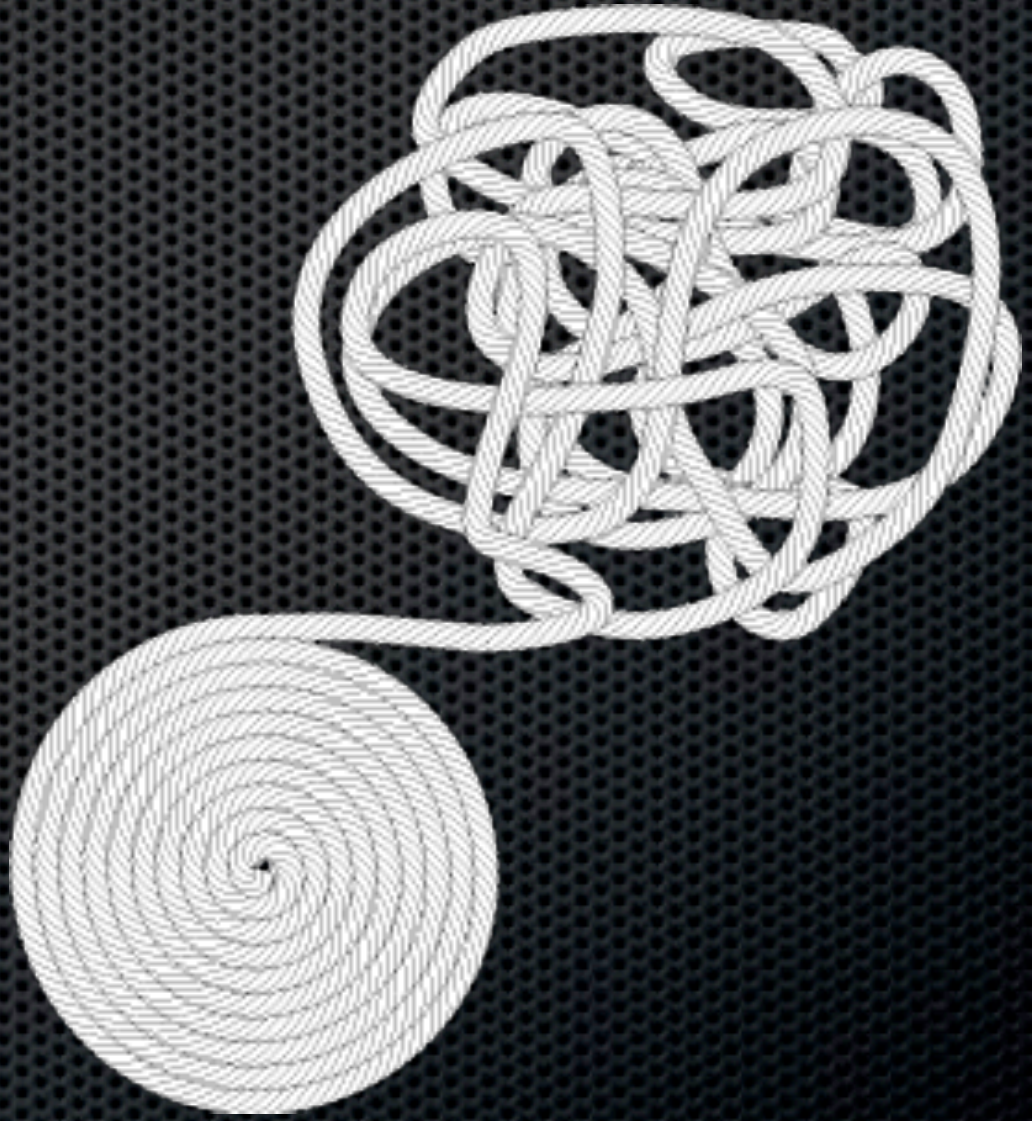
Where's my Architecture?

Chris Chedgey

Structure101



Discovering/defining architecture

- Real architecture
- Existing codebase structure
- Well-structured containment
- Creating well-structured containment
- Levelization
- Making it real

+ *Examples*

About Structure101 Inc.

Structure101

- Since 2000/2007
- Team in Ireland, France, India, Spain, Canada, …
- Web+channel sales

"Structure101 shaved months of calendar time and man years of effort off the project"
Bill Jackson, Netflix

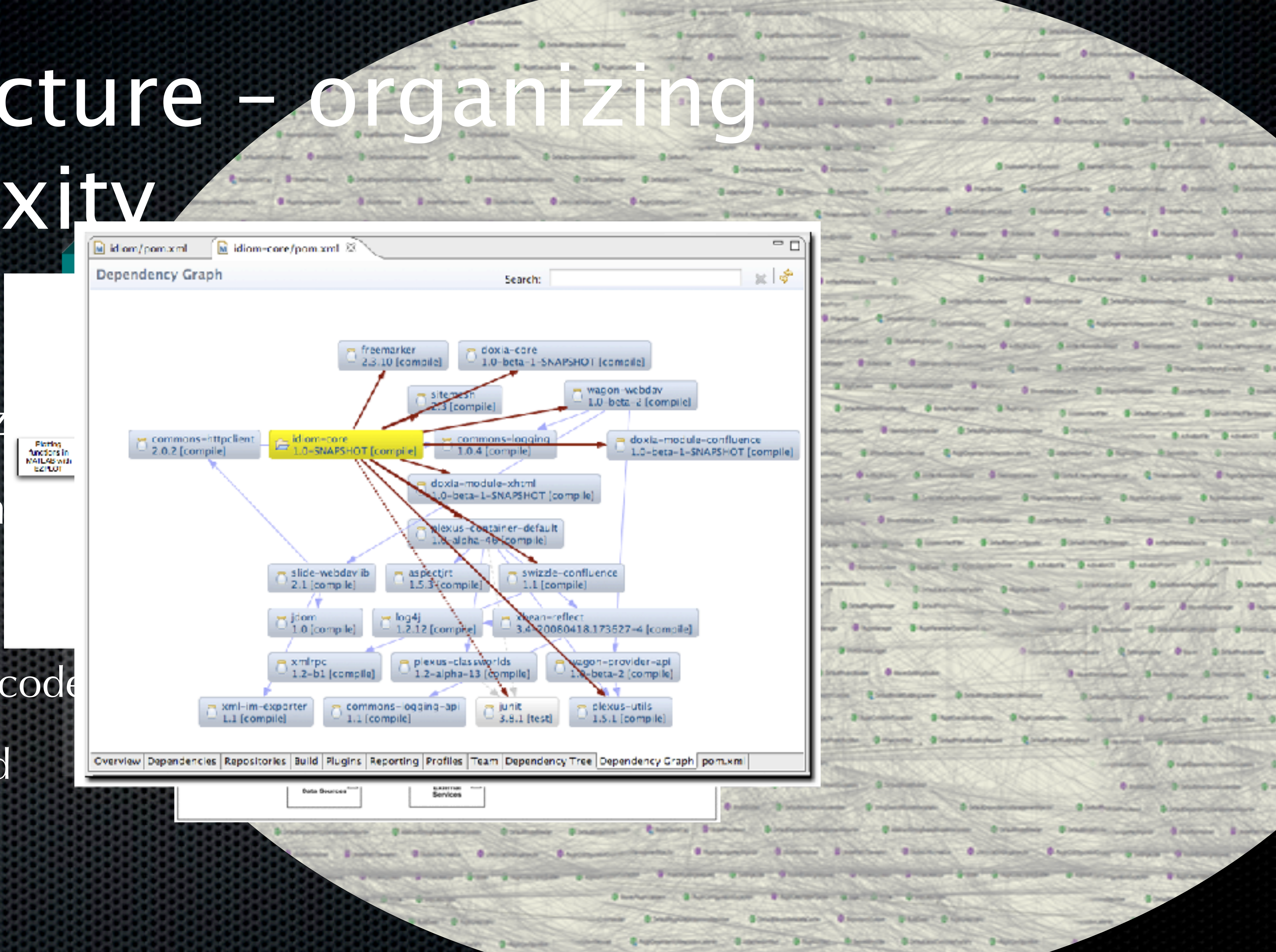
Why Structure?

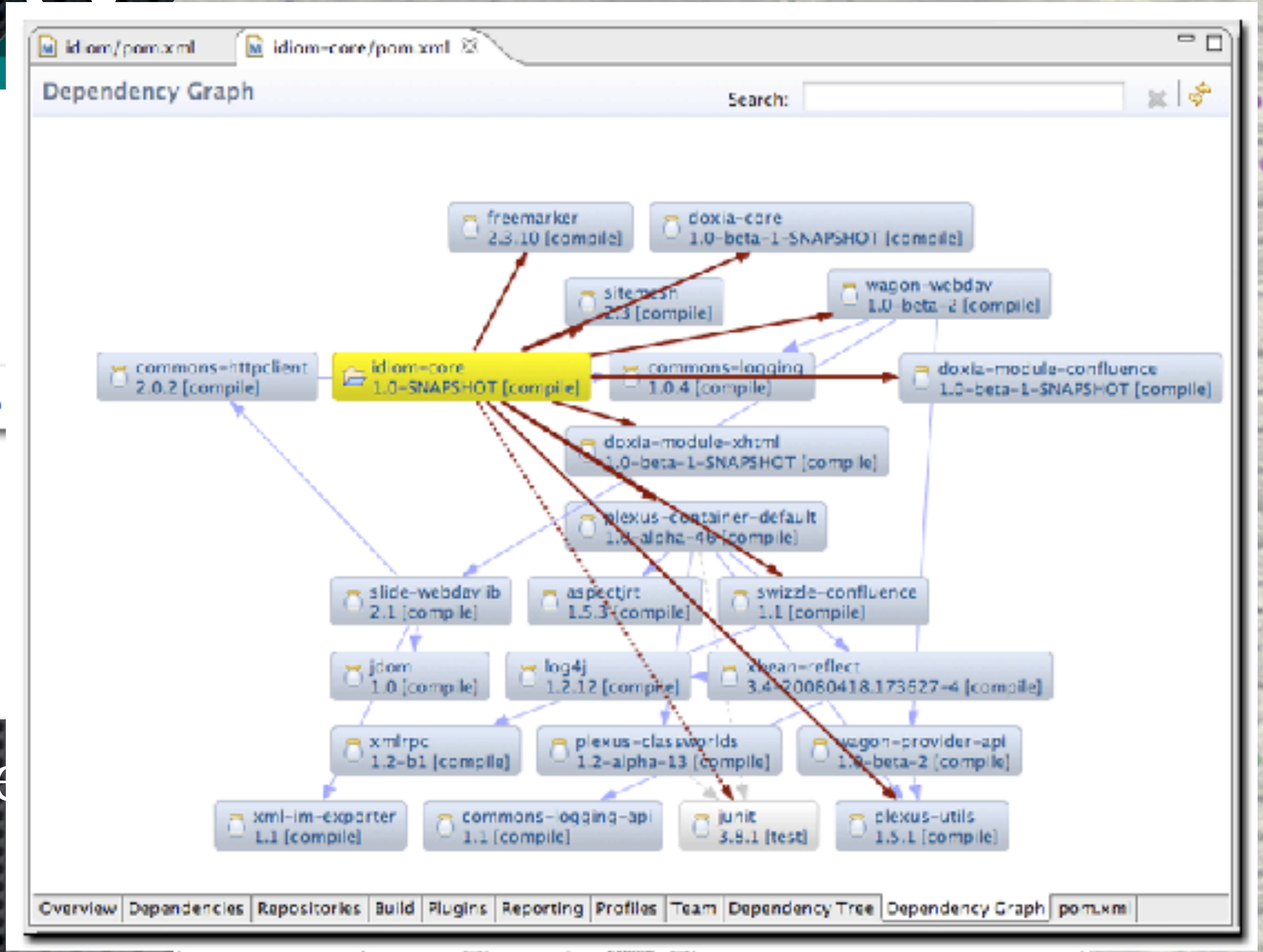
"Well-structured software is delivered in half the time, at half the cost, with 8x less bugs"
US Air Force study.

- When a codebase grows beyond a certain size, without a guiding architecture, developers start drowning in an expanding sea of source files
- This is a huge, pervasive driver of cost which impacts all development activities
- ***Discovering defining an architecture for an existing code base is a much lower cost and risk than struggling on… or starting over***
- But this has required a new kind of tool → ***Structure101***

Architecture – organizing complexity

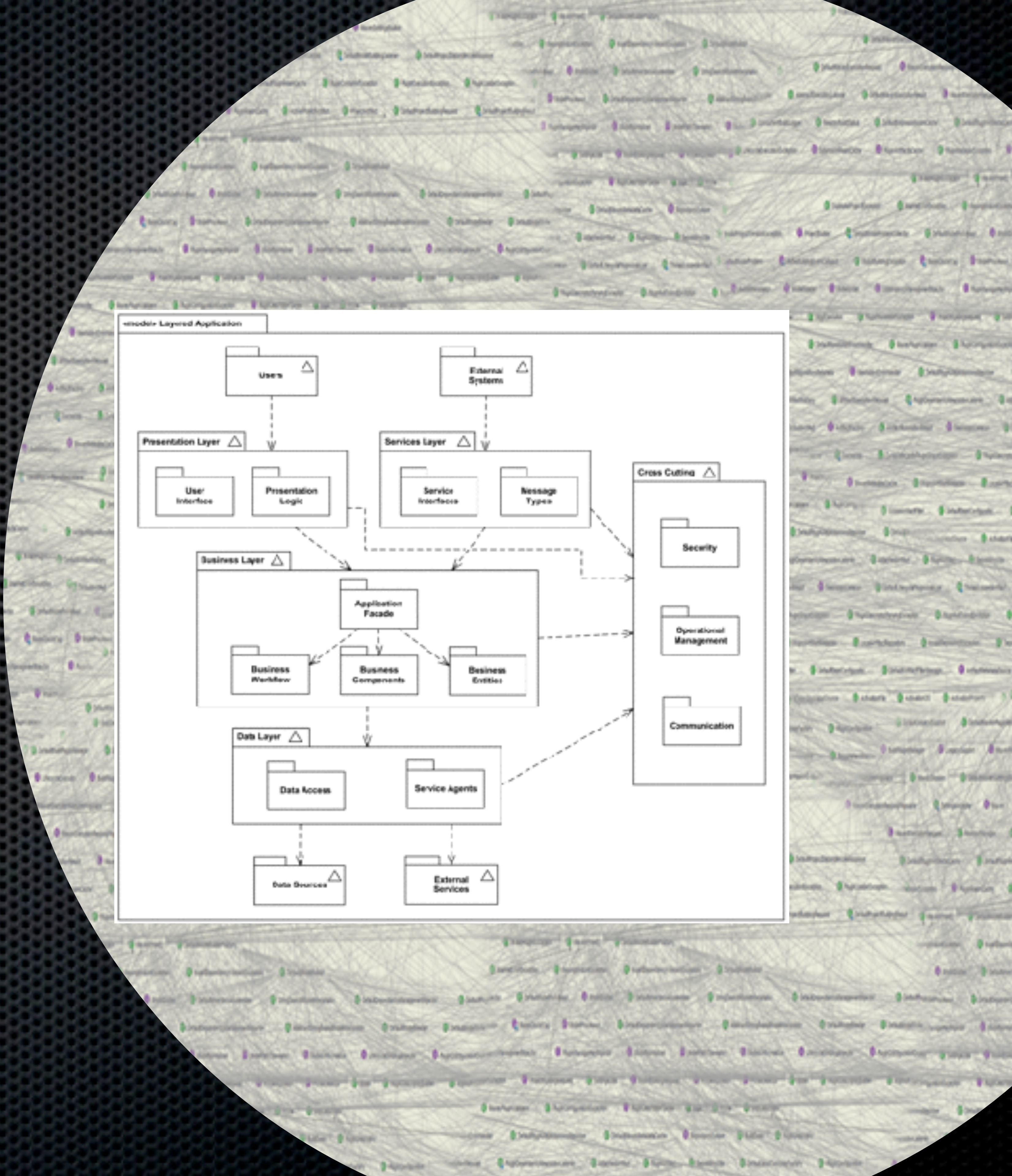
- Well organiz
- Inter-contain
- Real?
    - Maps to code
    - Validated




