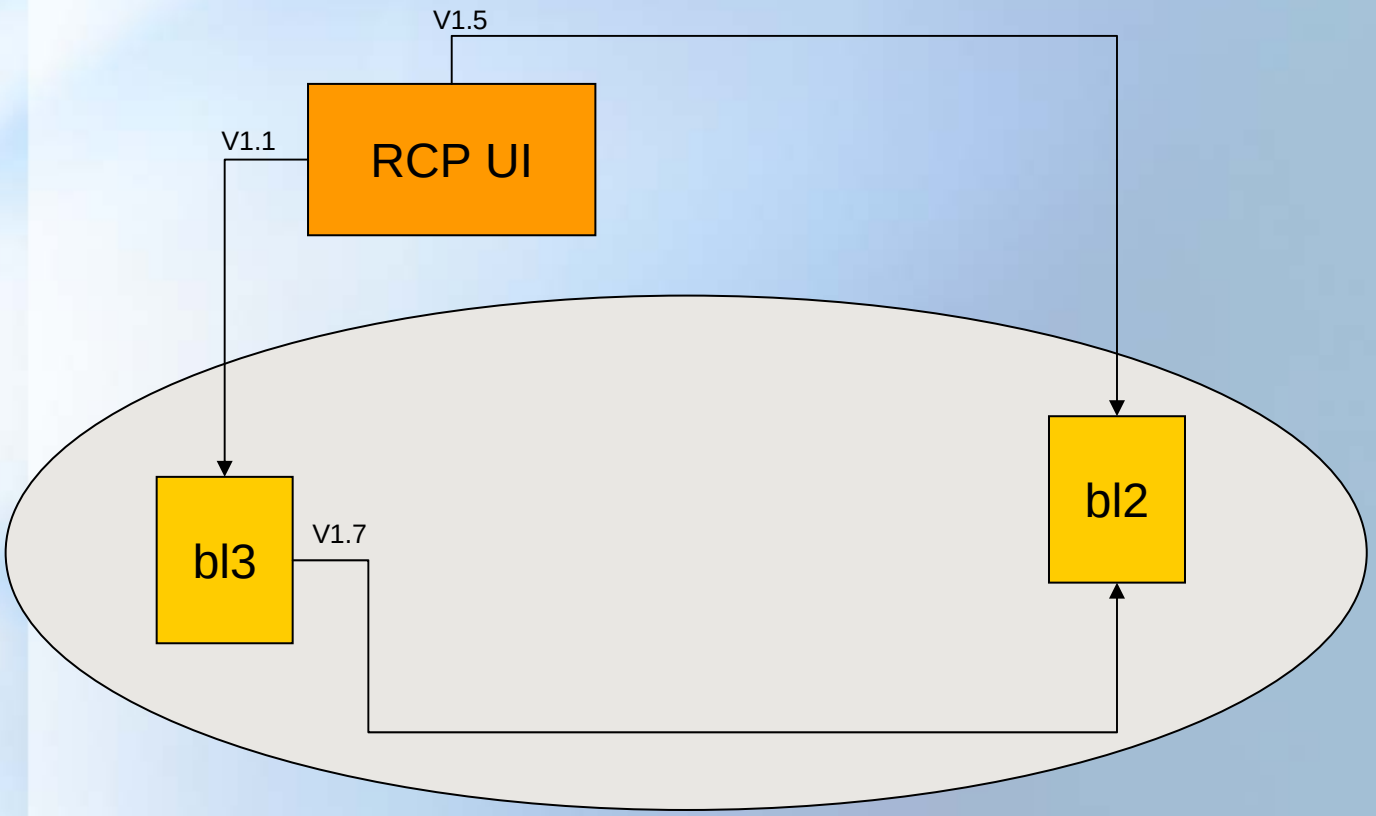
Kinder-Eggs

Excitement

Play

Fun

Transitive dependencies

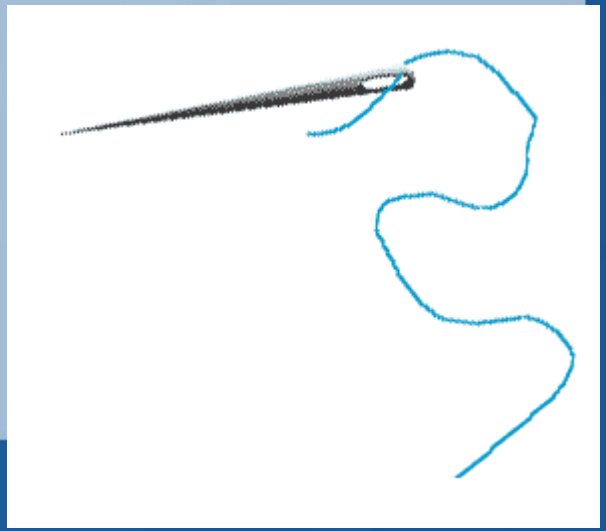


Version conficts needs to be solved by soft „**version ranges**“ and „**dependency mediation**“

Cyclic dependencies

Weave all aspects into a solution

- Compile time dependency management
  - OSS, large community: Maven?
- Runtime dependency management
  - OSS, rock solid: Spring?



Compile time dependencies managed by Maven?

*„Maven is a software **project management** and comprehension tool. Based on the concept of a project object model **(POM)**, Maven can