

# 10 Years of Research around Software Evolution: A “Florilege”

<http://stephane.ducasse.free.fr>  
Stéphane Ducasse



# Me in a Nutshell

RMOD team (7 permanents, 20 people)

4 years scientific advisors of Inria Lille (300 people)

Wrote several open-source books

One of the leader of the Pharo community

- <http://www.pharo.org>

One of the past core dev of Moose data and code analysis platform

- <http://moosetechnology.org>

Co-founder of <http://www.synectique.eu>

**synectique**  
Inventive Analysis

# Roadmap

Food for thought

Context

Software Maps

Evolution the large

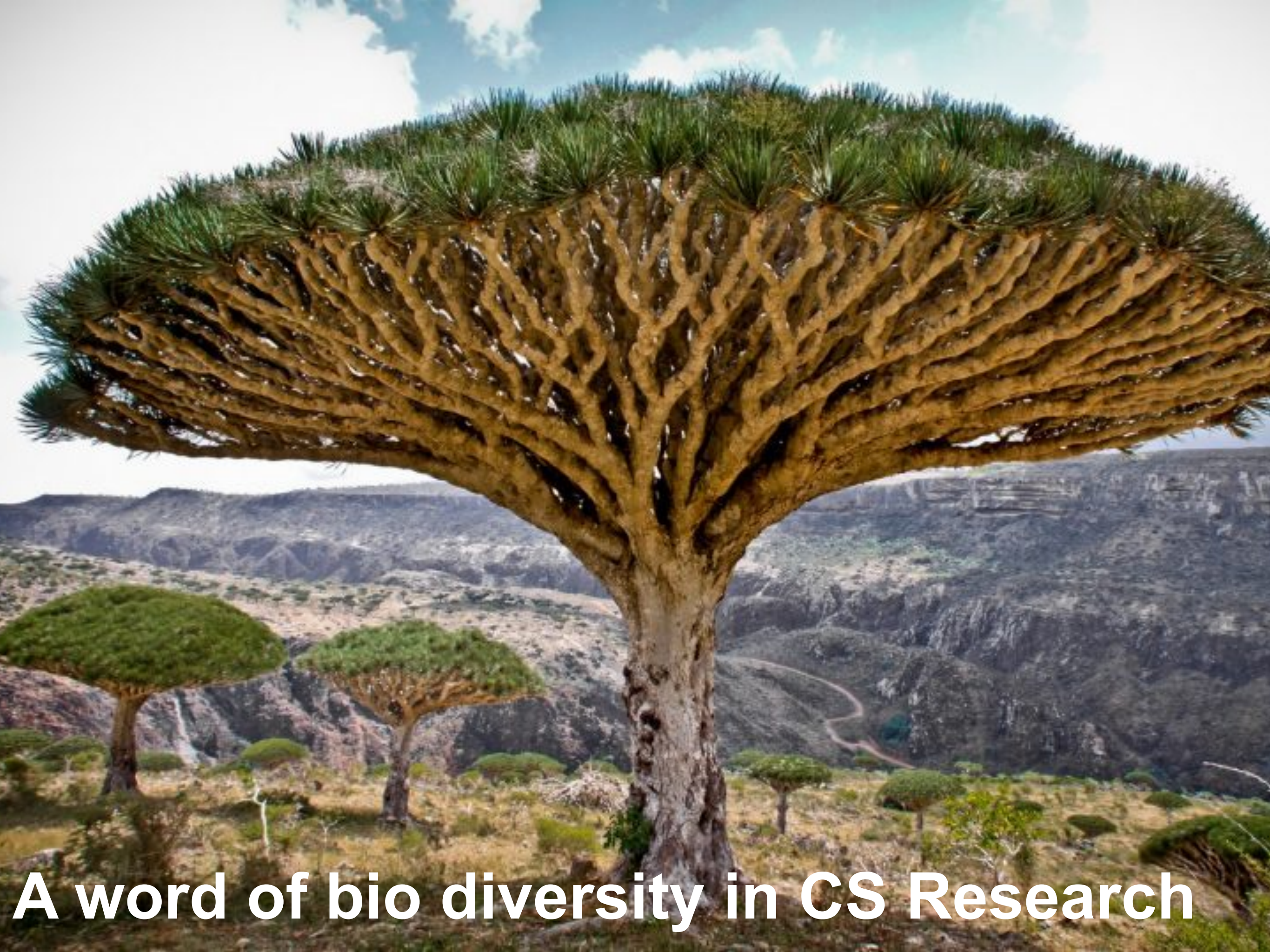
Runtime

Others

Current work







**A word of bio diversity in CS Research**



**In biology  
diversity is a measure of  
ecosystem wealth**

# Endemic

adjective, Also, endemical  
1.

natural to or characteristic of a  
specific people or place; native;  
indigenous



**An endemic species is one whose habitat is restricted to a particular area.**



As of 1990, Socrota counted  
700 endemic species





**As of 1999, England counted 47  
floral endemic species**

...



# How do we value diversity in CS research?

**Can we do research on something else than Java/JavaScript?**



# How people escaped Cobol?

**How people escaped Cobol?  
How will us escape Java?**

**How can we influence that the next language will exhibit different properties if we do not explore different approaches?**



# Final thought about career evaluation...

How many papers is worth a book?

How many papers is worth a book for non-academix?

How many papers for a *real* end users?



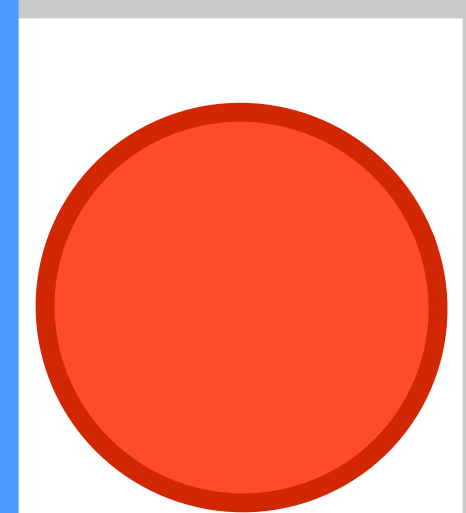
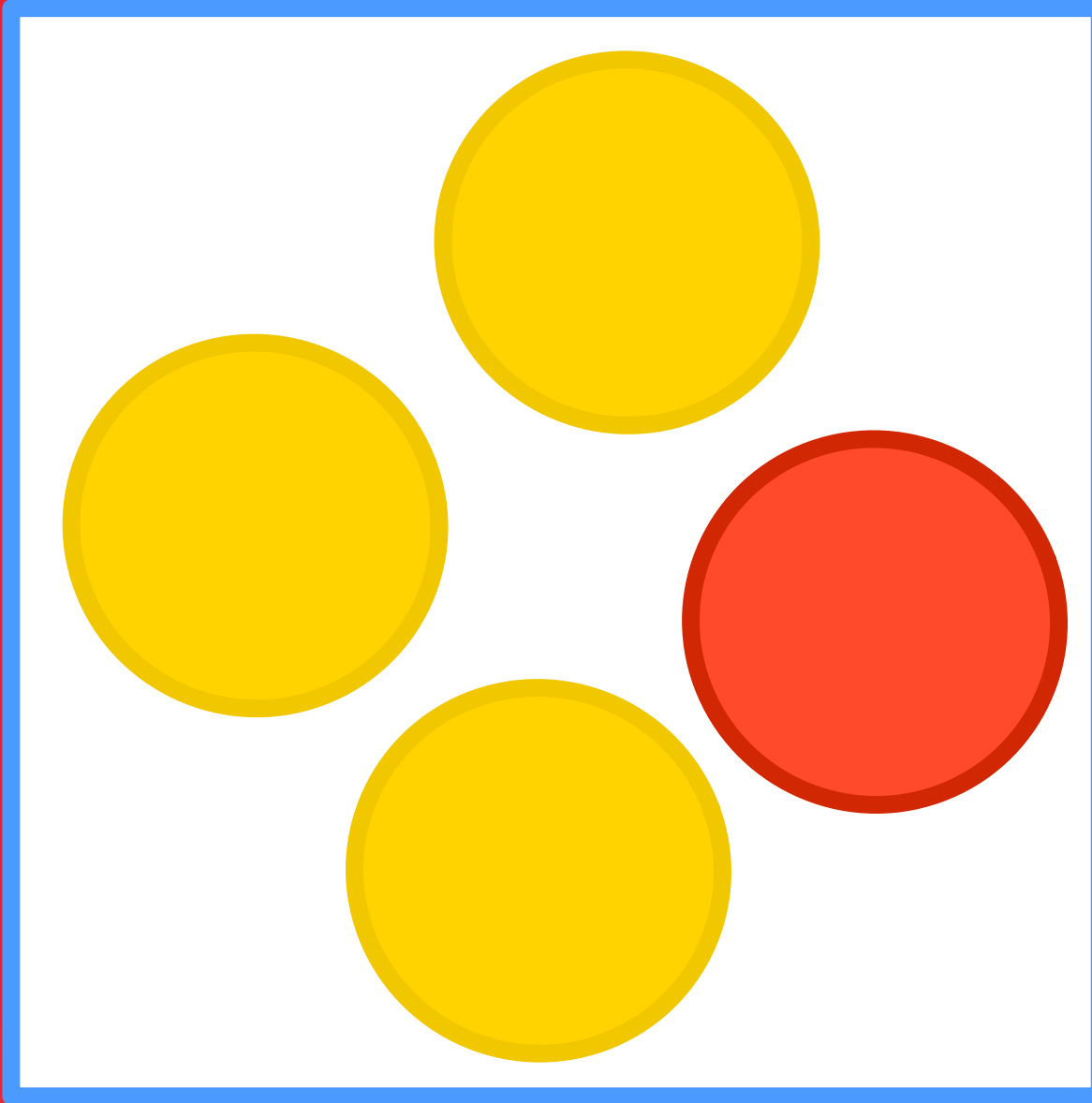