idiom/pom.xml
idiom-core/pom.xml
Dependency Graph
Search:
freemarker 2.3.10 [compile]
doxia-core 1.0-beta-1-SNAPSHOT [compile]
sitemesh 2.3 [compile]
wagon-webdav 1.0-beta-2 [compile]
commons-httpclient 2.0.2 [compile]
idiom-core 1.0-SNAPSHOT [compile]
commons-logging 1.0.4 [compile]
doxia-module-confluence 1.0-beta-1-SNAPSHOT [compile]
doxia-module-xhtml 1.0-beta-1-SNAPSHOT [compile]
plexus-container-default 1.0-alpha-46 [compile]
slide-webdavlib 2.1 [compile]
aspectjrt 1.5.3 [compile]
swizzle-confluence 1.1 [compile]
jdom 1.0 [compile]
log4j 1.2.12 [compile]
xbean-reflect 3.4-20080418.173527-4 [compile]
xmlrpc 1.2-b1 [compile]
plexus-classworlds 1.2-alpha-13 [compile]
wagon-provider-api 1.0-beta-2 [compile]
xml-im-exporter 1.1 [compile]
commons-logging-api 1.1 [compile]
junit 3.8.1 [test]
plexus-utils 1.5.1 [compile]
Overview
Dependencies
Repositories
Build
Plugins
Reporting
Profiles
Team
Dependency Tree
Dependency Graph
pom.xml

Real Architecture

- Map/blueprint for developers
- Phased testing and release
- Divide work across organizations, teams, individuals
- Modularity: interfaces + info hiding
- Reuse or replace subsystems or layers
- Impact/regression control
- Help new developers
- …
- *Agile Engineering*



Controlling Architecture

- New project
  - Define architecture that maps to the evolving codebase
  - Communicate, enforce, evolve
- Existing codebase
  - ***Discover***/define architecture that maps to the evolving codebase
  - Communicate, enforce, evolve

The structure of a codebase

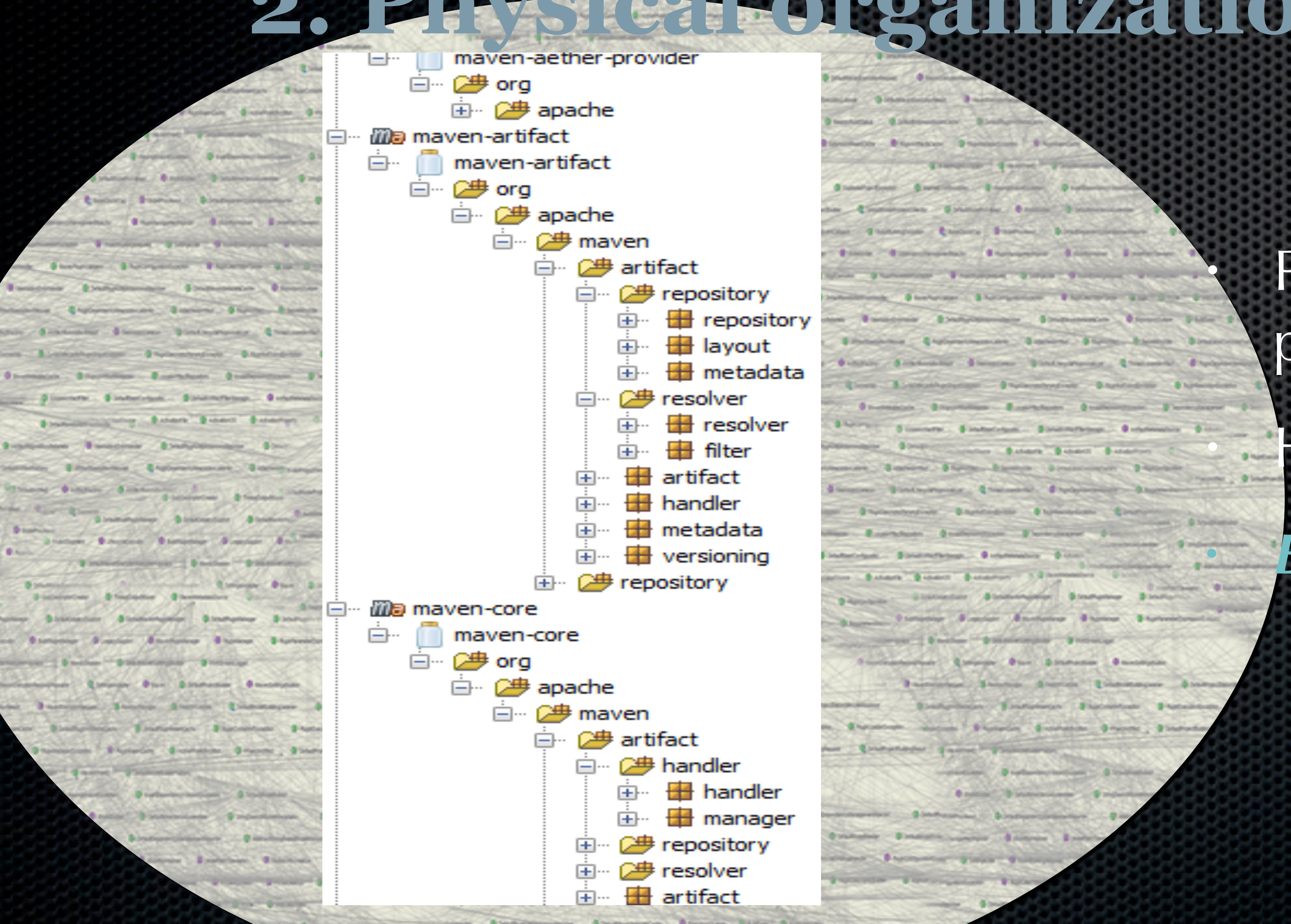
What we have (raw material)

1. Implementation

- Thousands of source files
- Countless interdependencies
- *Not an "architecture"*

What we have

2. Physical organization

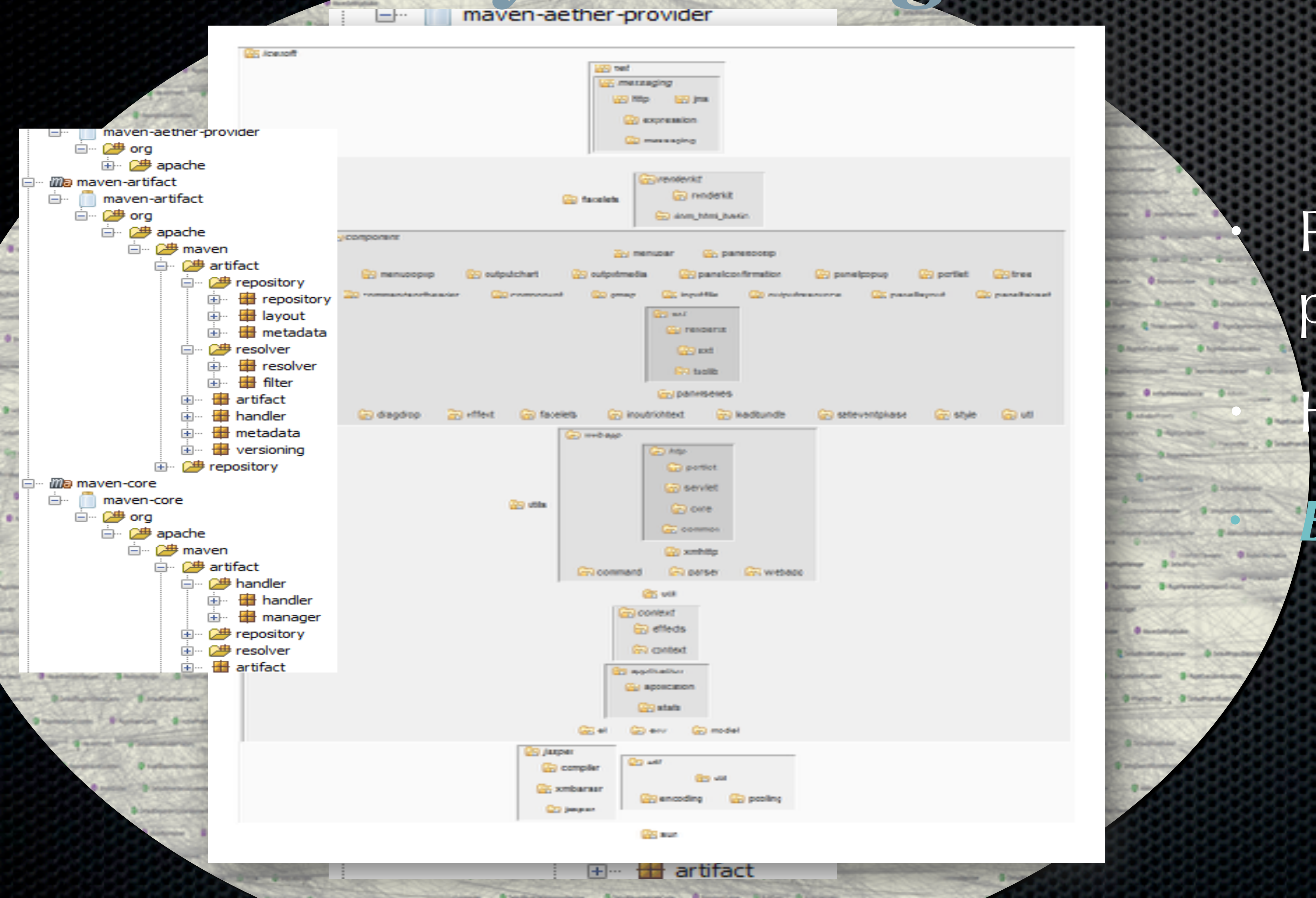


```
      maven-aether-provider
         org
            apache
maven-artifact
      maven-artifact
         org
            apache
               maven
                  artifact
                     repository
                        repository
                        layout
                        metadata
                     resolver
                        resolver
                        filter
                     artifact
                     handler
                     metadata
                     versioning
                  repository
maven-core
      maven-core
         org
            apache
               maven
                  artifact
                     handler
                        handler
                        manager
                     repository
                     resolver
                     artifact
```

- Packages, jars, Maven projects, …
- Helps to find files
- *But is it an "architecture"?*

What we have

2. Physical organization

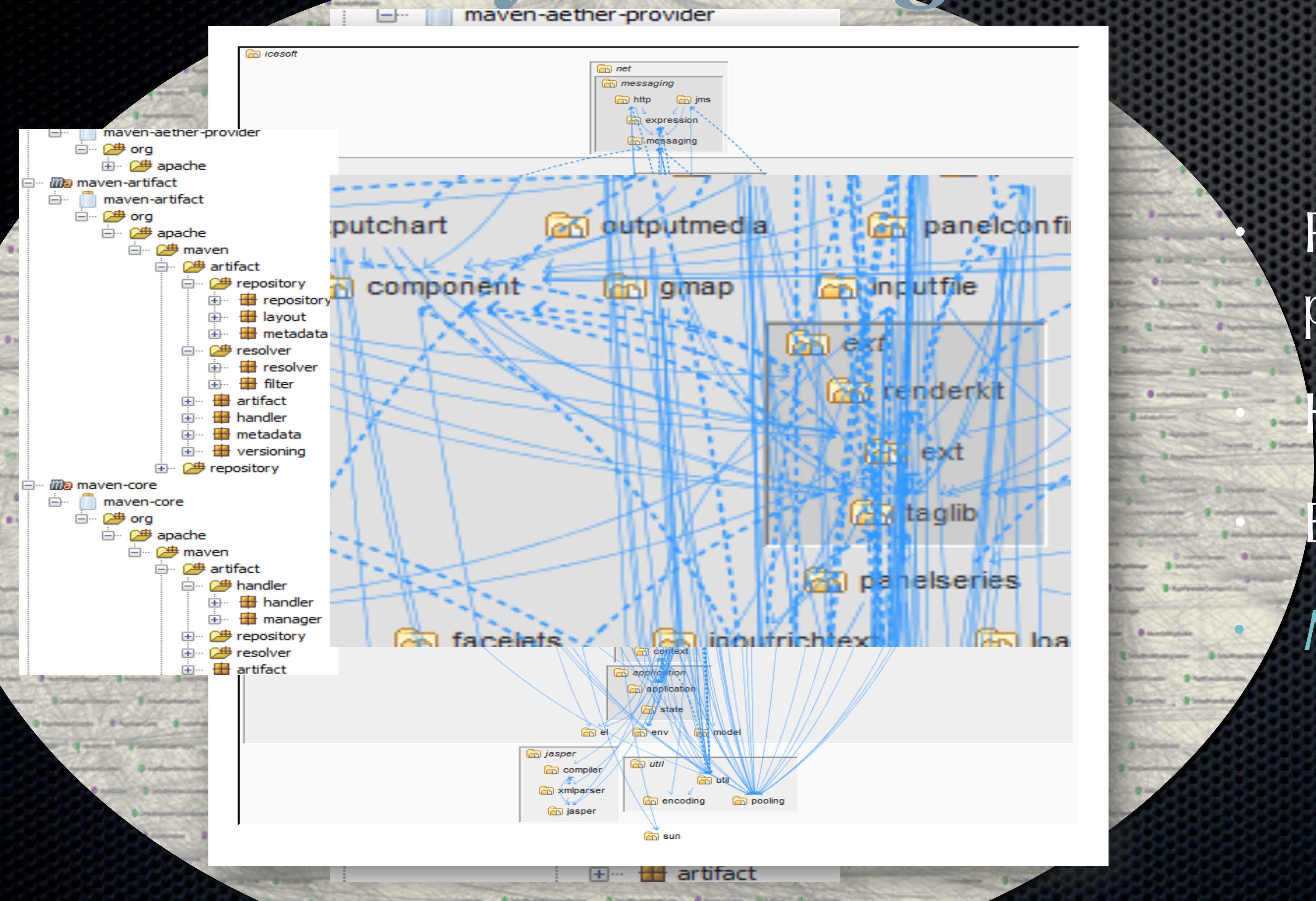


```
maven-aether-provider
  org
    apache
maven-artifact
  maven-artifact
    org
      apache
        maven
          artifact
            repository
              repository
              layout
              metadata
            resolver
              resolver
              filter
            artifact
            handler
            metadata
            versioning
          repository
maven-core
  maven-core
    org
      apache
        maven
          artifact
            handler
              handler
              manager
            repository
            resolver
            artifact
```