manage a project's build, reporting and documentation **from a central piece of information**.“*

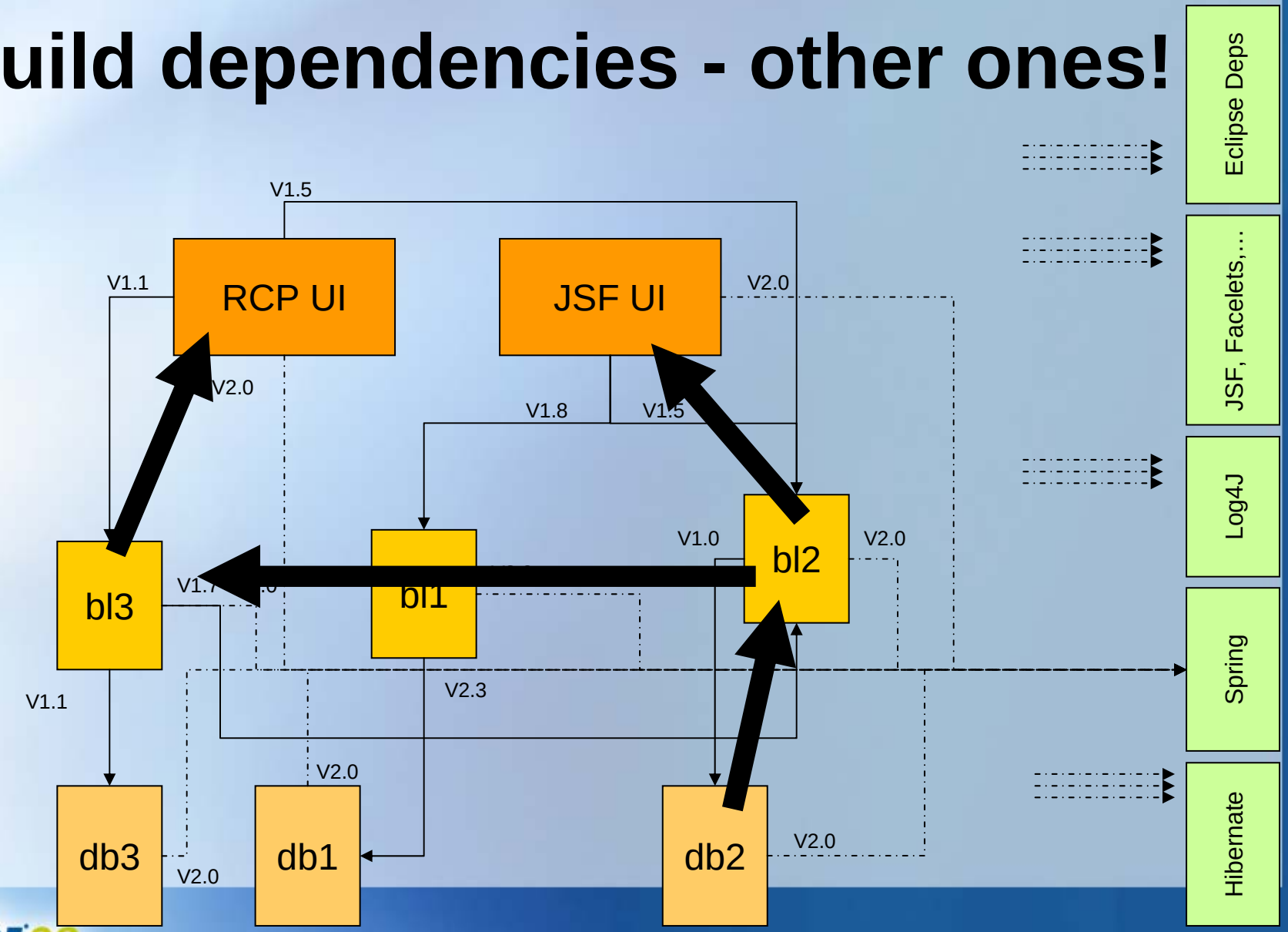
```
<project>
   (…)
   <dependencies>

        <dependency>
            <groupId>persistence-subsystems</groupId>
            <artifactId>db2-core</artifactId>
            <version>2.0</version>
            <scope>compile</scope>
        </dependency>

        <dependency>
            <groupId>persistence-subsystems</groupId>
            <artifactId>db2-mock</artifactId>
            <version>2.0</version>
            <scope>test</scope>
        </dependency>
   (…)
```