Now the fun

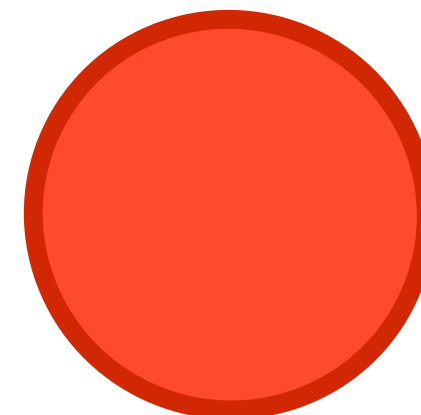


Lo

vision



**RMoD**





# Long term research vision

How to build and evolve  
***ever-running*** software  
systems?

# Ever-running systems

- Systems that last 10-30 Years+
- Also Systems that we cannot stop
  - 30 min downtime per year, fine for extra minute is 30 K\$

# Objectives in synergy

## **1: How to maintain/evolve large software systems?**

Moose: a platform for software and data analysis

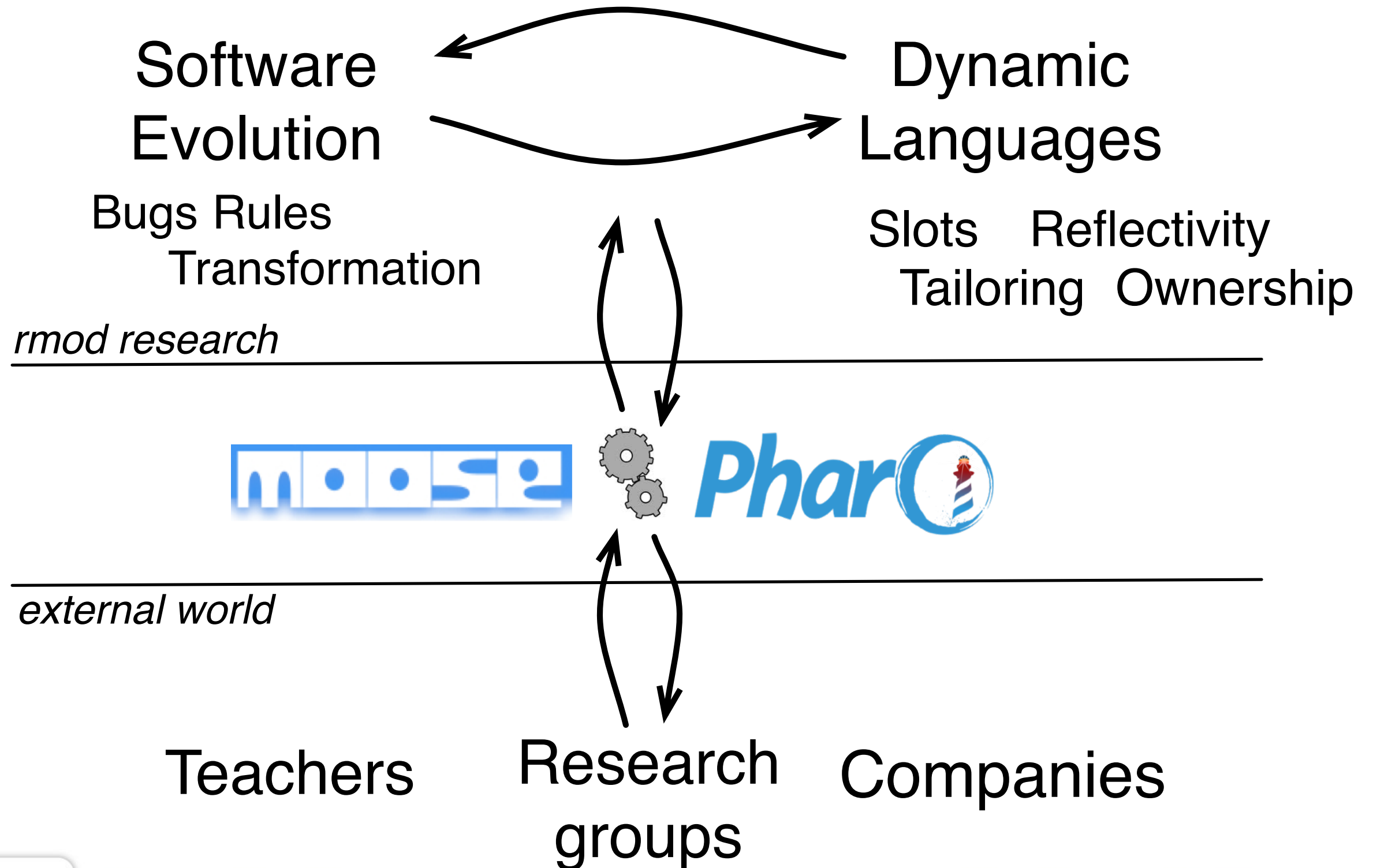
Synectique.eu

## **2: Infrastructure for ever-running systems**

## **3: Ecosystem around Pharo**

Platform&dynamic language used to create wealth and innovation

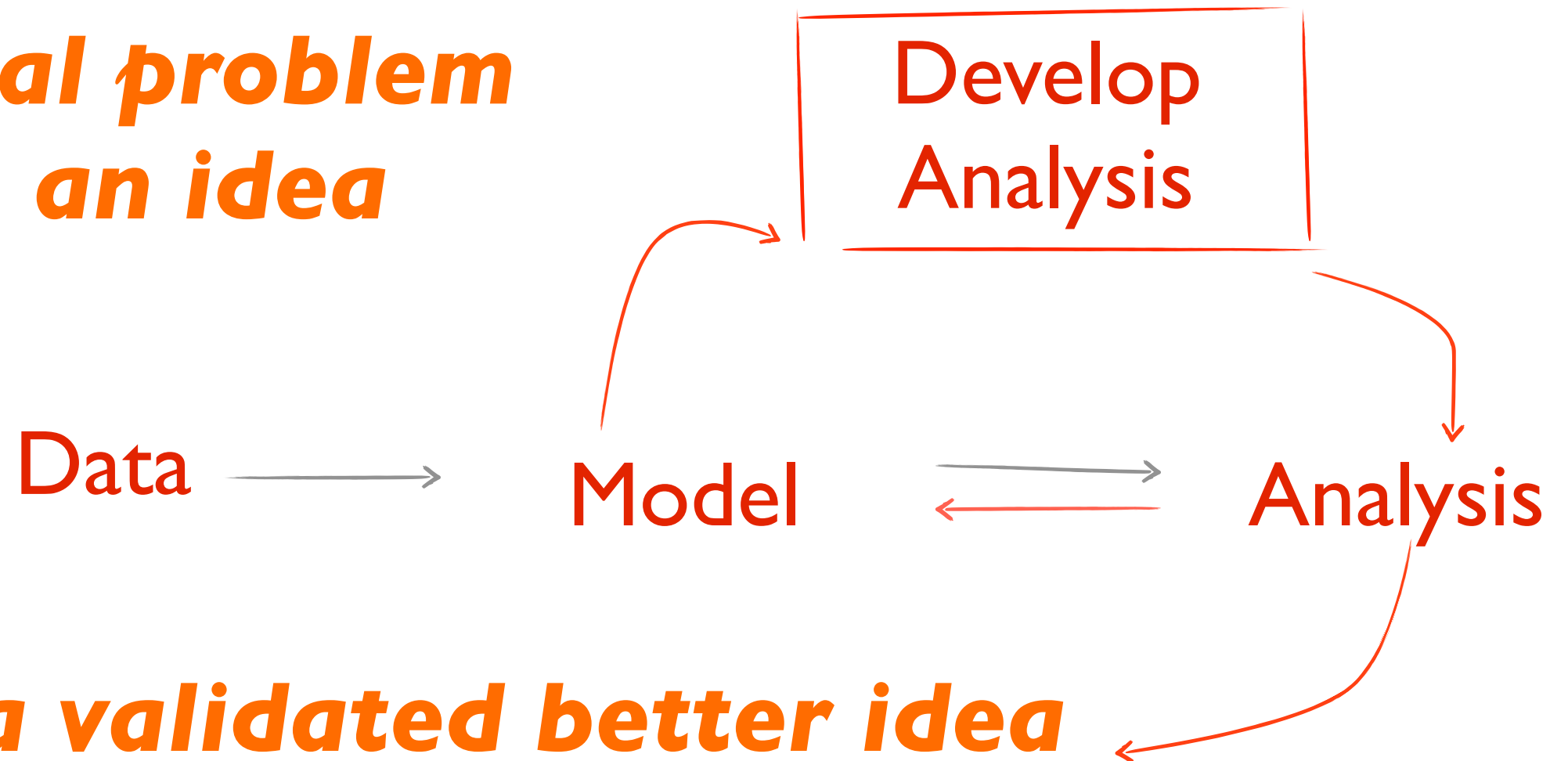
# Objectives in synergy



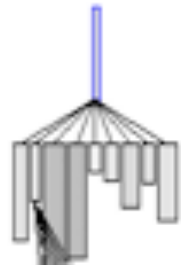


# Encouraging feedback loop

***a real problem  
+ an idea***

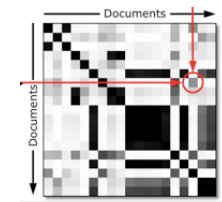
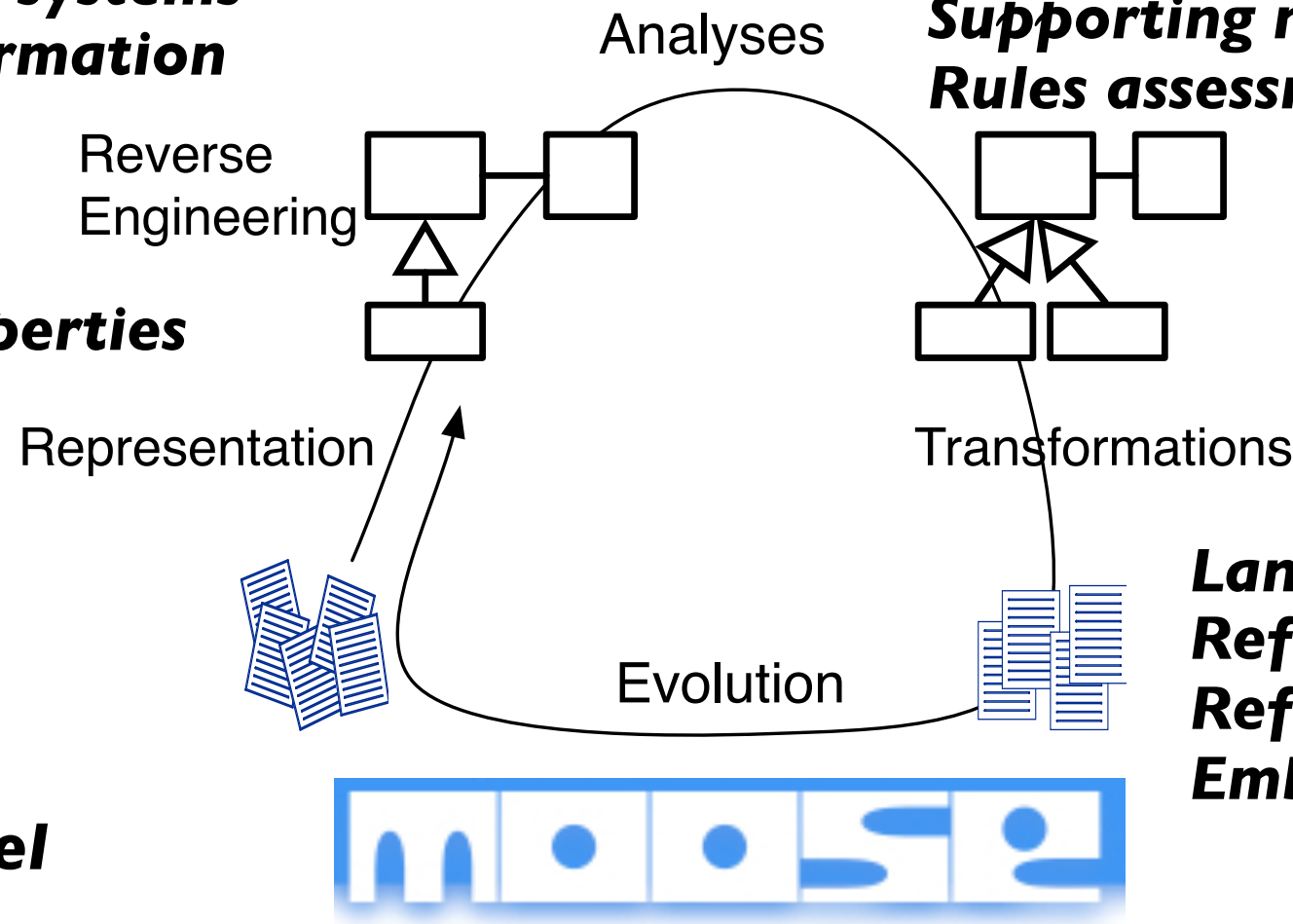


***a validated better idea  
and many more new ones***