- Packages, jars, Maven projects, …
- Helps to find files
- *But is it an "architecture"?*

What we have

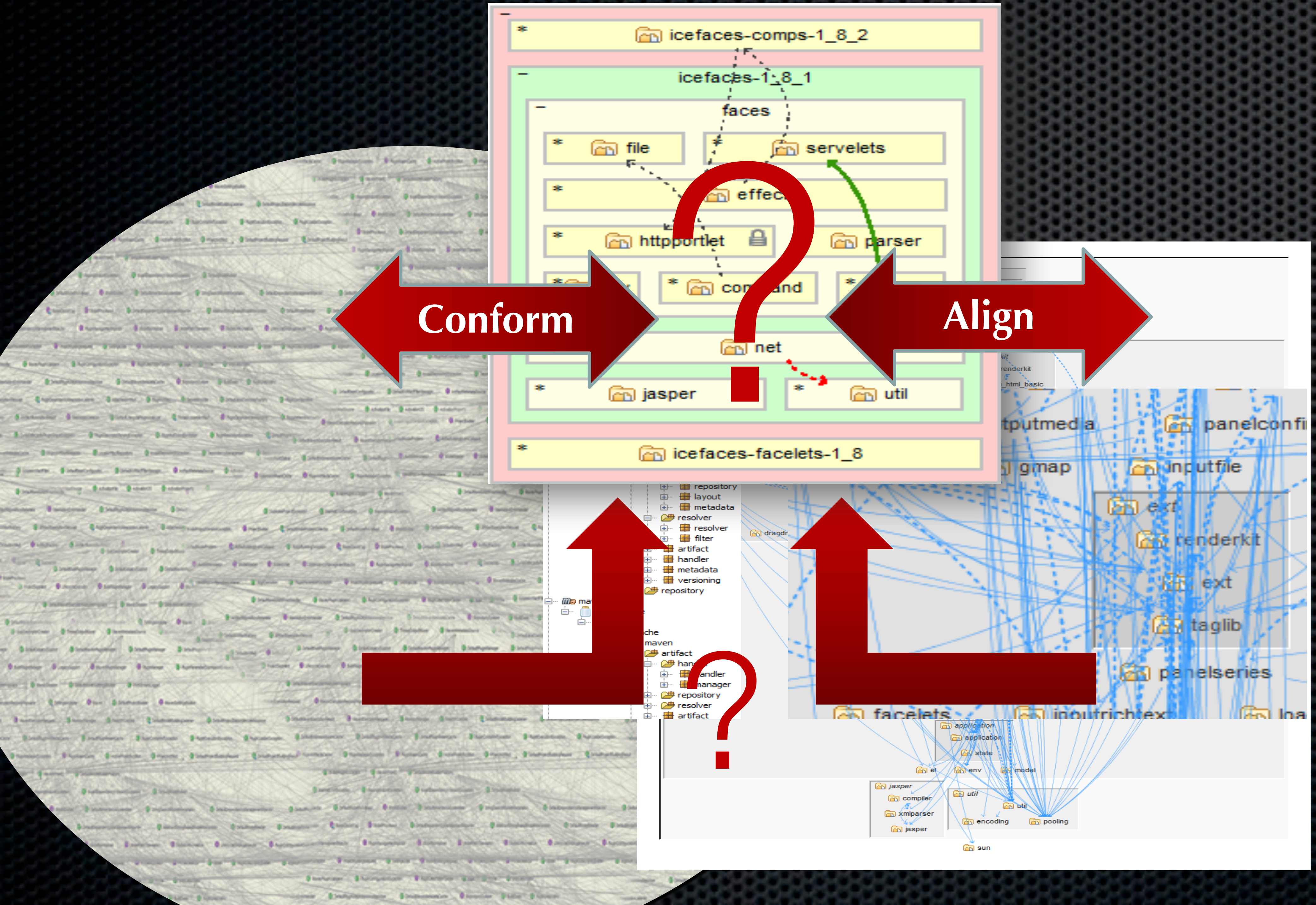
2. Physical organization



- Packages, jars, Maven projects, …
- Helps to find files
- But is it an "architecture"?
- *Not usually an "architecture"*

What do we need?

3. "Architecture"



- What is it?
- How do we get it?
- How do we make it real?

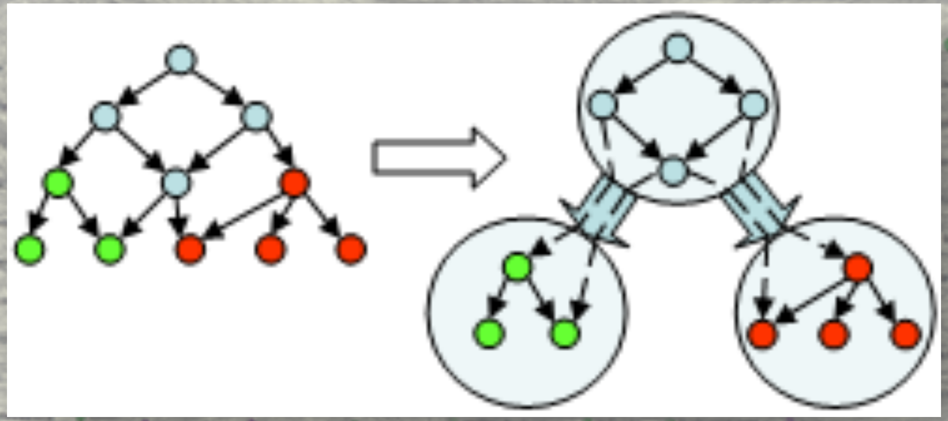
“Well-structured containment”

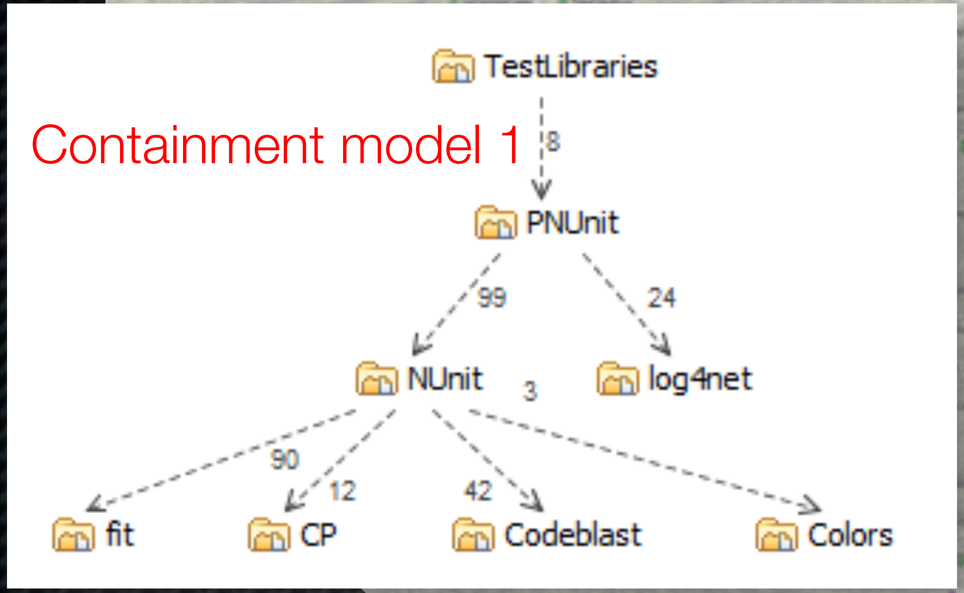
Containment

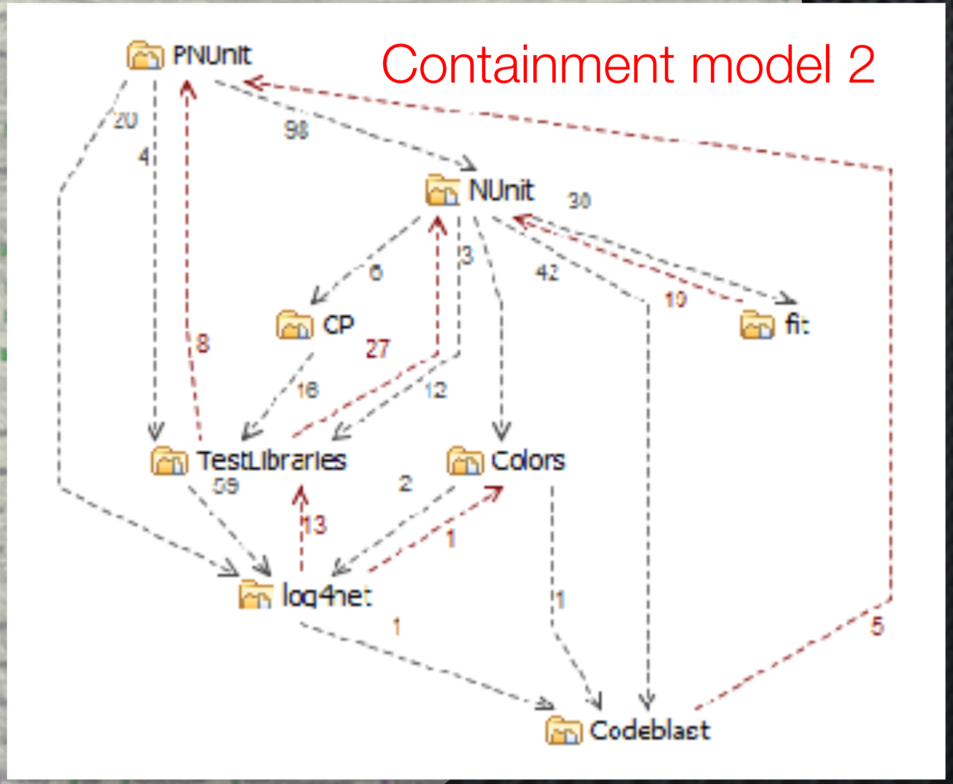
- Divide and conquer
- Code → method → class → package → subsystem → …
- “***Fat***” = too much in one place
- Grow and divide

Containment creates dependency

- Different containment
- Very different dependency
- *Containment is key to controlling dependency*

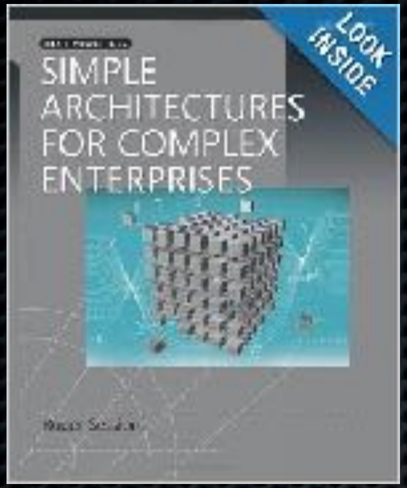


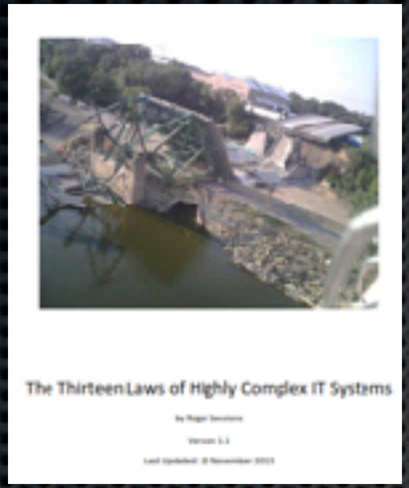


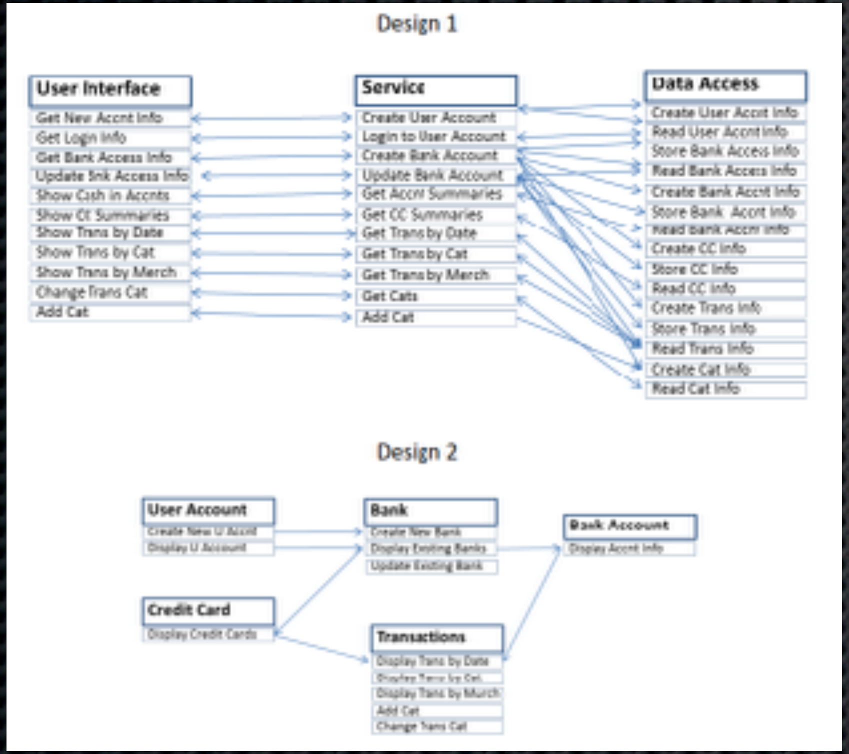


Dependency creates complexity

Roger Sessions:

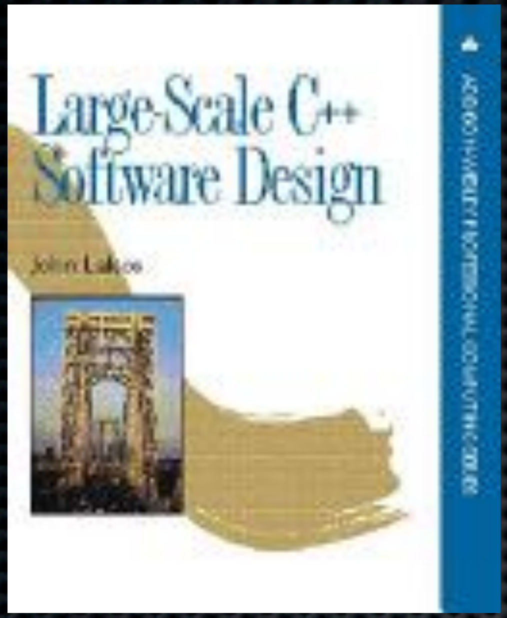


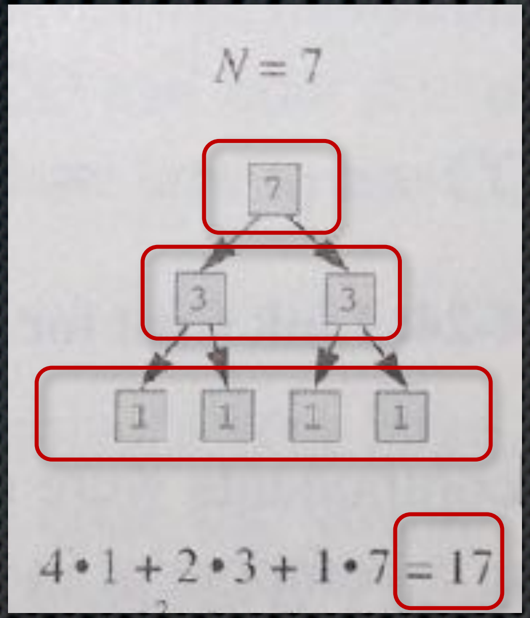




- ***Law 3***. Complexity is driven by interdependencies.
- ***Law 10***. Complexity is an undesirable architectural attribute of an IT system.
  - ***Reliability***: Most IT failures are due to complexity.
  - ***Auditability***: complex systems are extremely difficult to audit for regulatory compliance.
  - ***Security***: complexity increases the chances of fraud and vandalism.
  - ***Alignment***: complexity results in poor alignment between IT systems and business needs.
  - ***Cloud***: complexity results in inefficient use of cloud resources.
  - ***Maintainability***: complexity makes system maintenance much more difficult.
  - ***Agility***: complexity makes change much more difficult.
  - ***Scalability***: complex systems are hard to scale up when user demand exceeds expectations..

Dependency is cumulative





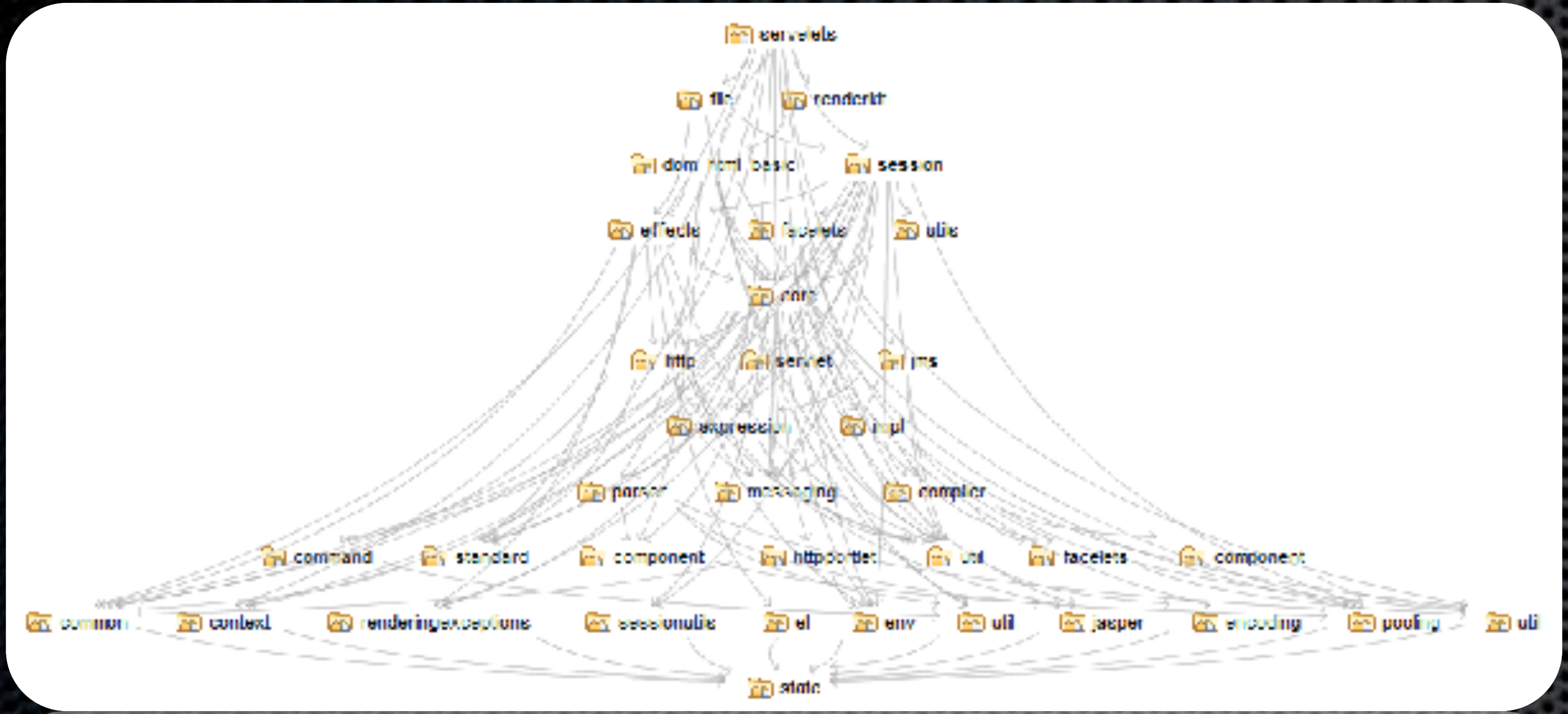
Cumulative Component Dependency (CCD)
- John Lakos

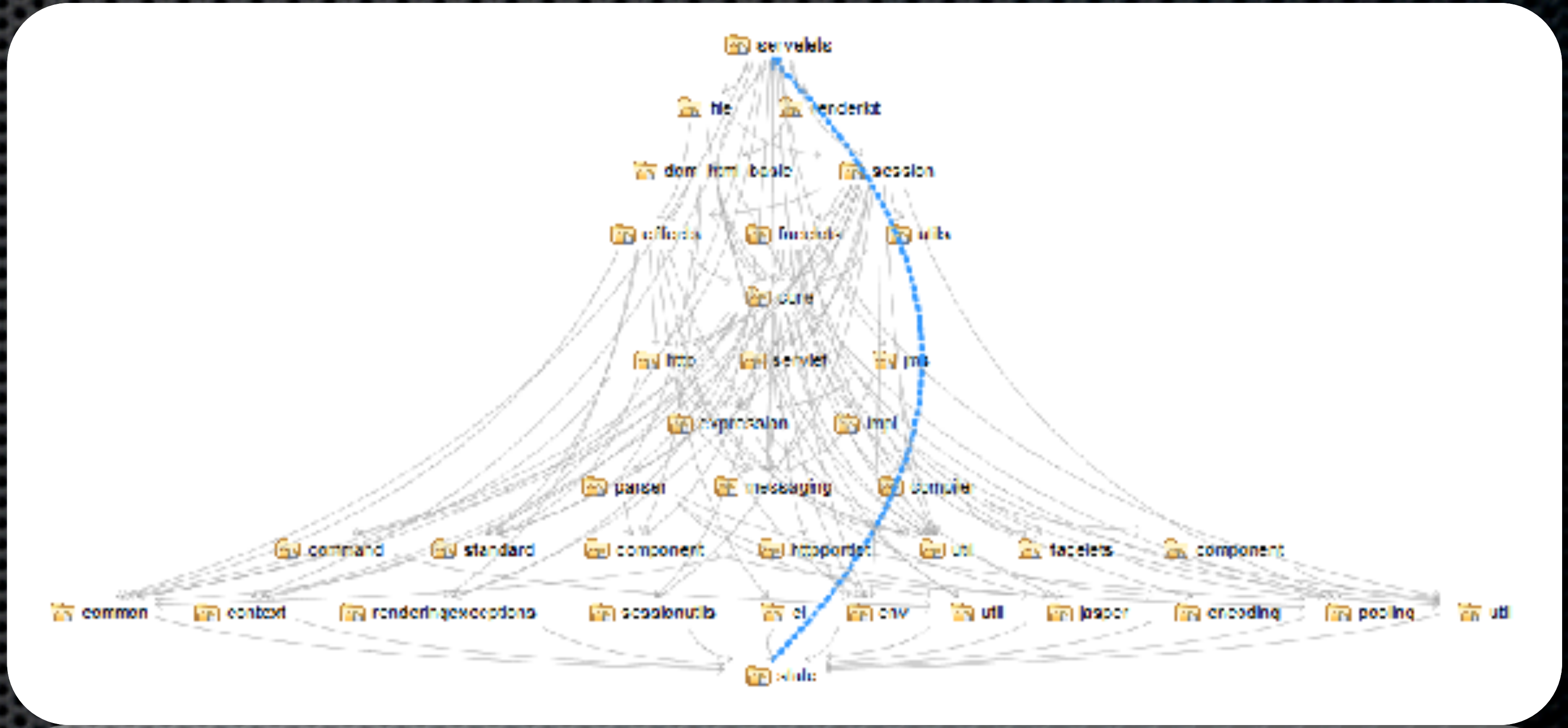
"In software architecture, resource constraint is not the big expense right now... it's coupling"
-Neal Ford, Thoughtworks

"Law 8. Complexity increases exponentially"
-Roger Sessions

Cycles explode dependency

"Cyclic dependencies have the greatest capacity to increase the overall cost of developing and maintaining a system"
- John Lakos

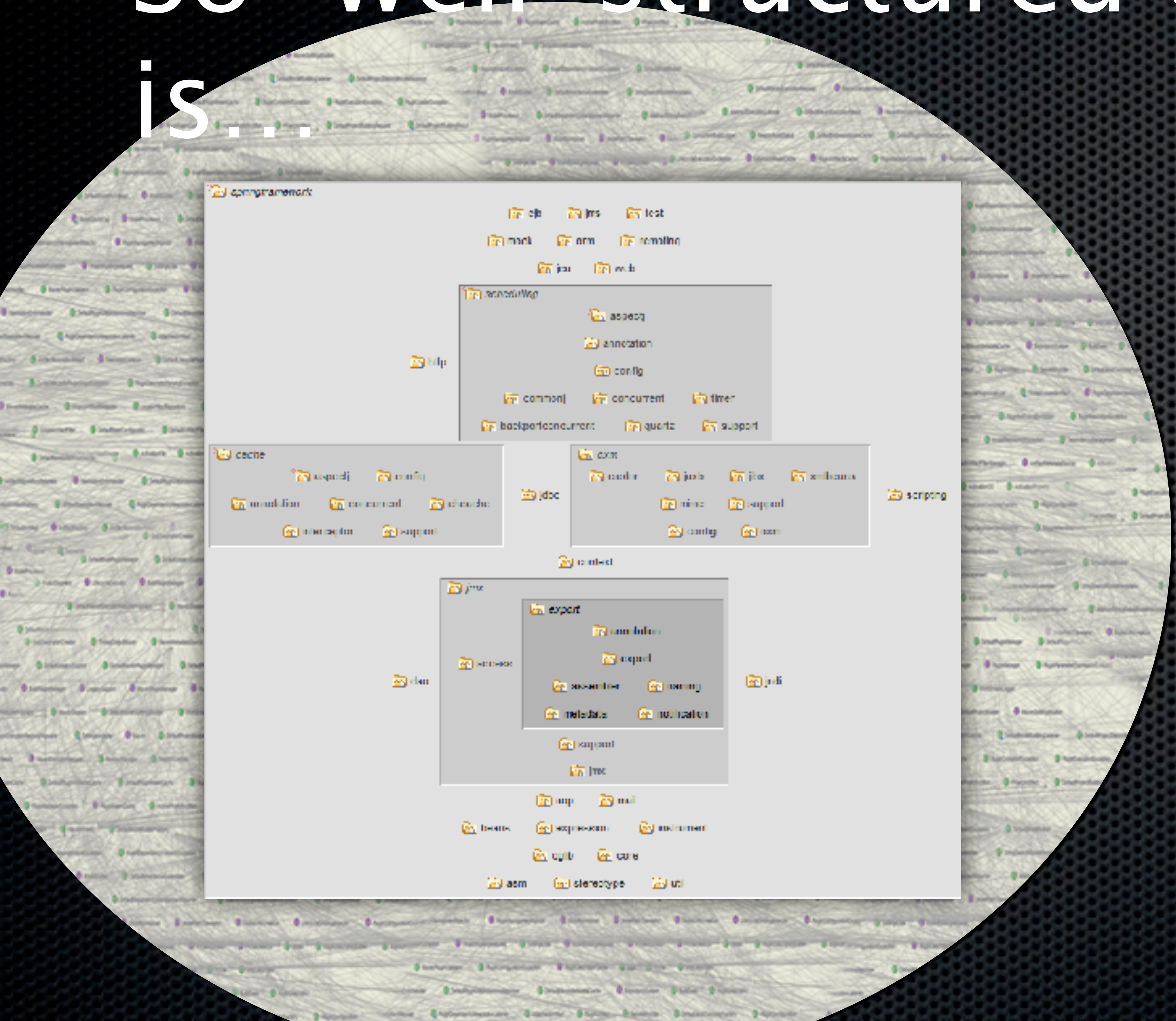




- CCD = (1*1) + (11*2) + (7*3) + …
  - < 164

- CCD = 36^2
  - = 1,296 !!!