Build manager?

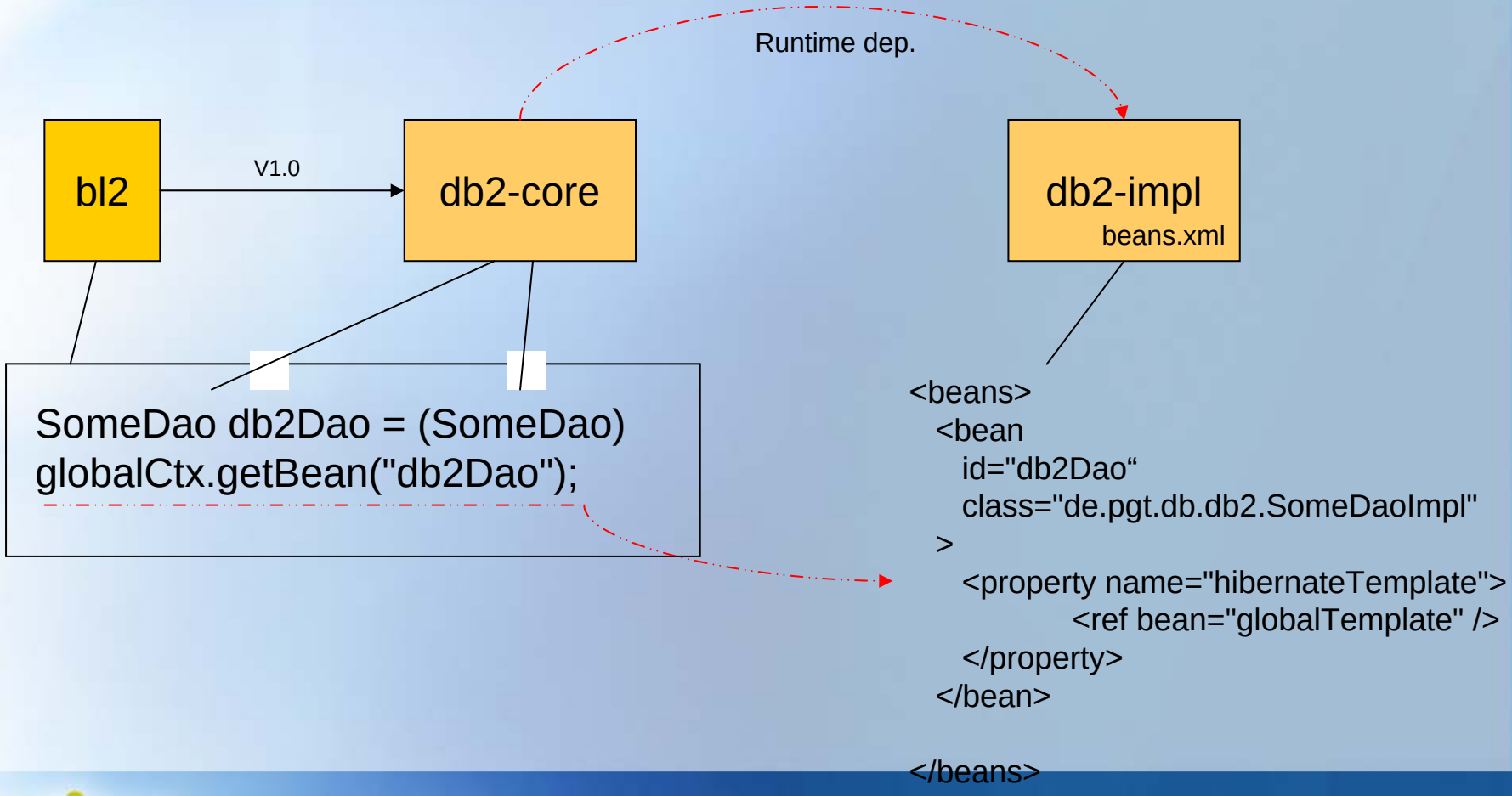
- Don‘t do it manually!
  - Cruise Control
  - Continuum
  - Hudson
  - **Bamboo**
  - Pulse
  - …