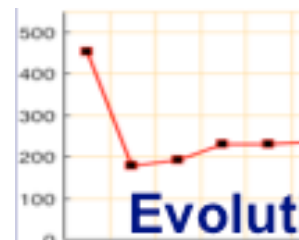
**Understanding large systems**  
**Static/Dynamic information**  
**Feature analysis**  
**Class understanding**  
**Package blueprints**  
**Distributions of properties**

**Software metrics**  
**Quality models**  
**Debuggers**  
**Duplicated code identification**  
**Tests**  
**Automatic migration**  
**Cycle and layer**  
**Supporting merge/branch**  
**Rules assessment**

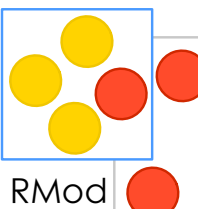


**Language Independent Refactorings**  
**Refactorings in the large**  
**Embedded procedures**

**Language meta model (FAMIX)**  
**Extensible reengineering environment**



**Reengineering Patterns**  
**Version Analyses**  
**Support Evolution**  
**Rules assessment**  
**Dependencies between/inside branches**  
**HISMO metamodel**



# Moose



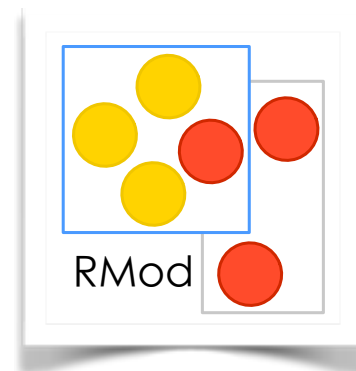
Moose is a platform for software and data analysis.  
It helps programmers craft custom analyses cheaply.