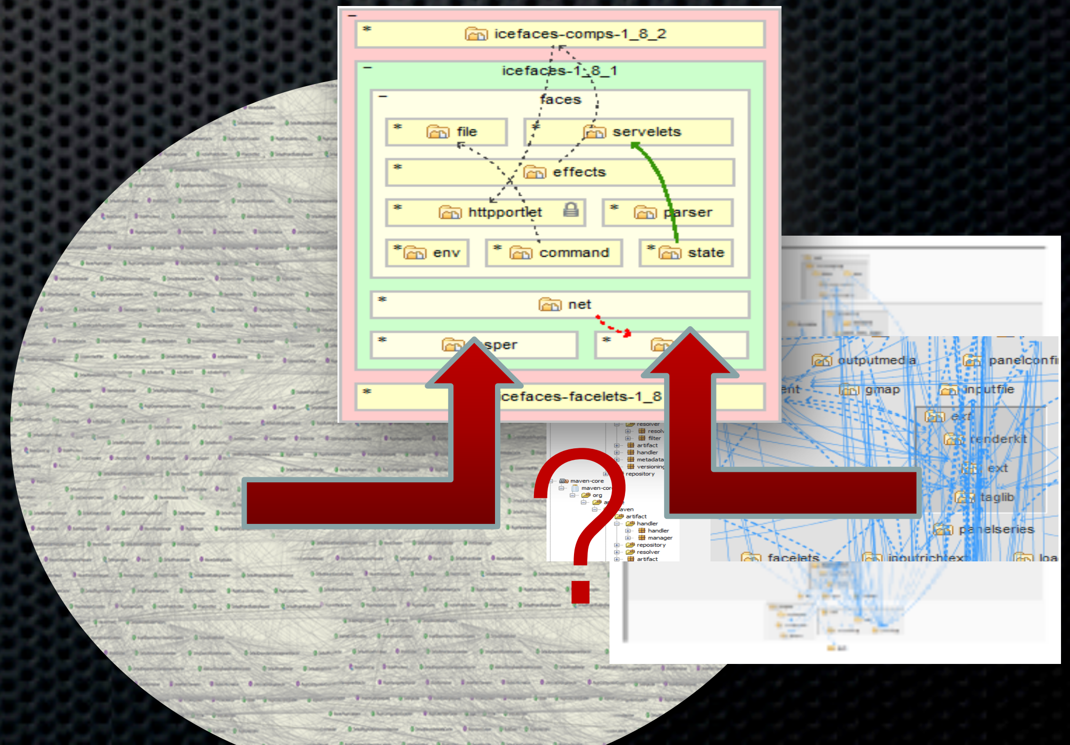
So “well-structured containment” is…



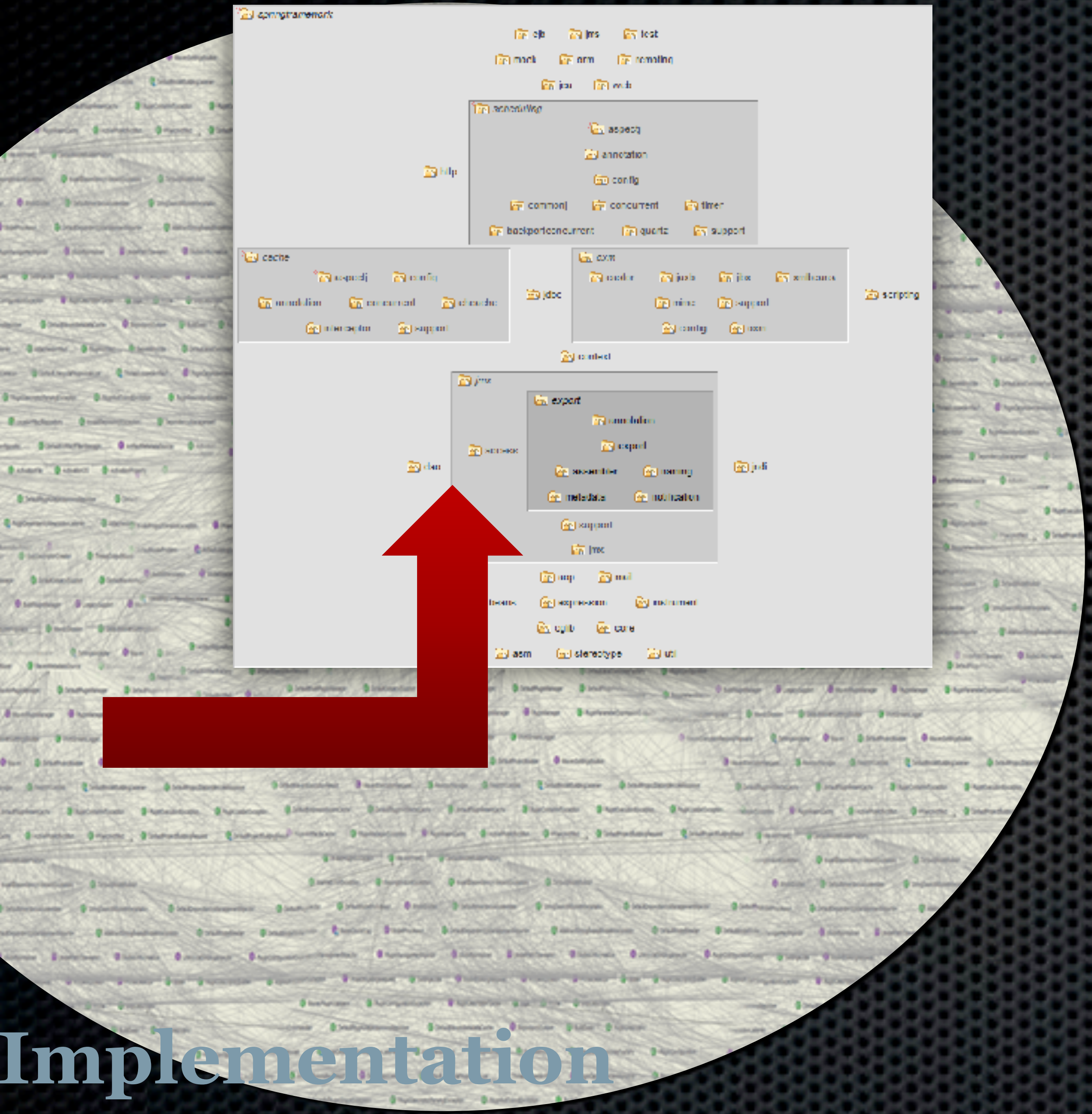
- No “tangled” containers
- No “fat” containers
- *… a foundation for “architecture”*
  - *Modules/Rules*
  - *Communication*
  - *Enforcement*
  - *Controlled evolution*

“Well-structured containment”

HOW DO WE GET IT?

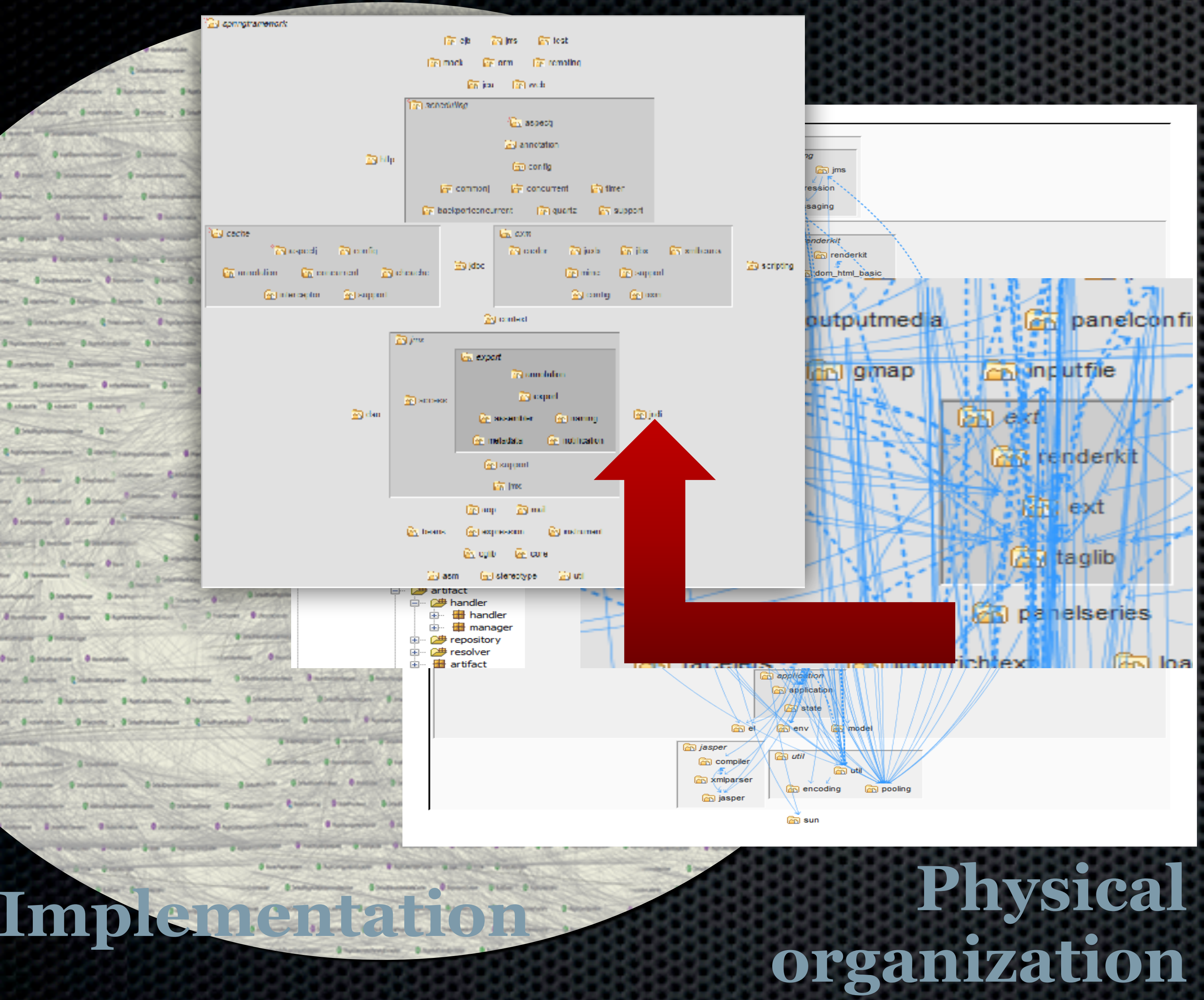


Using source files



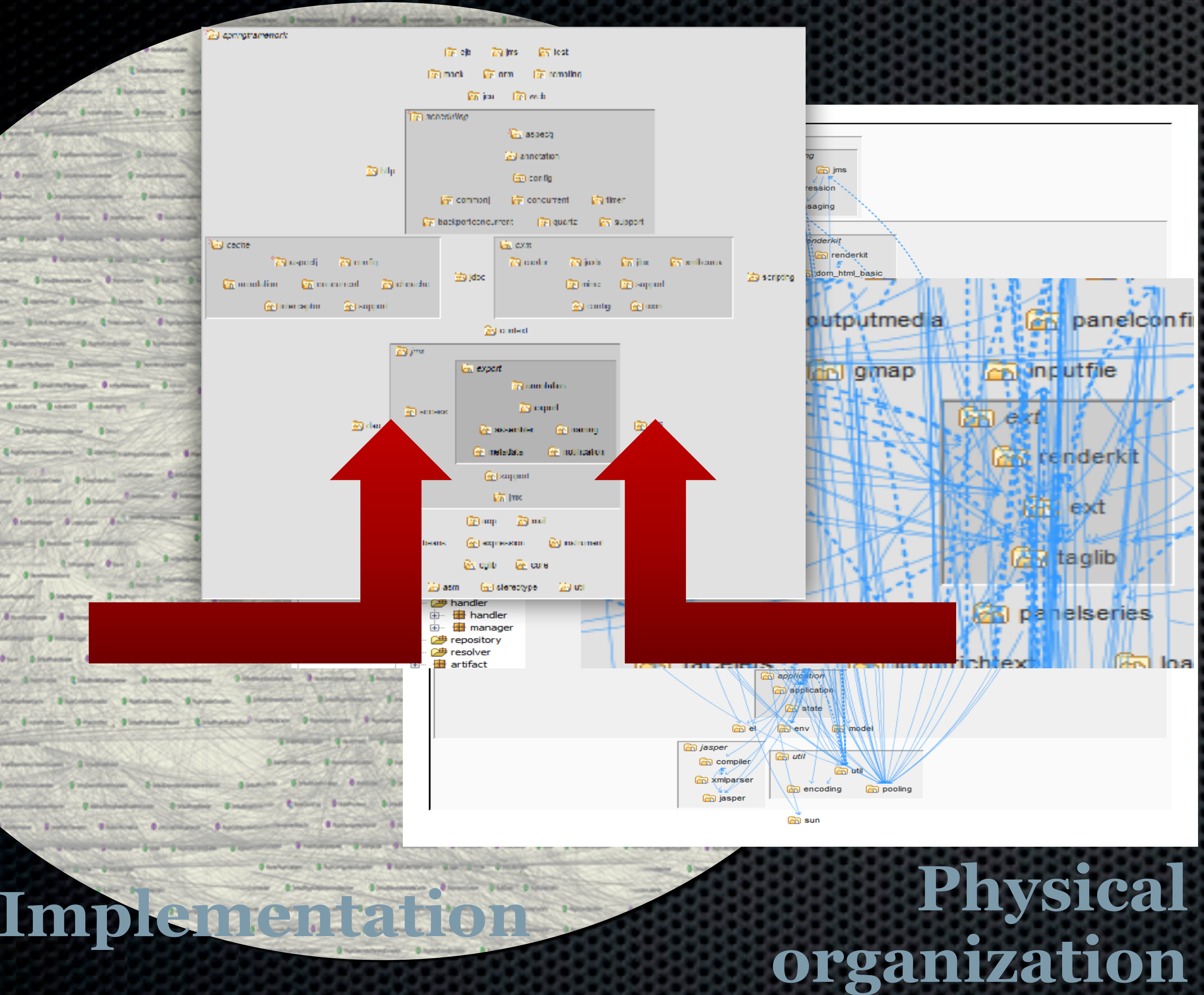
- Recursively group cohesive clusters of files
- *Bust or isolate large file-level tangles*
- Can be partly automated