http://damagecontrol.codehaus.org/Continuous+Integration+Server+Feature+Matrix

Build dependencies - other ones!



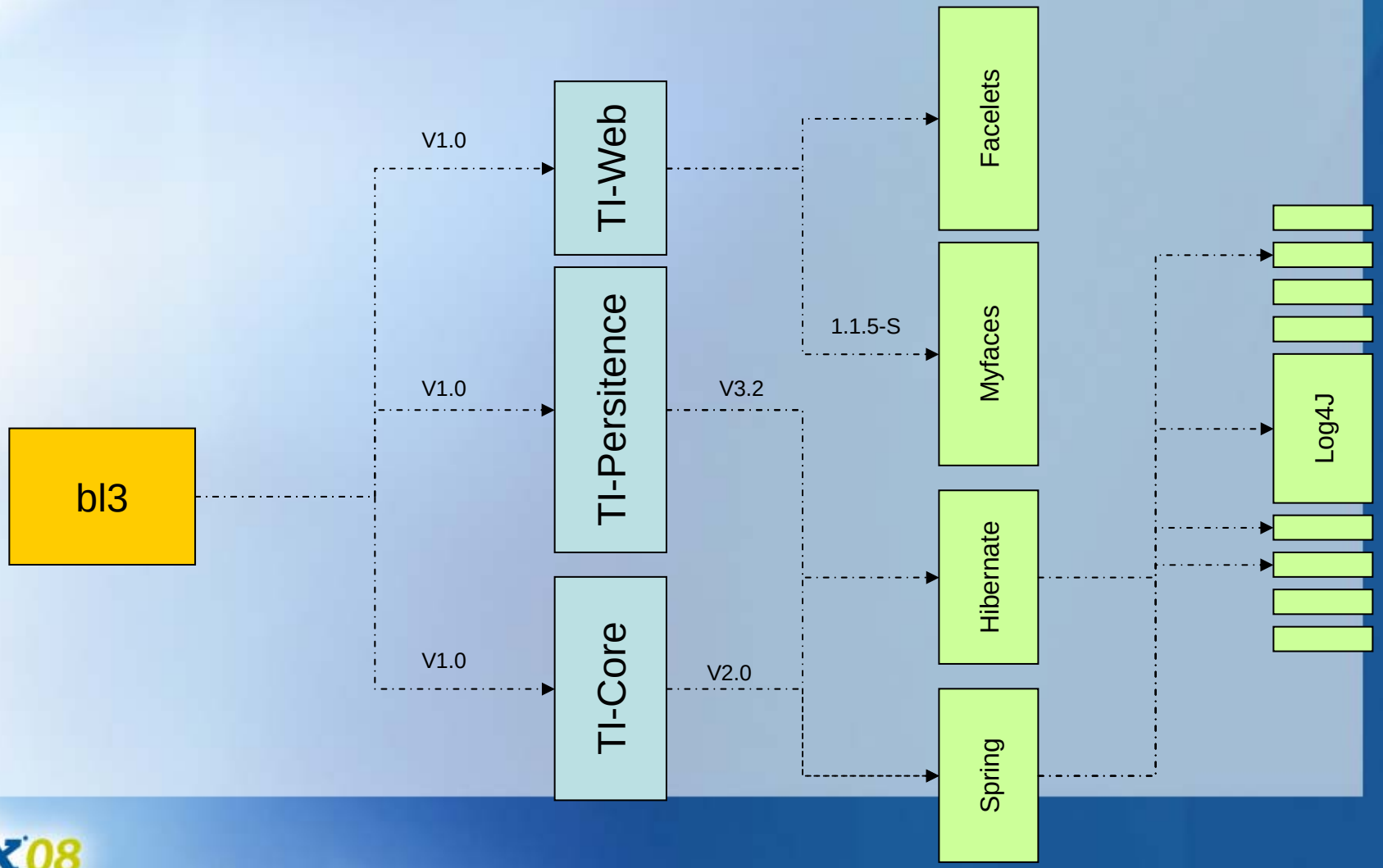
Runtime dependencies management with Spring?