<http://moosetechnology.org>

## A community work

synect?que  
Inventive Analysis

 SCG



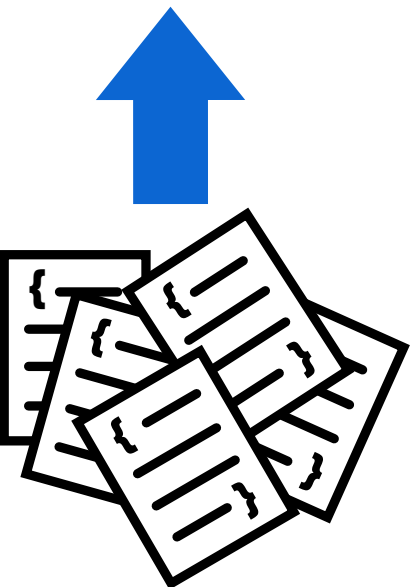
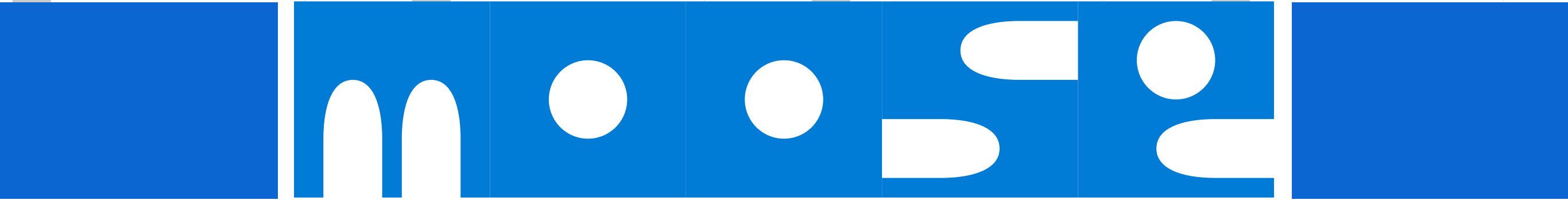
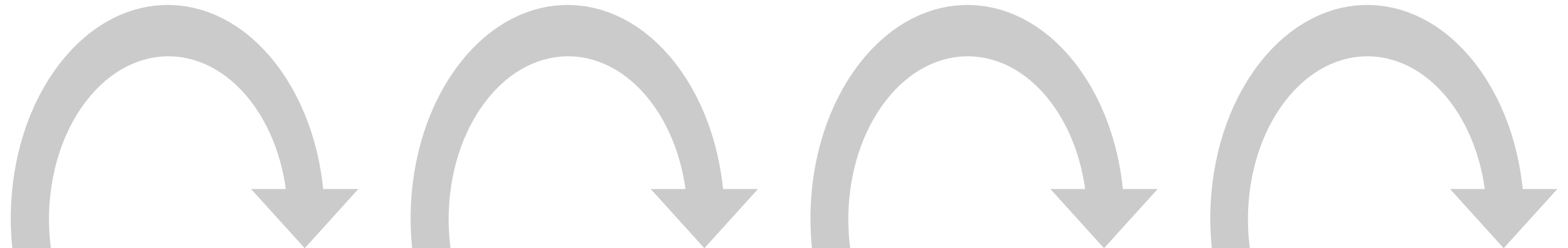
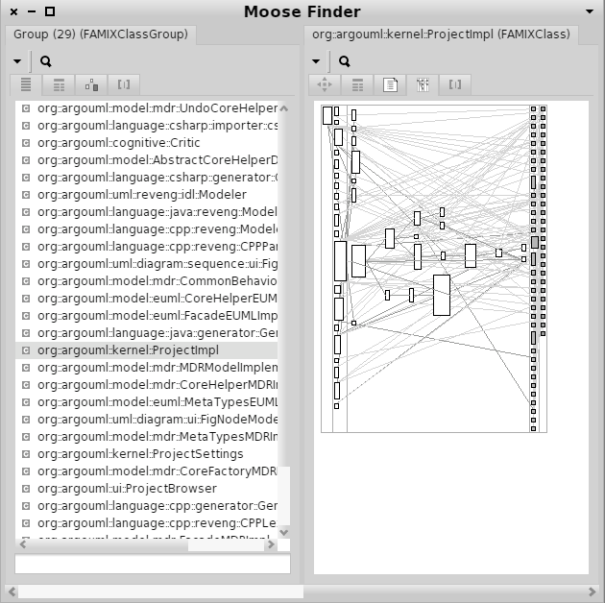
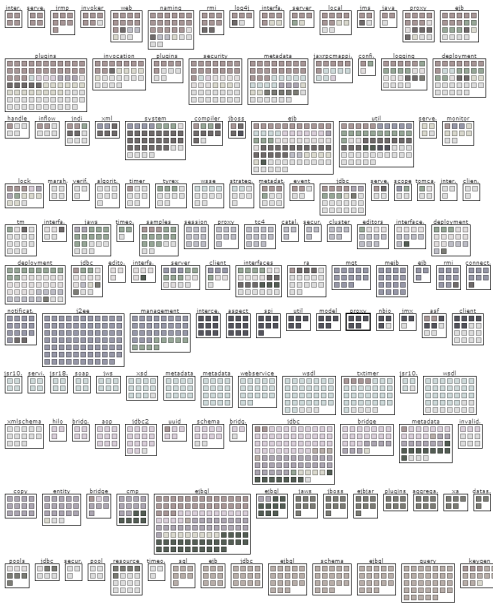
Glamorous Team  
Roassal ObjectProfile