Using physical organization



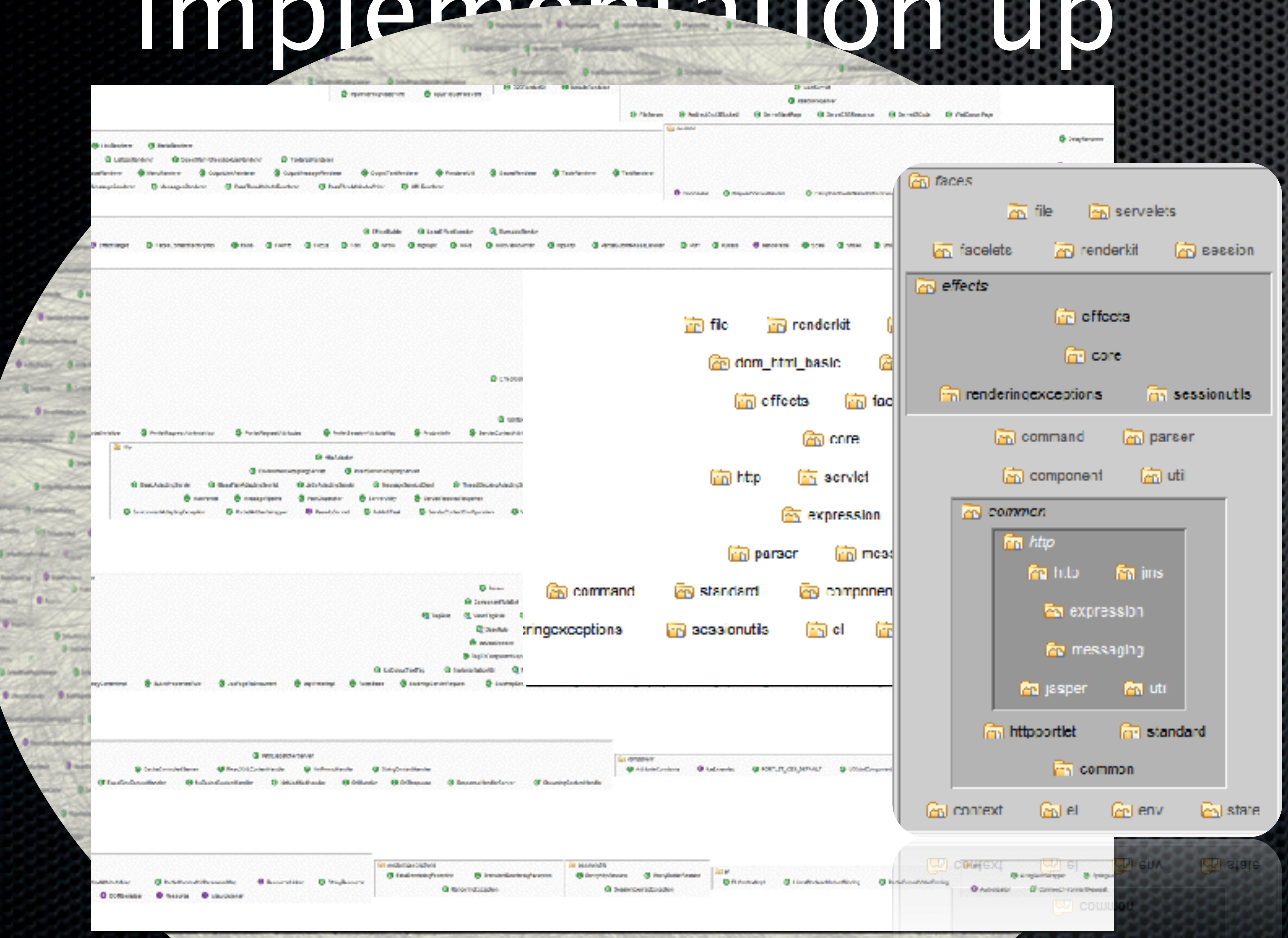
- Restructure/refactor
- Disentangle
- Preserve familiar structures
- Guided/manual reorganization
- Can be harder

Draw on both implementation and existing physical organization



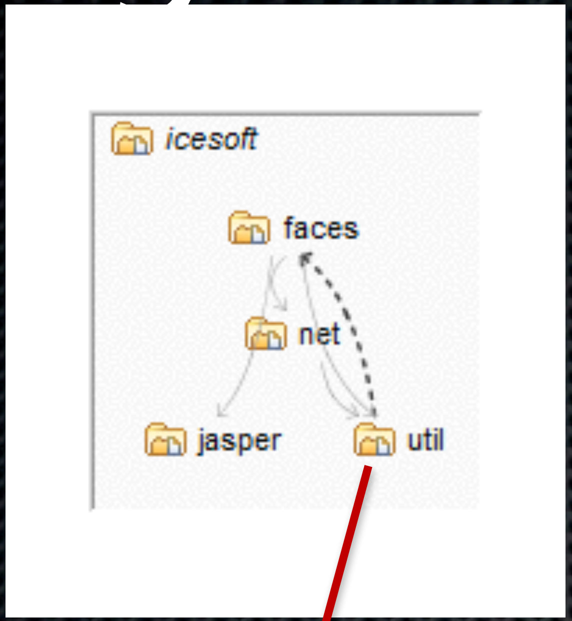
- Use the physical organization where it is reasonably well-structured
- Build a new structure where it isn't

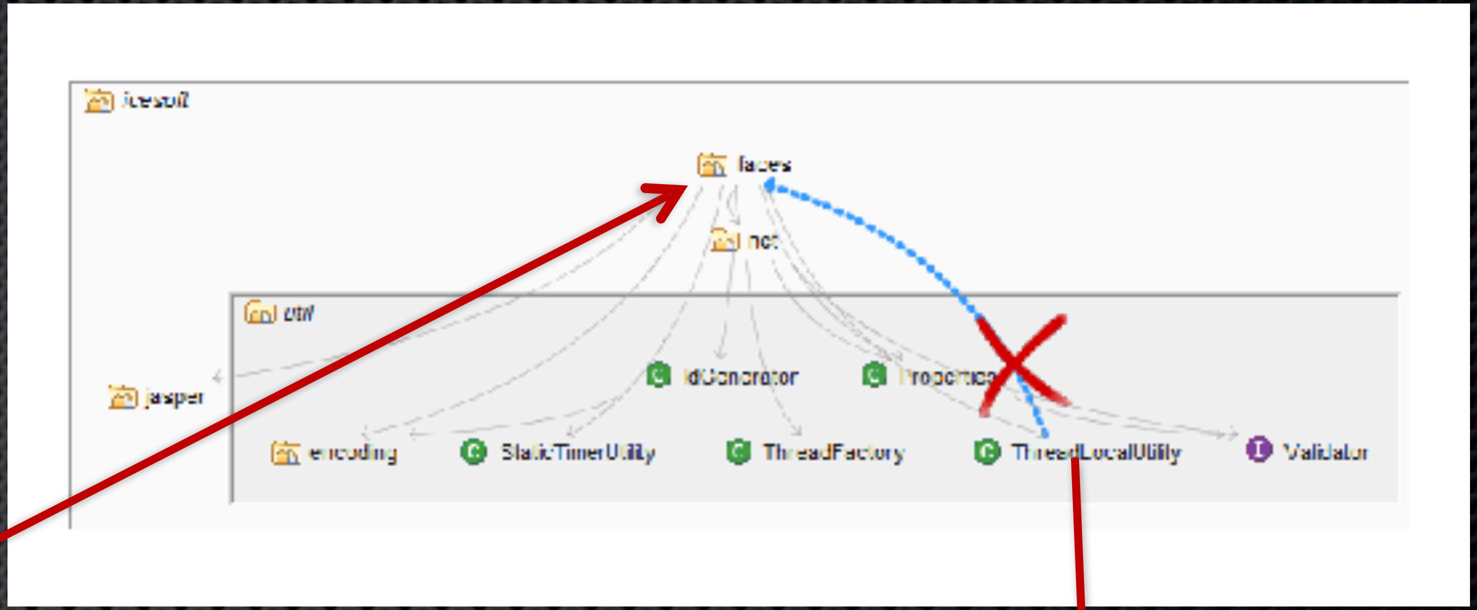
Building containment from implementation up

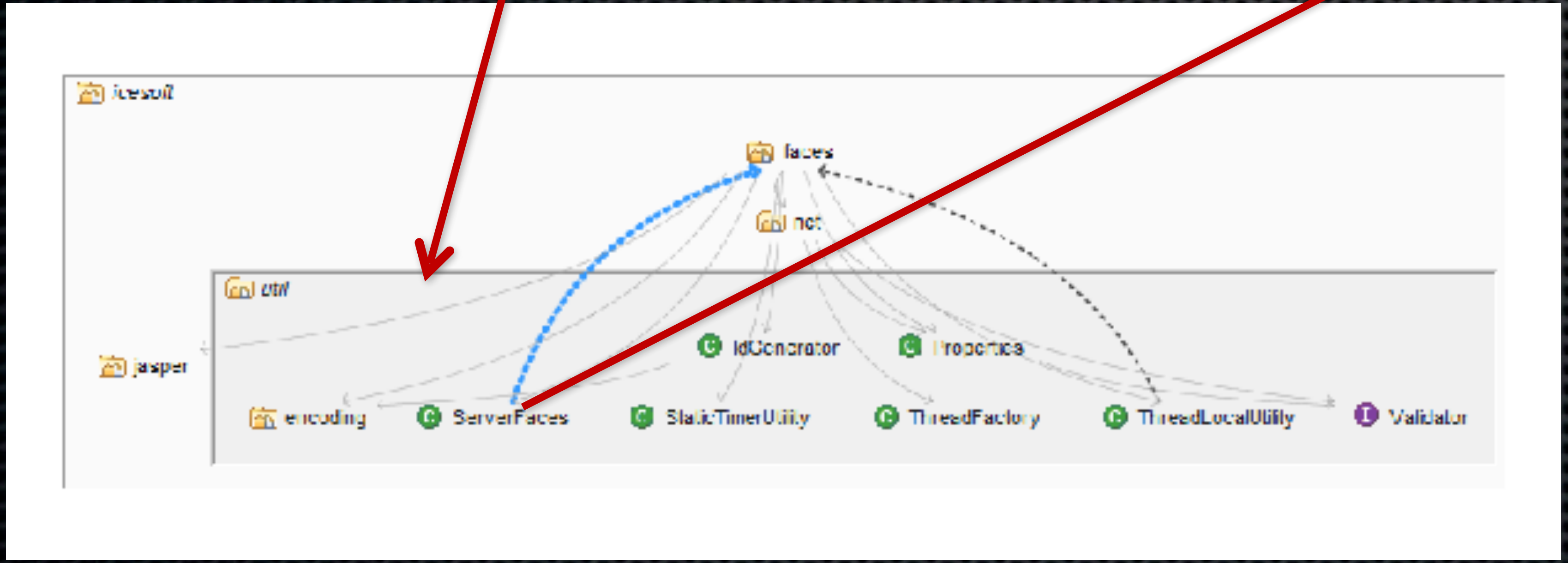


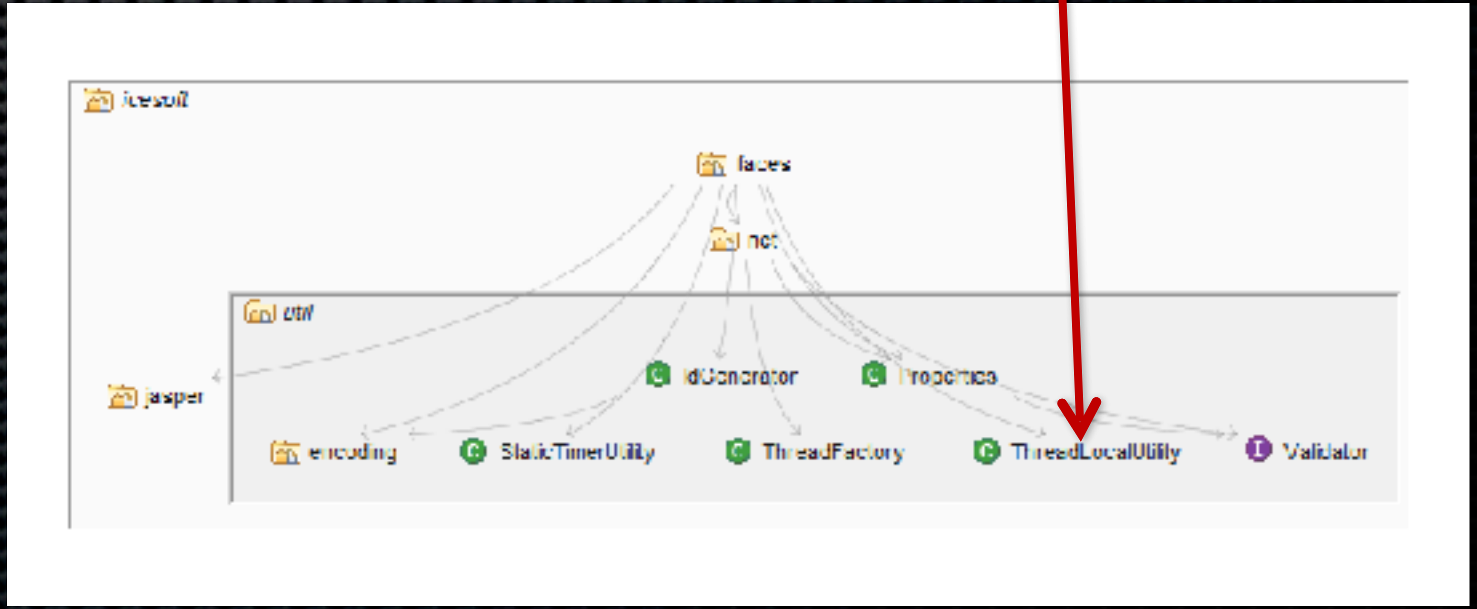
- Find "cohesive clusters" of source files
- (use automation)
- Wrap them into containers
- Find cohesive clusters of containers
- Wrap them into higher level containers
- Repeat