Name patterns

- Classifying modules by name patterns
- e.g.:
  - TI : technical infrastructure module
  - BL: business logic module
  - AA: application assembly module

Hiding complexity



ti-* modules

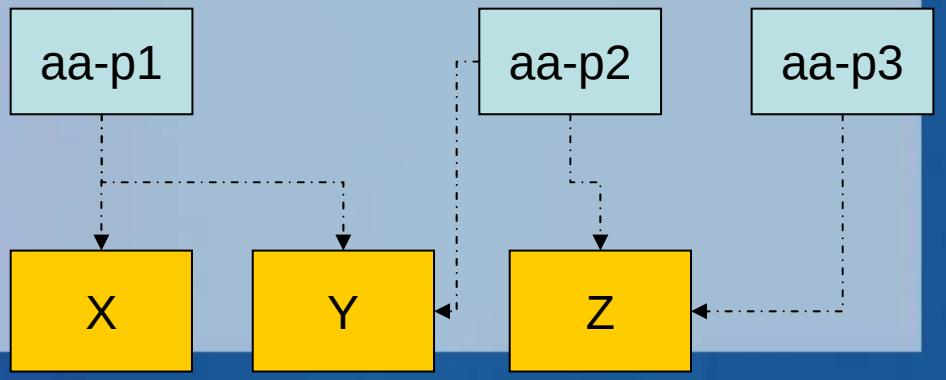
- ti-core
  - Spring Framework
  - Bootstrapping
- ti-hibernate
  - Hibernate
  - Generic DAO services (CRUD)
- ti-jsf
  - JSF, Facelets

bl-* modules

- bl-usermanagement
- bl-calendar
- bl-timetracking
- …

aa-* modules

- Aggregate modules
-
  - Coarse grained subsystems
  - Deployable artefacts



After all, looks good…

- ***"There must be a hitch somewhere"***

- Where is the hitch here?

Problem

Most technologies jeopardize any trial to create modularized architectures, no matter how simple or complex the modularization was designed....

Java will make modularization a first class citizen with the upcoming superpackages and dynamic modules.

OSGI solutions do have the same problems.

They all solve modularization, but they do not make technologies module aware.

Cool technologies

- Might be quite uncool when we try to create modules…
- We have many many component models, none of them introduce modularization concepts
- Health-Drinks can become Poison-Drinks

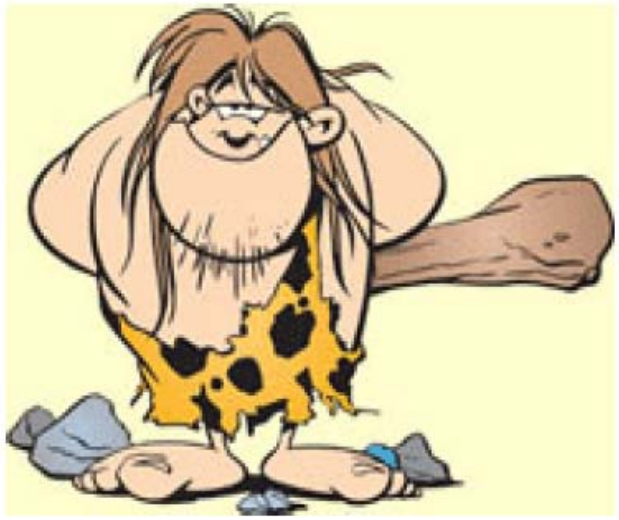
What next?