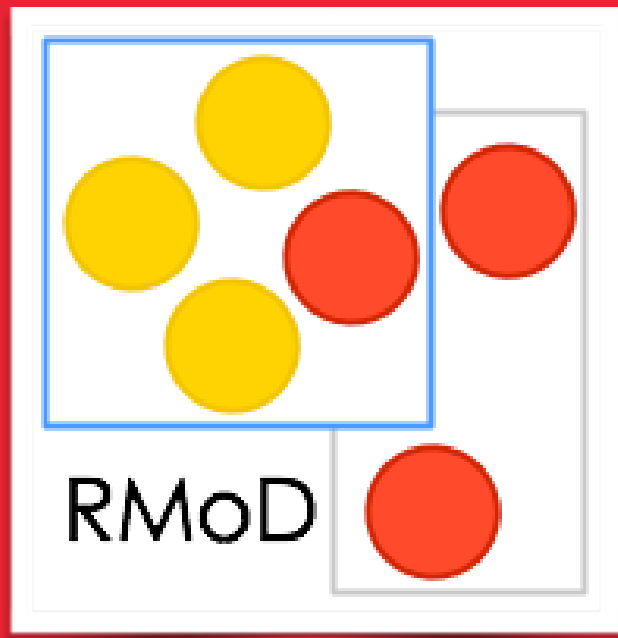
classes select: #isGod

McCabe = 21

LOC = 753,000

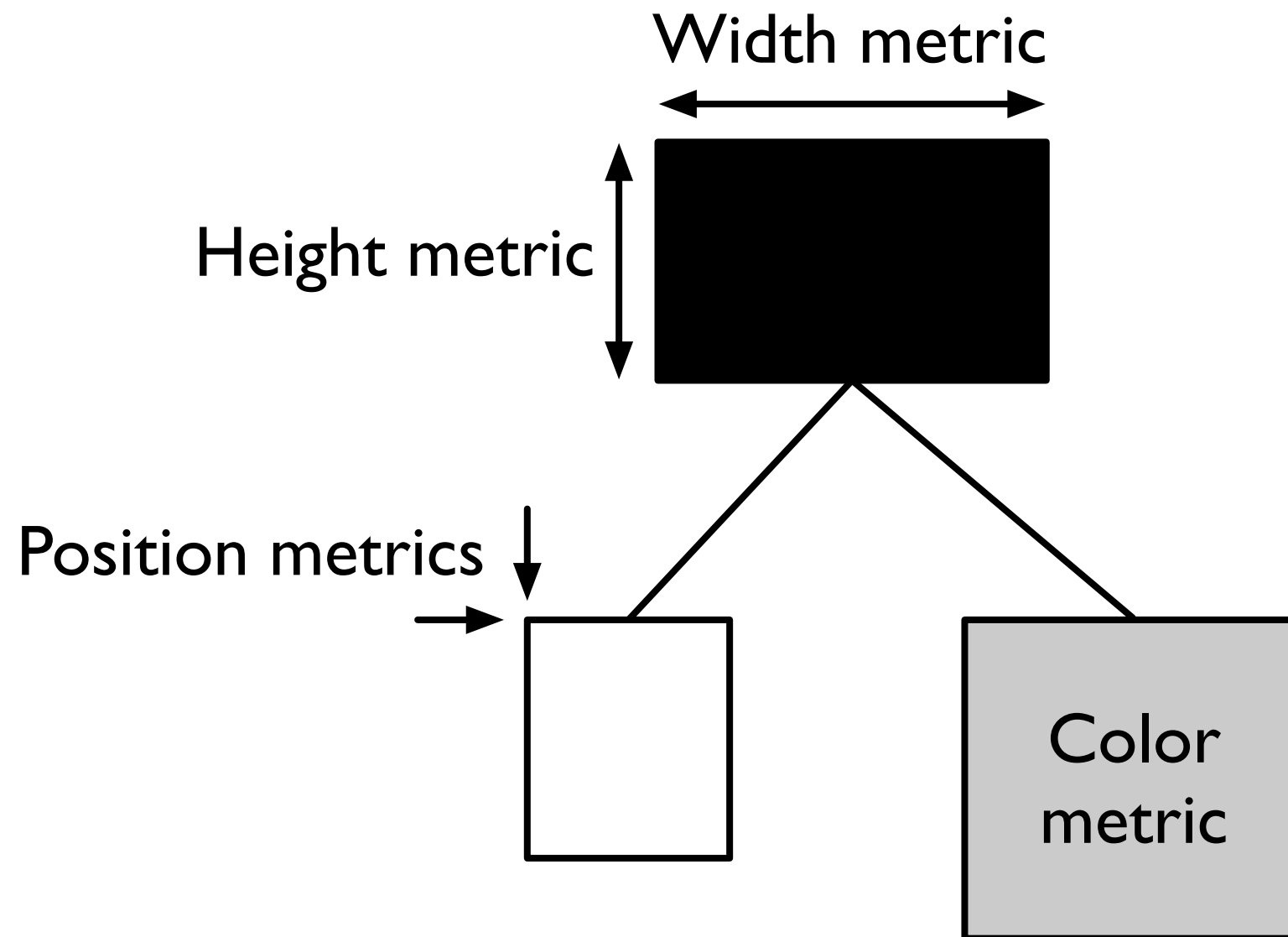




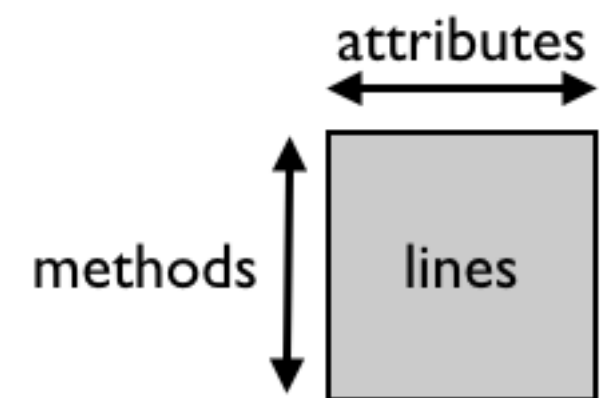
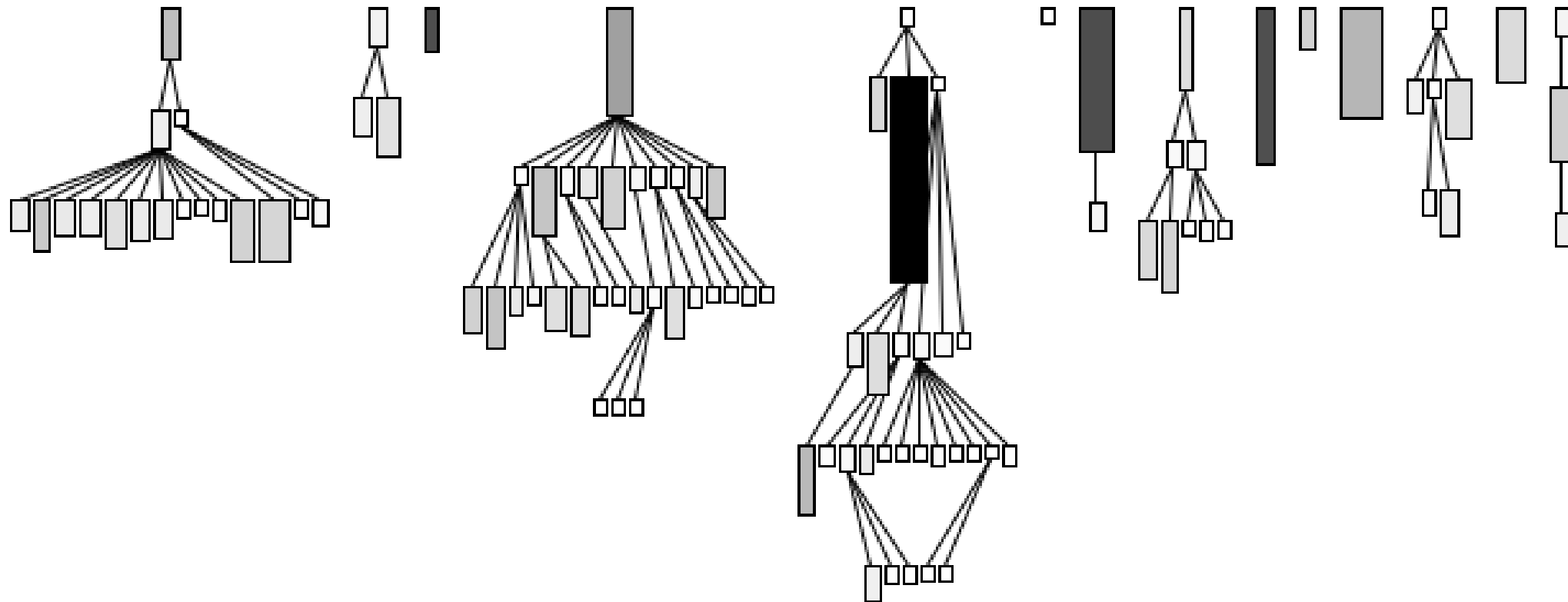


**Some software maps  
— to build yourselves at  
home**

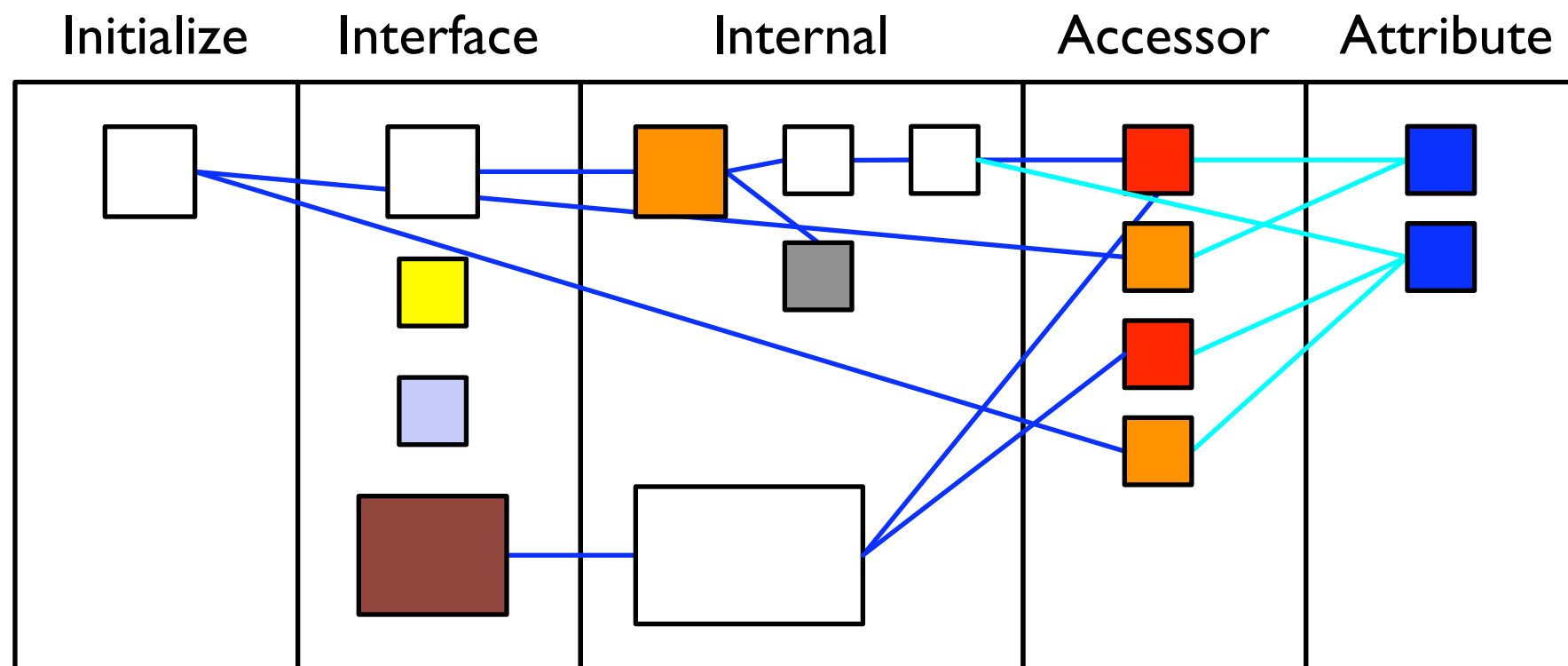
# First glance at large systems: Polymetric views [PhD Lanza]