Restructuring physical organization



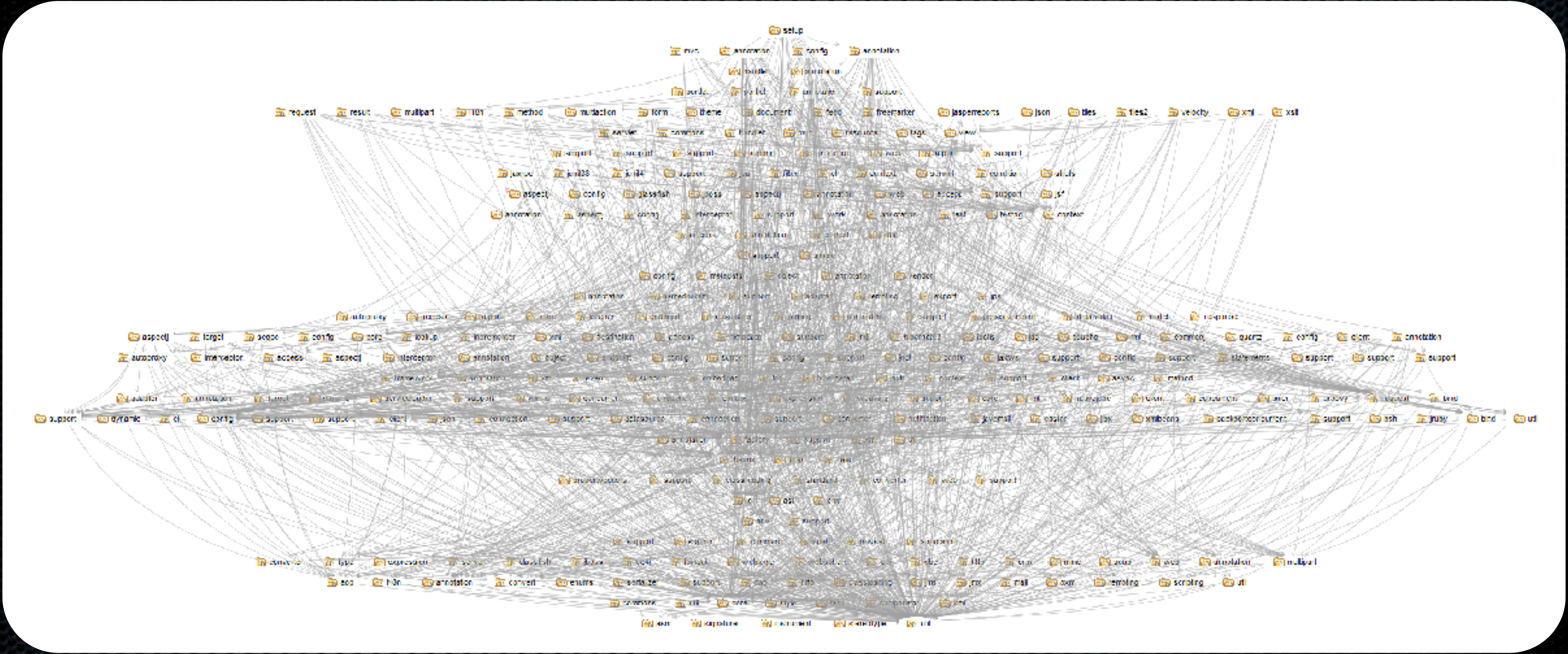




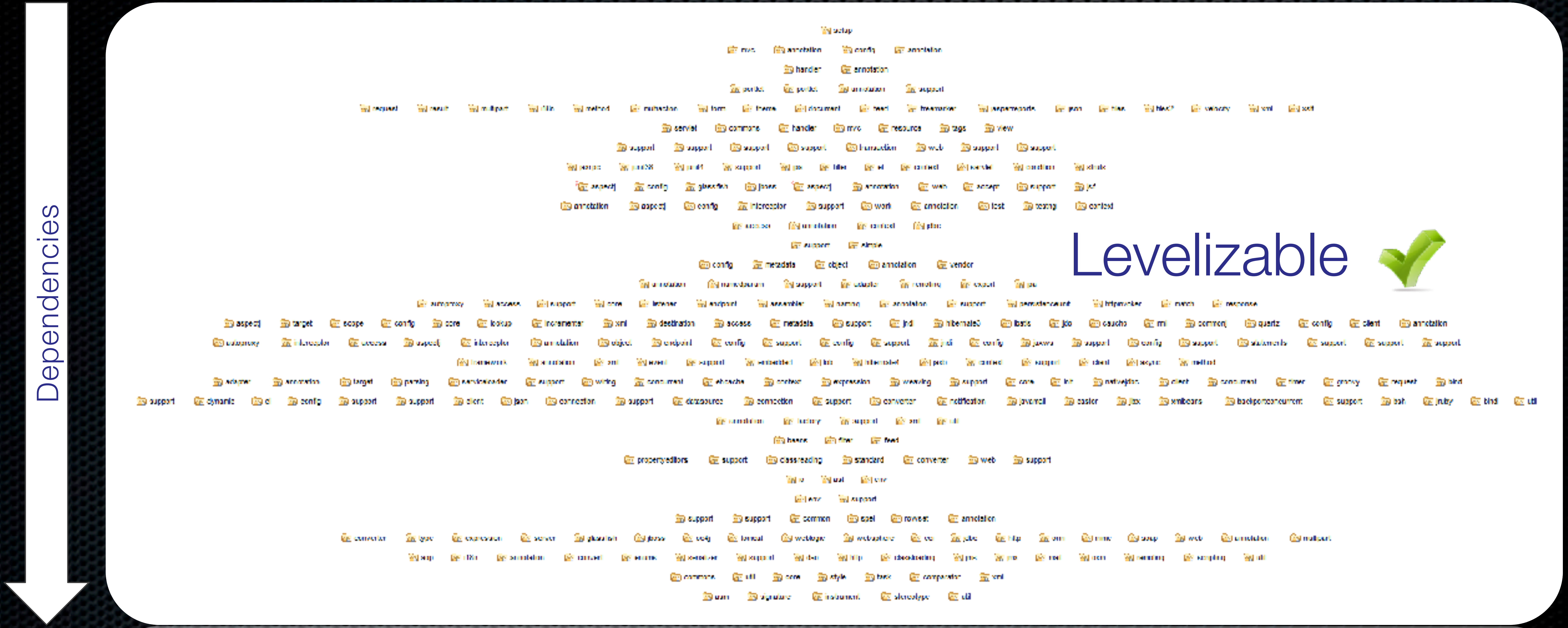


Key concept: Levelization

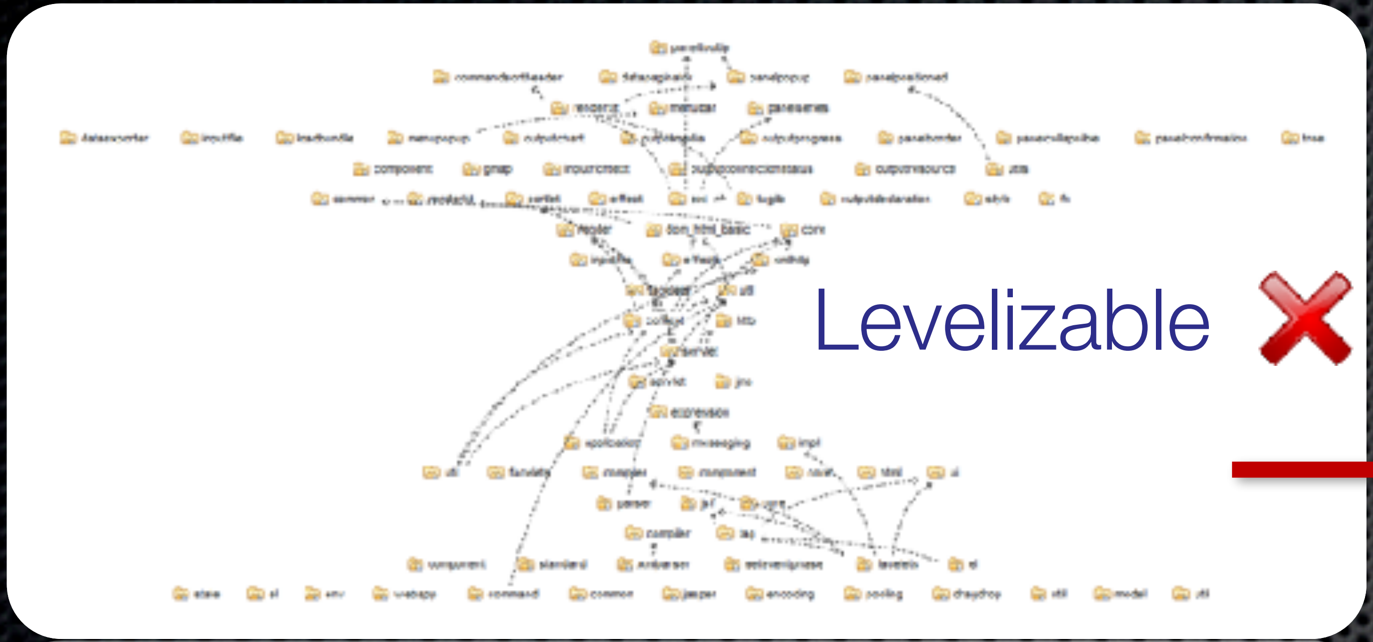
Levelization

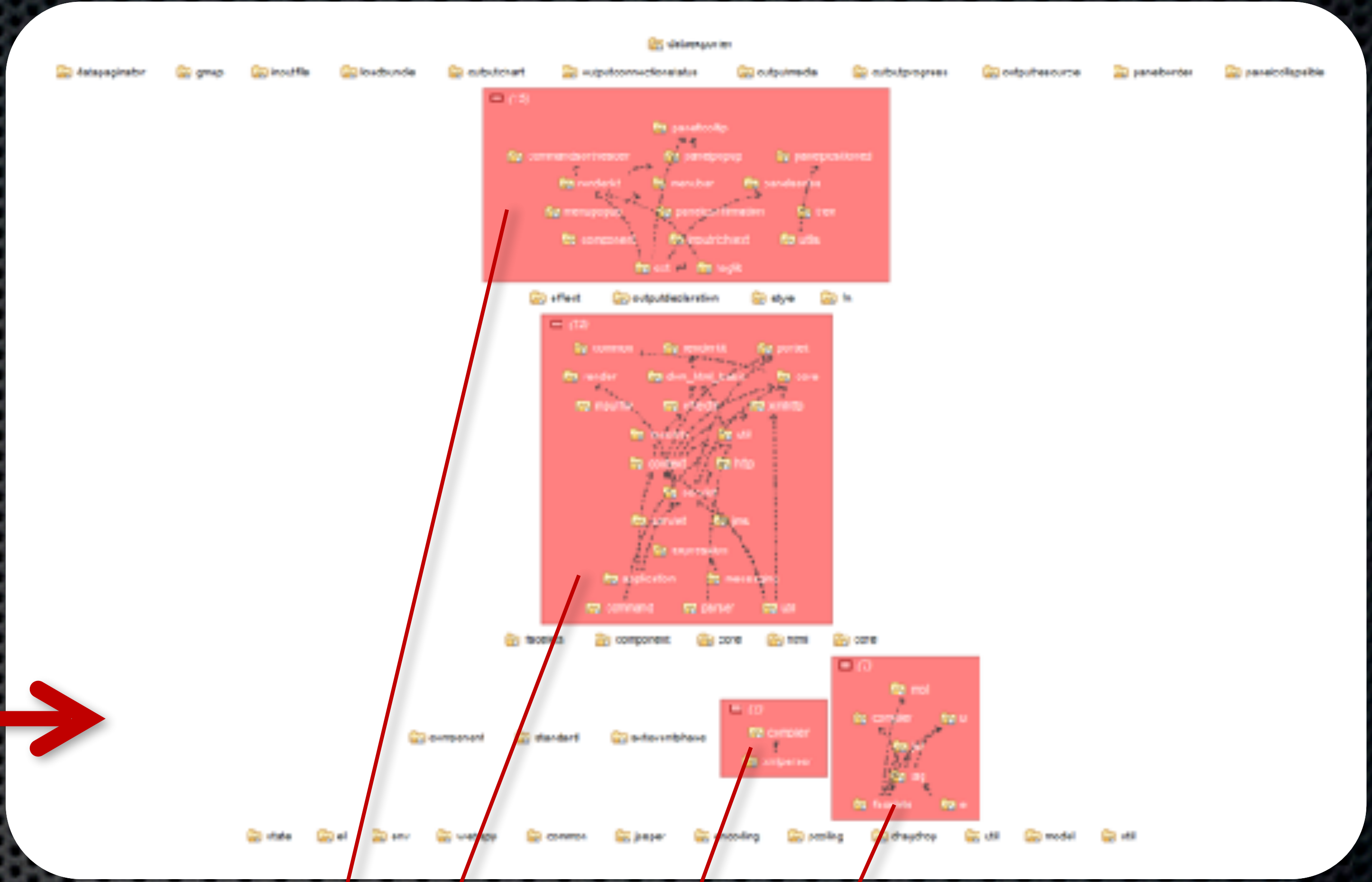


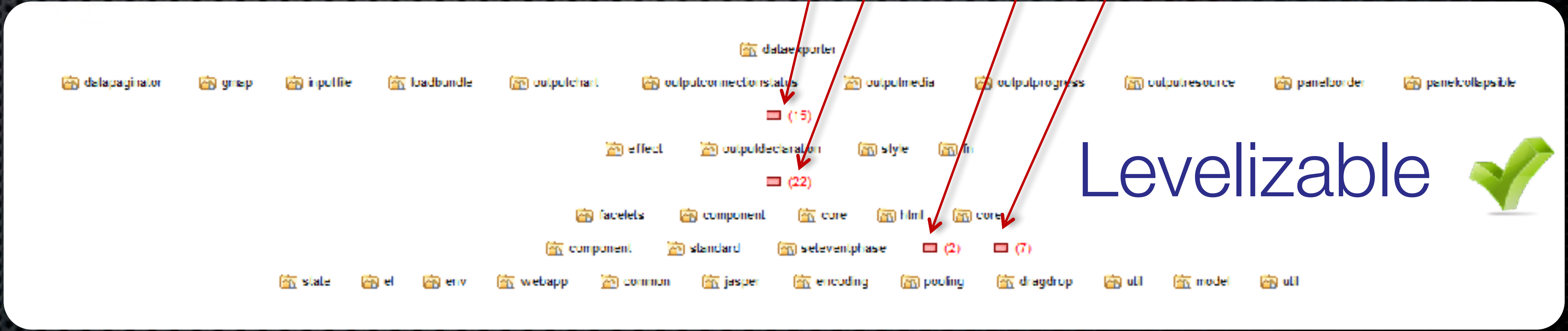
Levelization



"Tangle" - Every vertex reachable from every other vertex

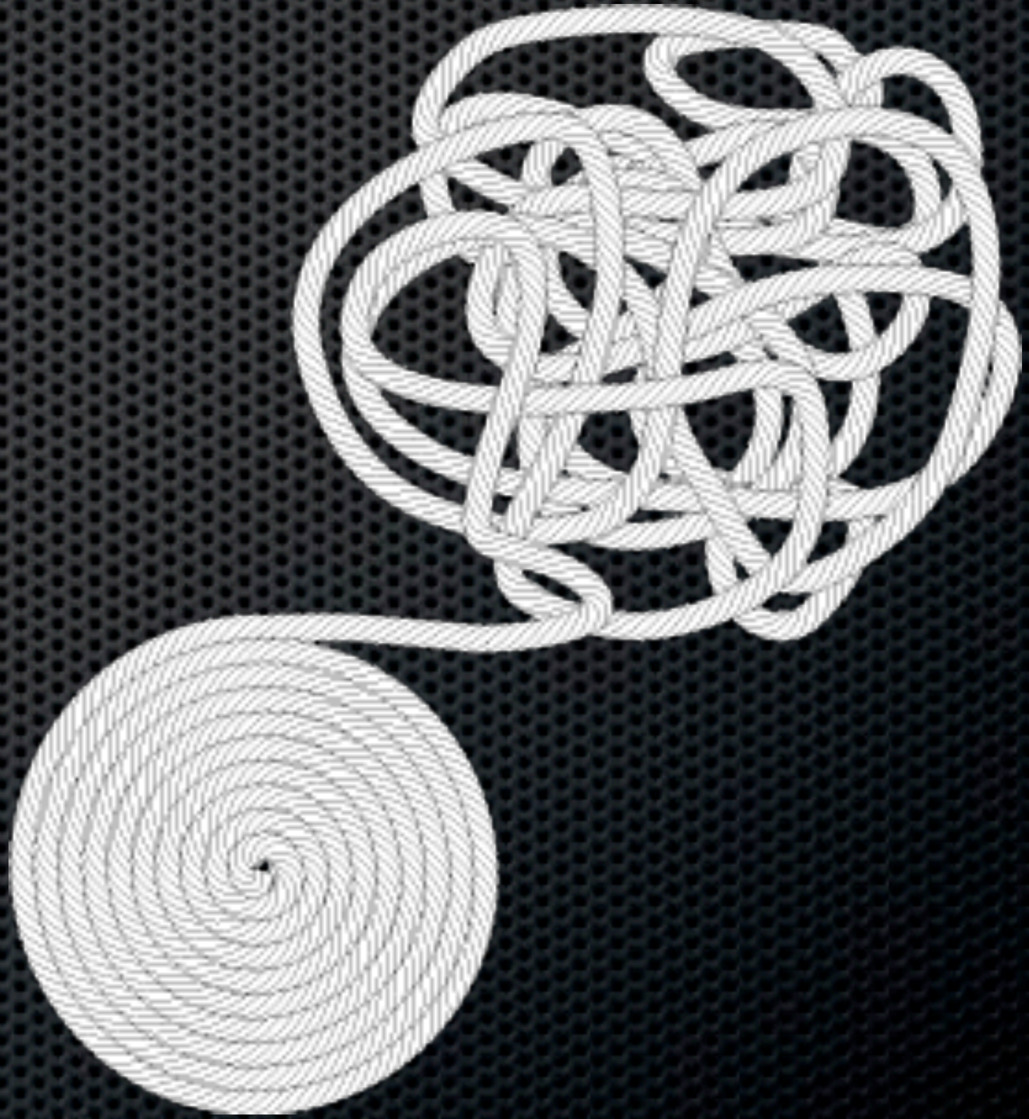






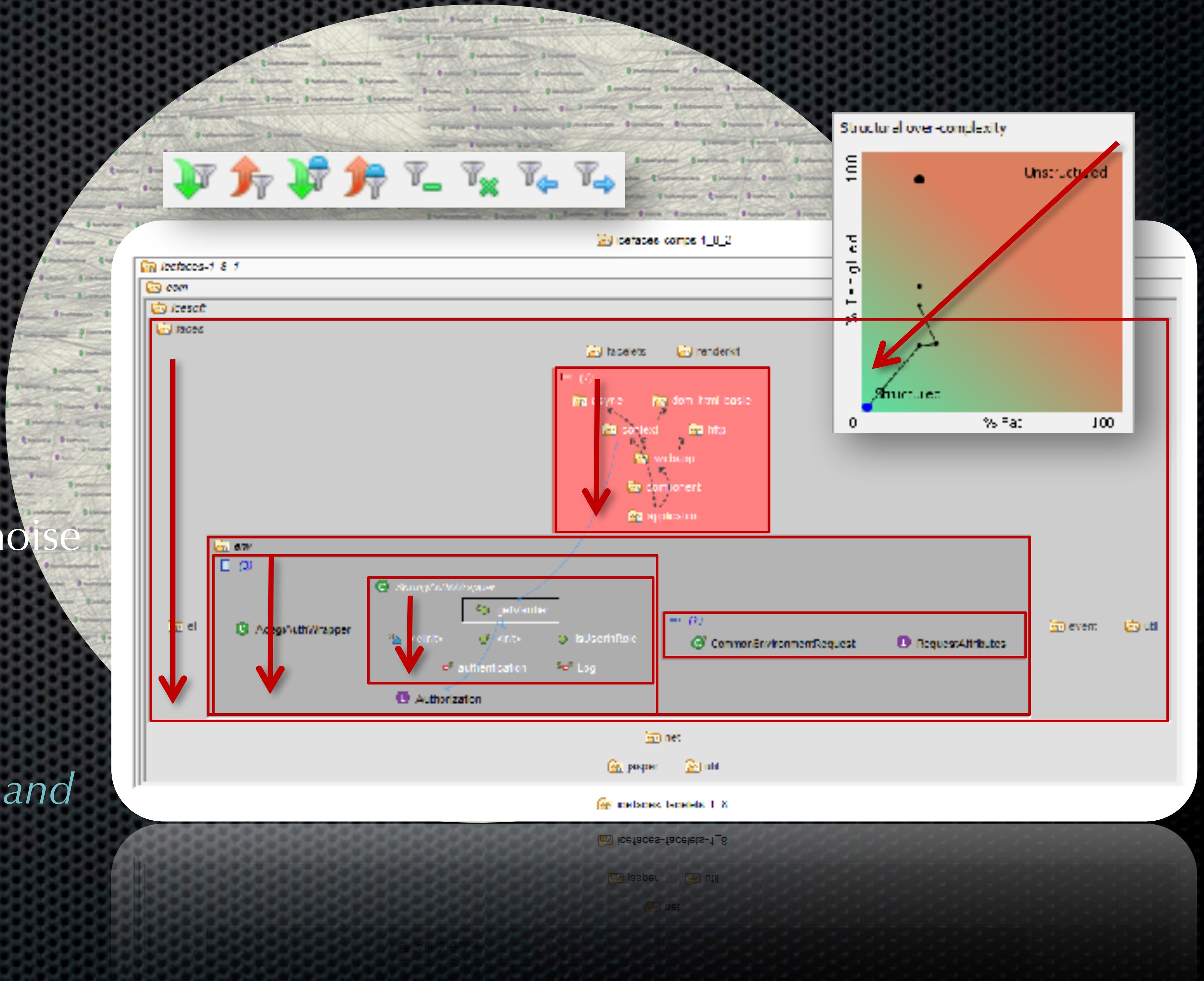
Structure101

Making it real



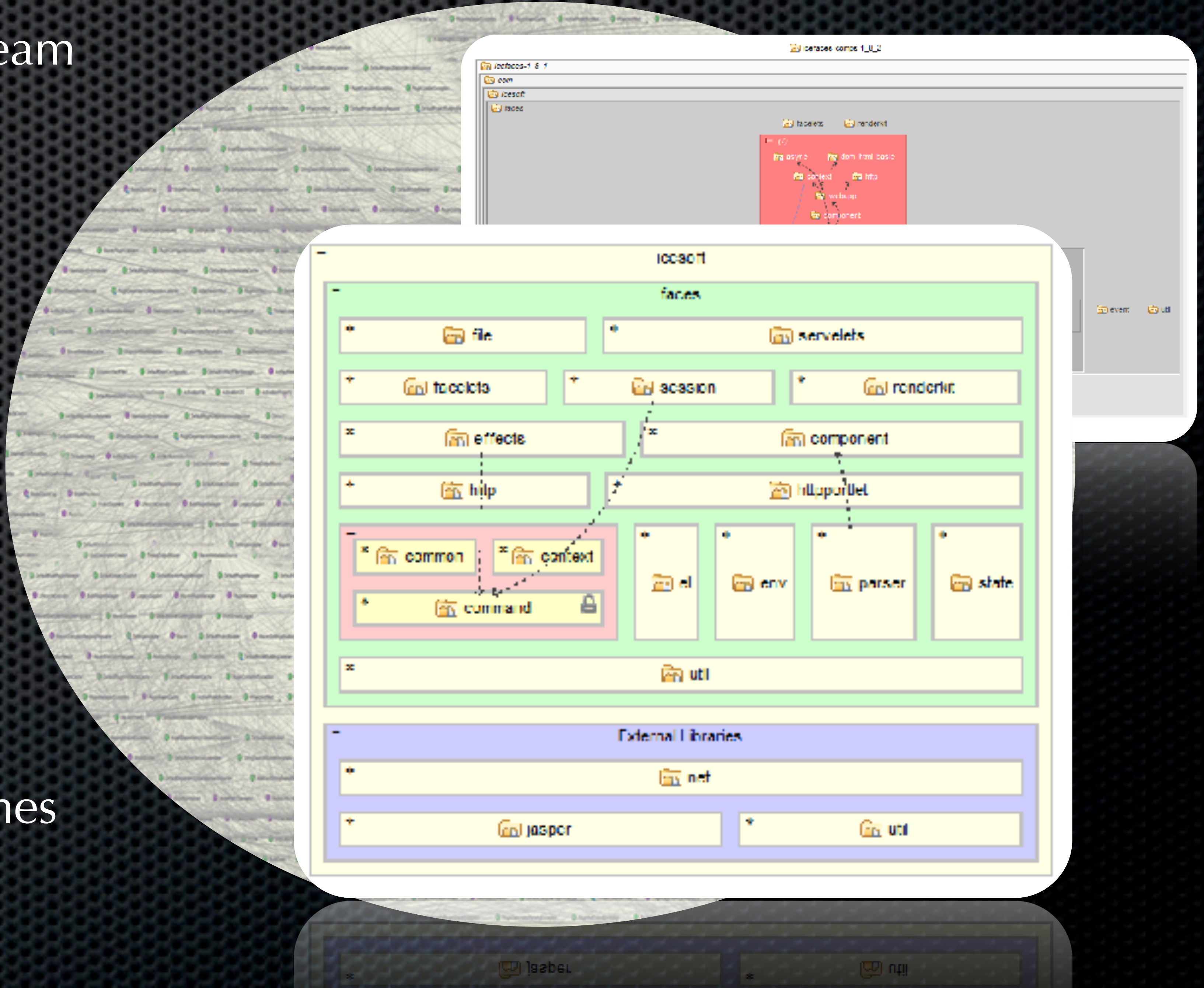
The Levelized Structure Map (LSM)

- Designed specifically for *containment modelling*
- *Expand/collapse* depth of scope
- *Auto-groups* tangles, cohesive clusters, disconnected clusters
- *Filter* items and dependencies to reduce noise
- Can be manipulated *interactively or automatically* to create well-structured containment model
- *Items are always levelized at every scope and after every change*