First think

Then take action!

Don‘t solve problems with brute force

- Don‘t try to solve everything with build tools



Antidote: http://antidoto.sf.net

- **Antidoto** is a java technology mashup enabling modularization for Java enterprise technologies such as Spring and Hibernate. Modern java technologies are component based, but lack modularization capabilities. Here is where the poisoning is happening and where **Antidoto** jumps in. To enable modularization, **Antidoto** defines a simple module metamodel, where a system represents the whole and subsystems are the composing parts. As such **Antidoto** defines a coarse grained component model. We prefer subsystem over component as the latter term is being used by many frameworks to describe fine grained components. By providing building blocks for modular architectures, **Antidoto** forms a thin modularization layer ontop of well known Java technologies.

Example from Antidoto: logging

- Log4J
  - Single configuration file
    - Different configuration domains
      - Configure appenders and layout
      - Configure categories and levels
    - Difficult if application server already uses Log4J (e.g. jBoss)

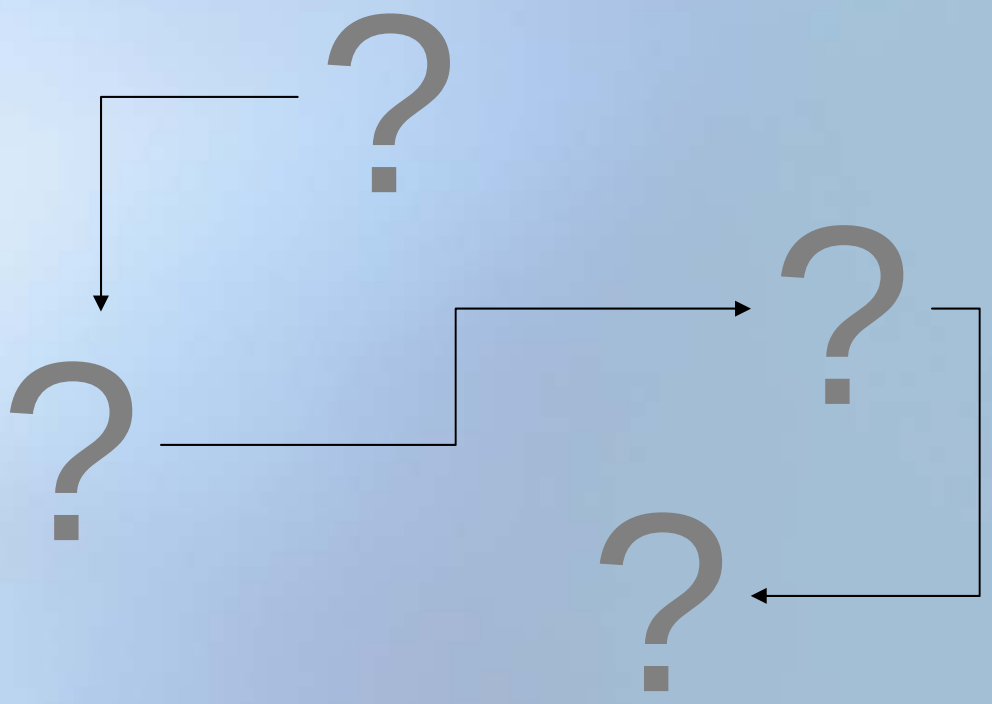
Solutions

- JBoss (or any other runtime already using log4j)
  - Add config elements for each application to JBoss config
  - Don‘t let the apps configure log4j (last wins)
- Merge fragments during buildtime
- Use the Log4J API at runtime

Using Spring custom namespaces

```
<?xml version="1.0" encoding="utf-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tiCore="http://antidoto.sf.net/subsystem/ti-core"
    xsi:schemaLocation=" http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
    http://antidoto.sf.net/subsystem/ti-core http://antidoto.sf.net/xsd/ti-core.xsd
    ">
    <tiCore:logger
        category="org.springframework" loglevel="DEBUG" />
    <tiCore:logger
        category="de.pgt.ti.core" loglevel="DEBUG" />
    <tiCore:logger
        category="some.test.category" loglevel="ERROR" />
</beans>
```

Time for questions?



Thank you!

- Papick G. Taboada
  - Technology Scout
  - Software Architect

  - Presentation material and more here:
    http://adminsight.de