# Understanding systems [PhD M. Lanza]

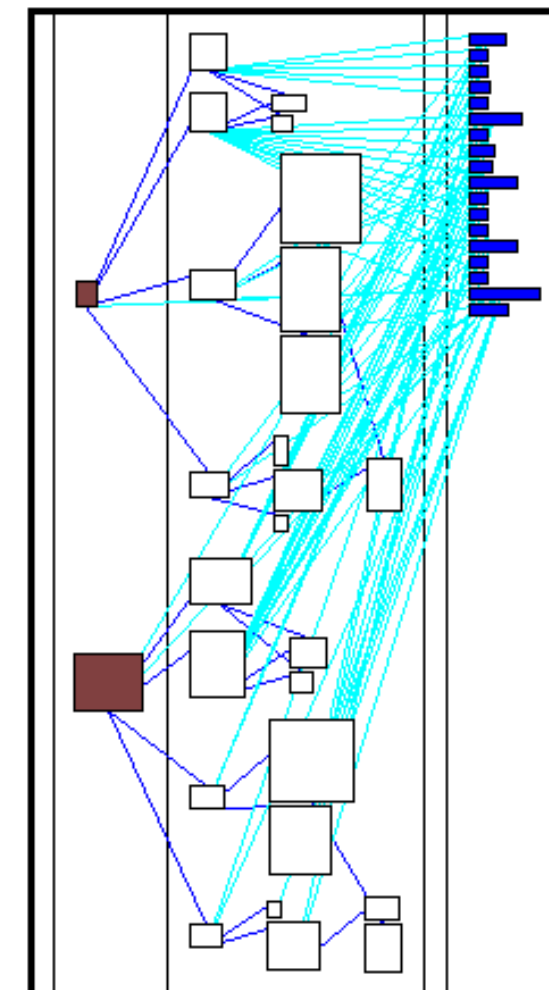
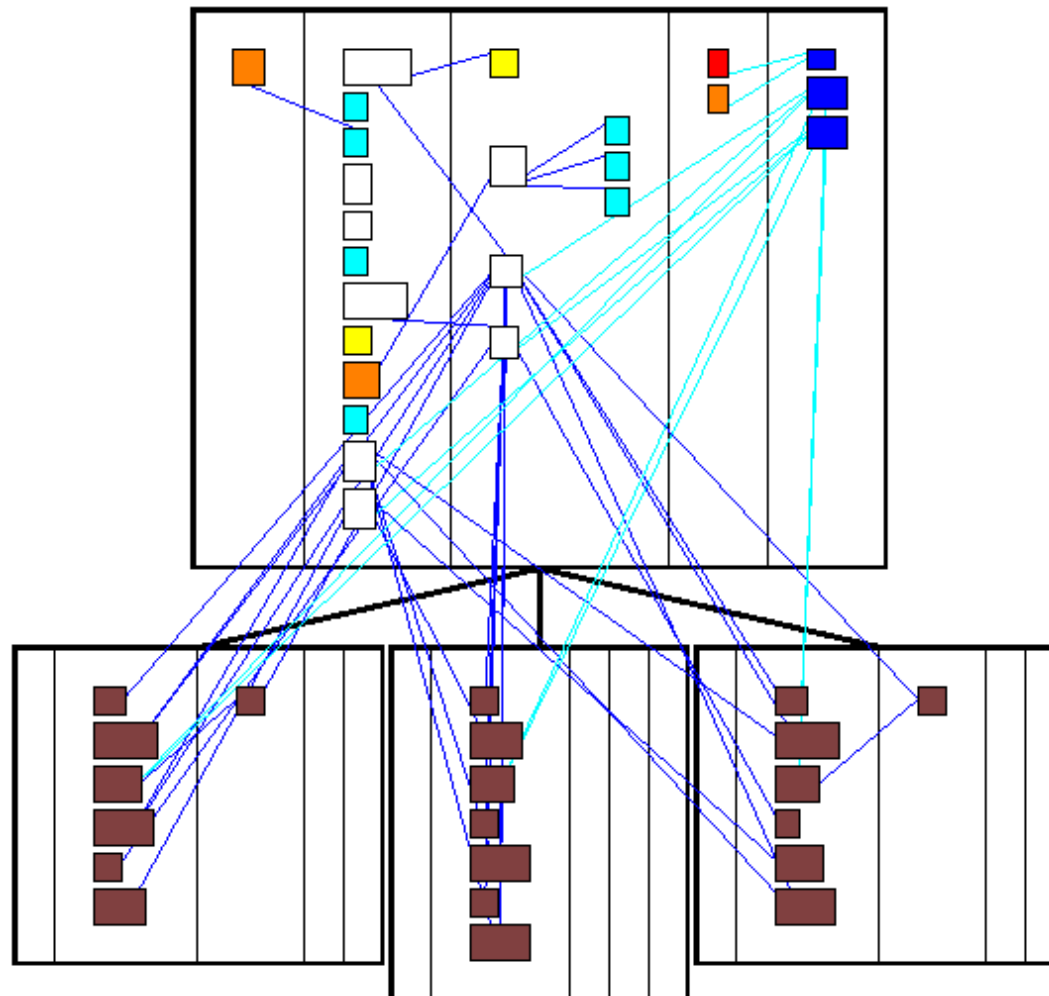


# Understanding a single class [PhD M. Lanza]

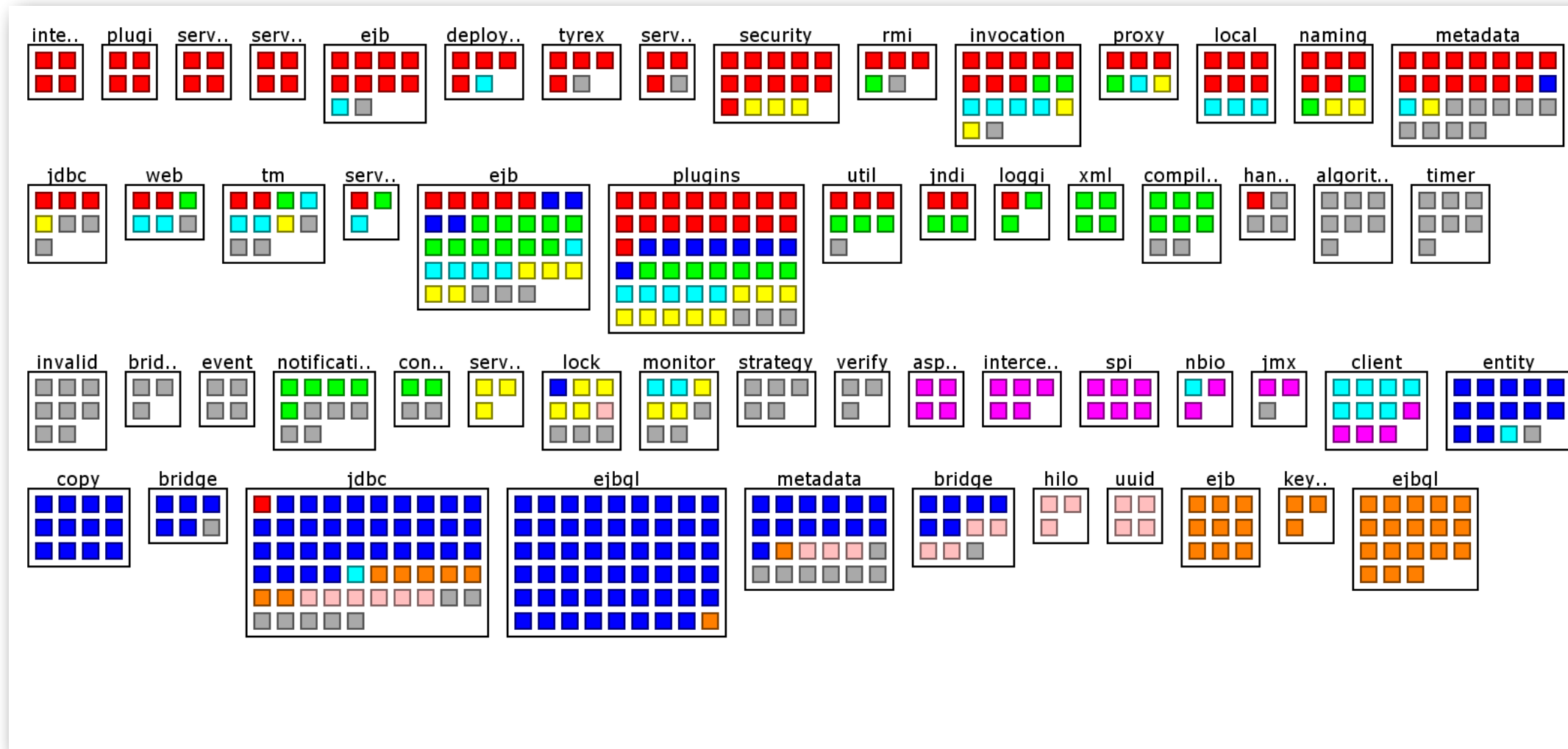




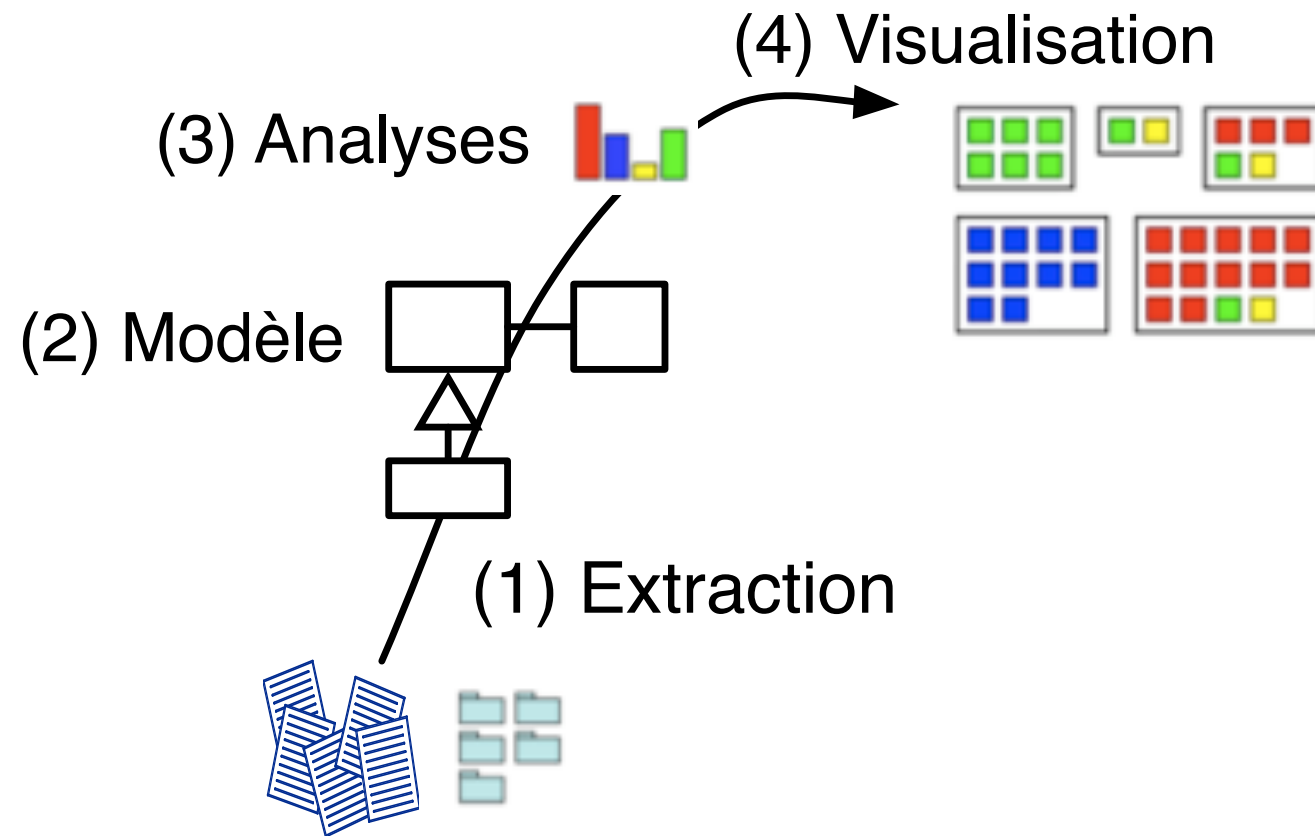
# Understanding classes [PhD M. Lanza]