The Architecture Diagrams

- *Communicate* important aspects of model with team
- *Define rules* for a containment model
- Cells *map* to code by patterns
- Dependencies *should* flow down
- Cell positioning expresses *many* rules, visually, intuitively
- Can have *many* diagrams
- *You define layering and visibility – not changed automatically*
- Used to *check* code changes at edit and build times

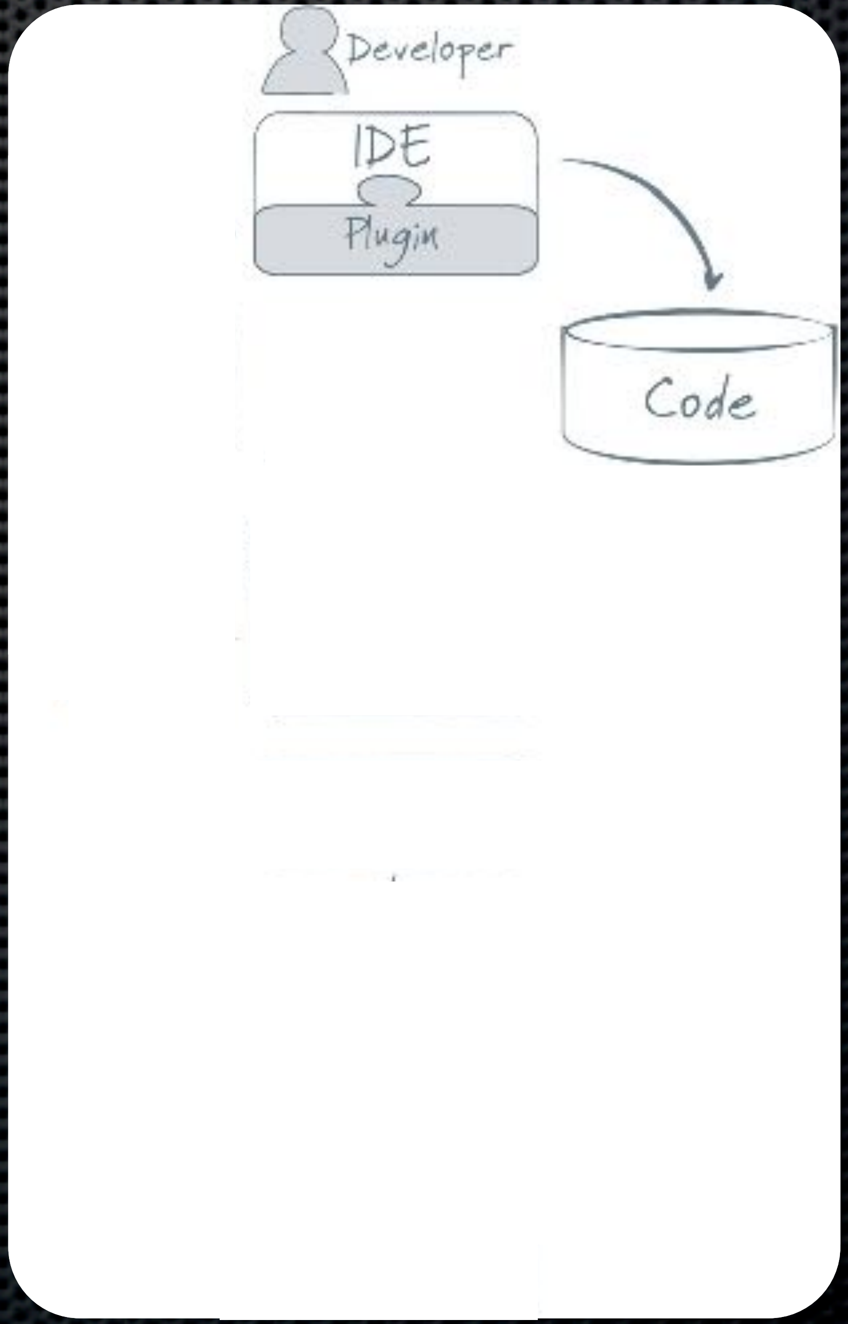


Step 1: Discover and define your architecture

- *Bootstrap step*
- Use *LSM* to create “well-structured containment model”
    - Get “Fat” and “Tangles” close to zero
- Use the *Architecture diagrams* to define dependency rules for your model
- Share your architecture

Step 2: Architecture-guided development

- Communicate
  - Compile-time checking
  - Build-time checking
  - Reporting
- Evolve
  - Update architecture when required
  - Adjust architecture ahead of development



Developer
IDE
Plugin
Code