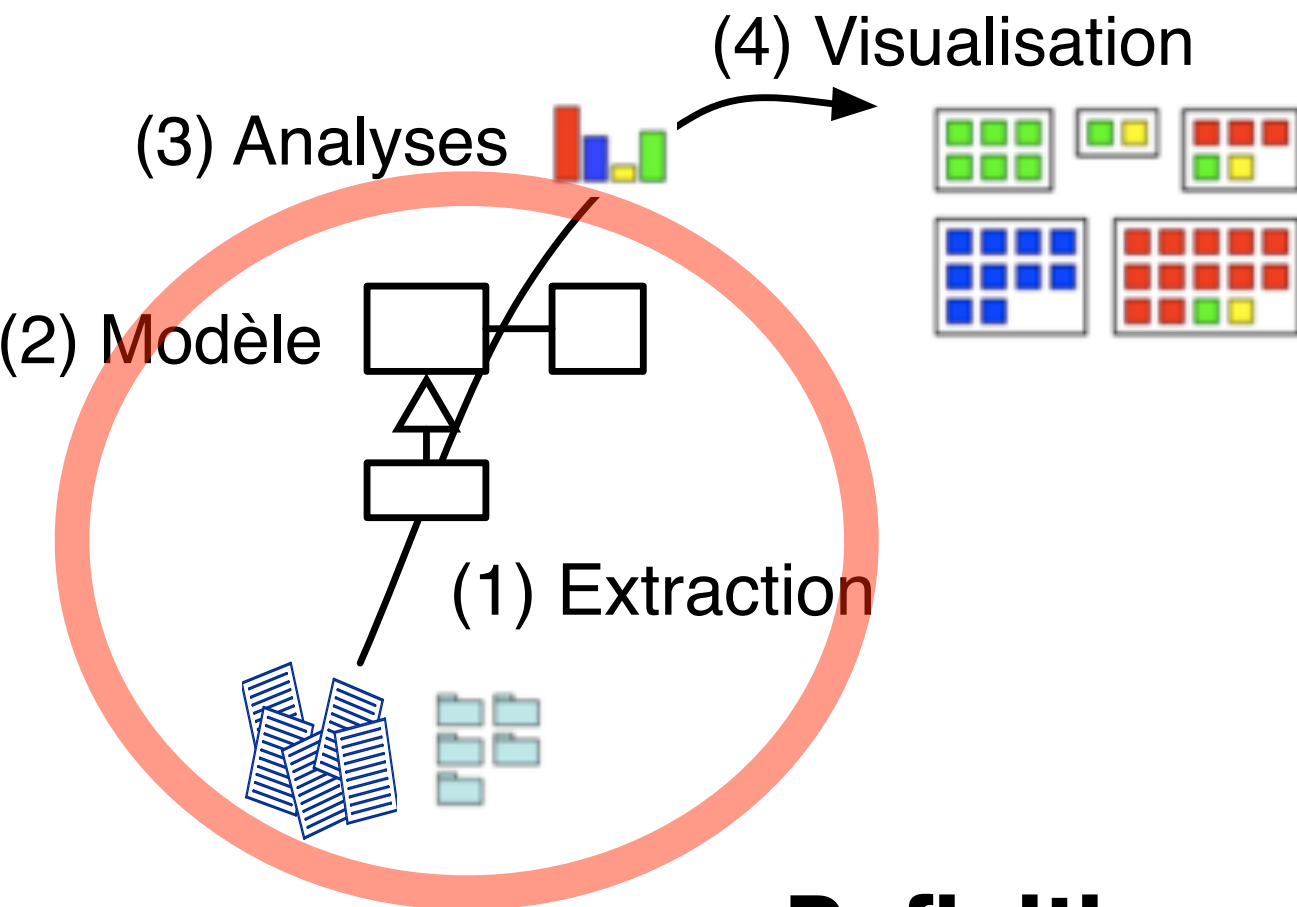
# How a property spread on a system?



# Example : Who is behind package X ?

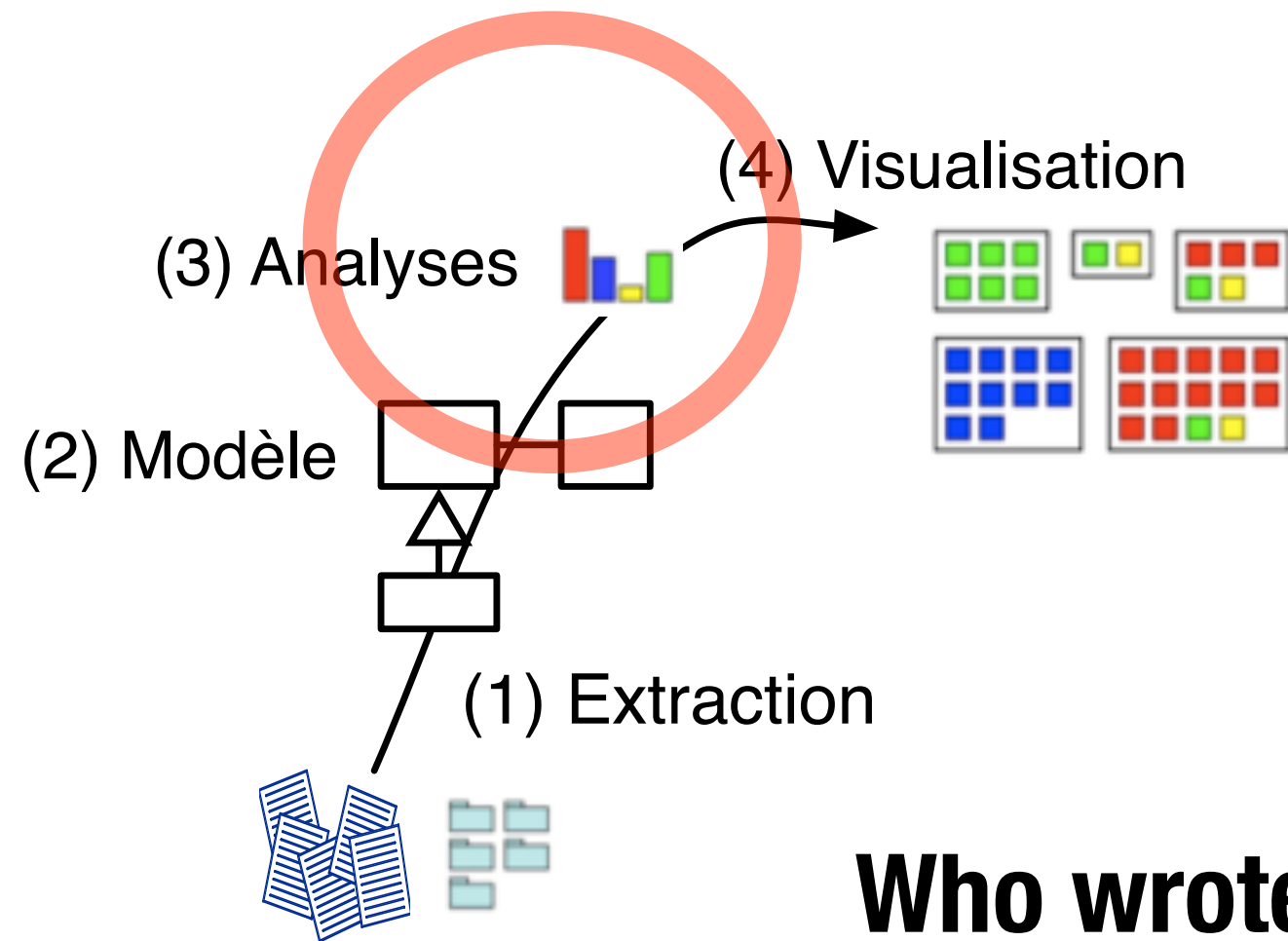


# Step 1 - Model Creation/Import



**Definition of a model to represent entities**  
**Data Extraction (CVS...)**

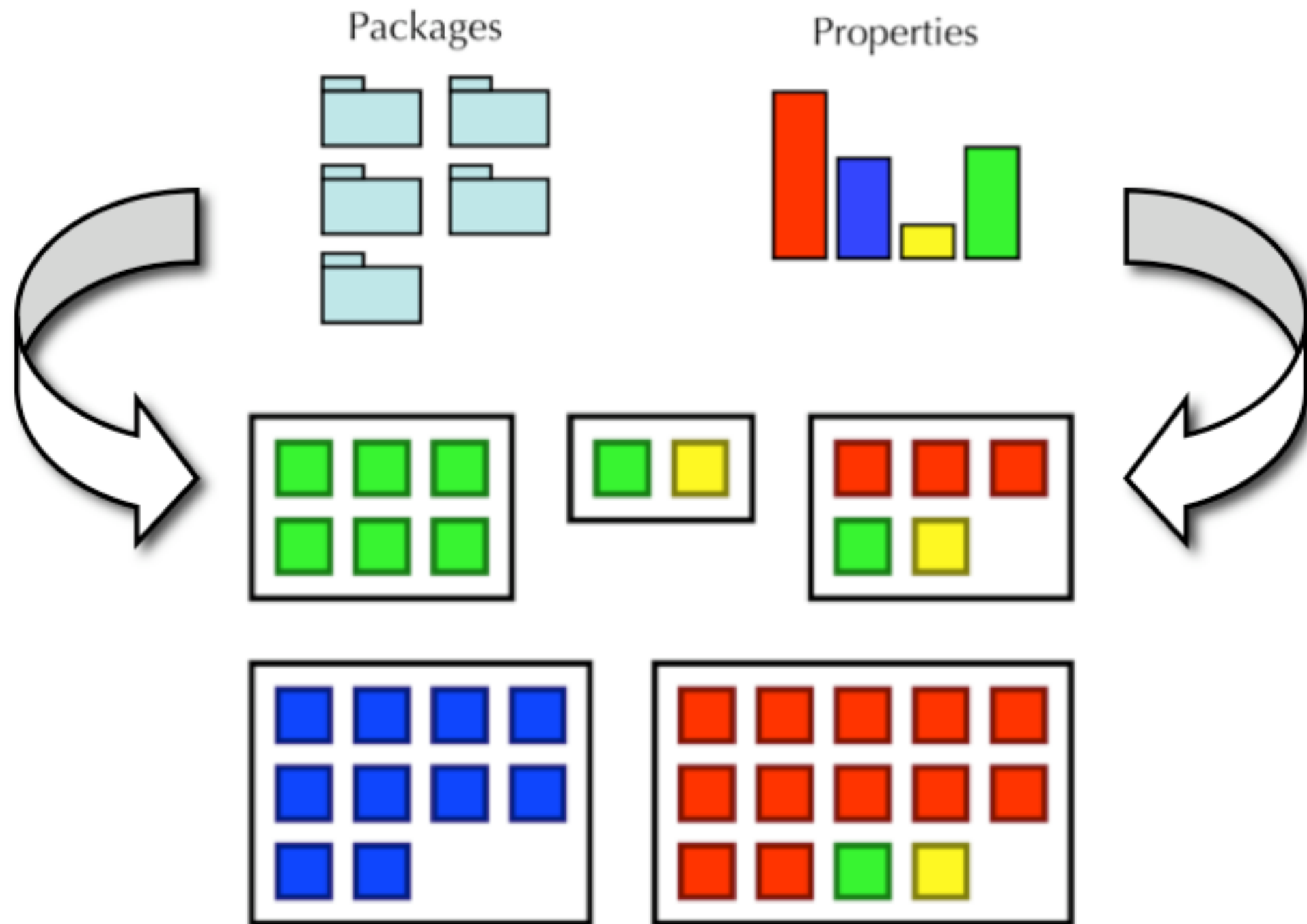
# Step 2 - Analyses



**Who wrote how many lines of code?**



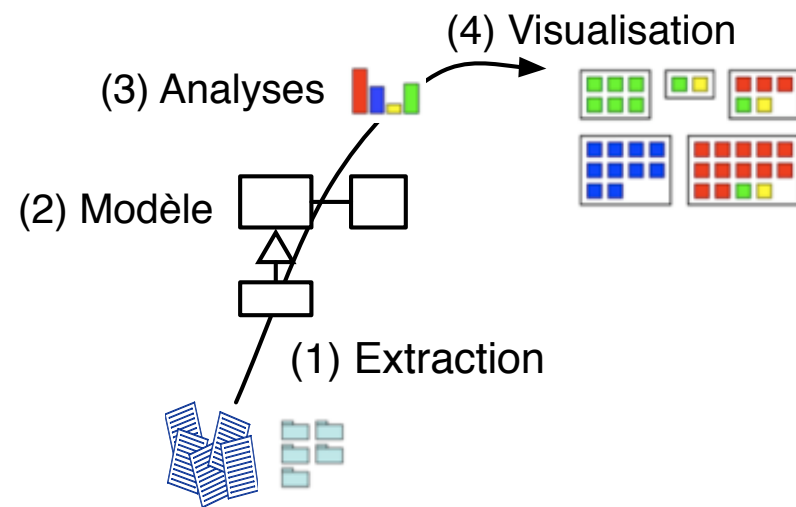
# Step : 3 - Creating the Map



# JBoss at a glance

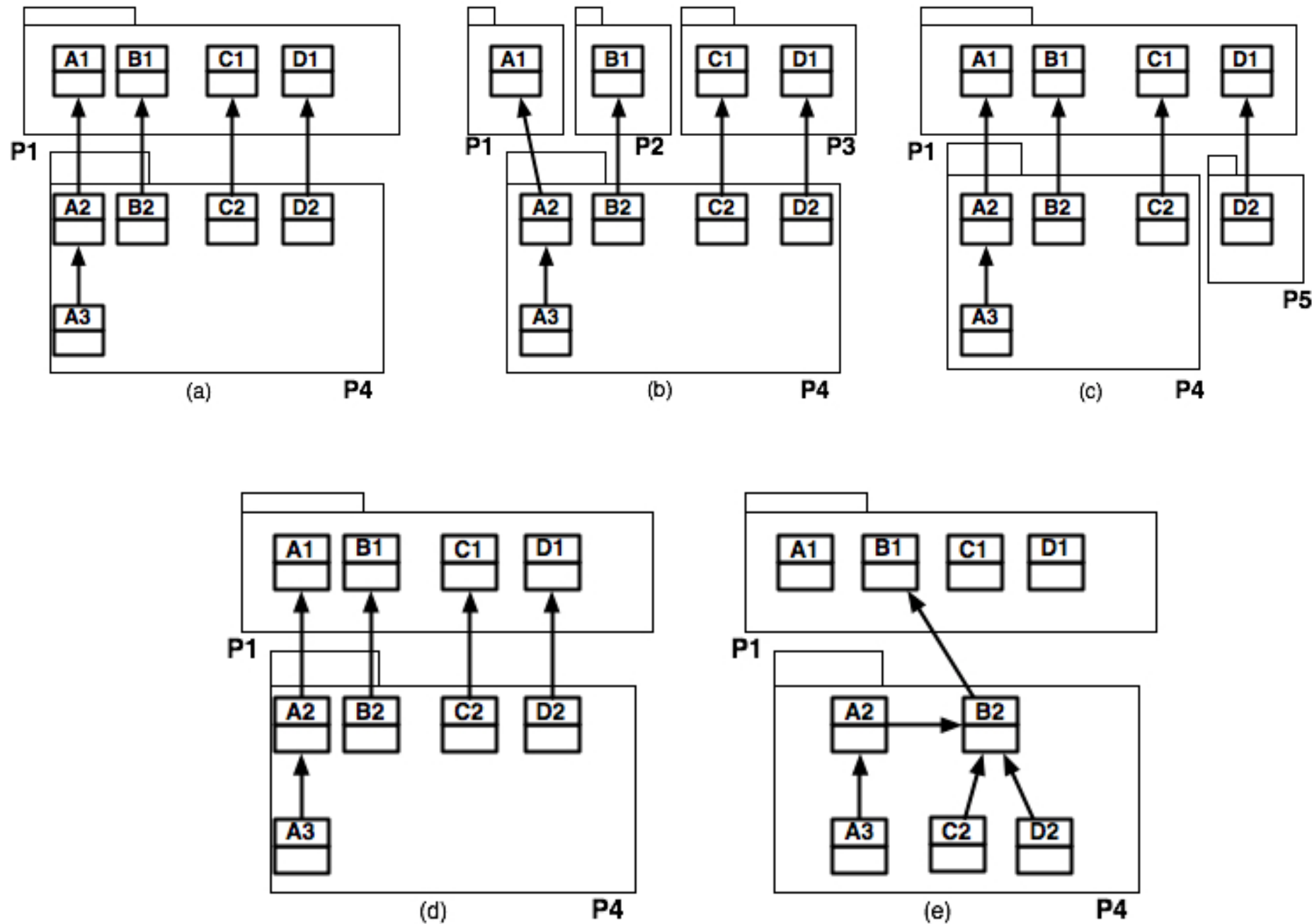
Interactive tool

Data in perspective



# How to support remodularisation?

[PhD H. Abdeen]



# How to support remodularisation?

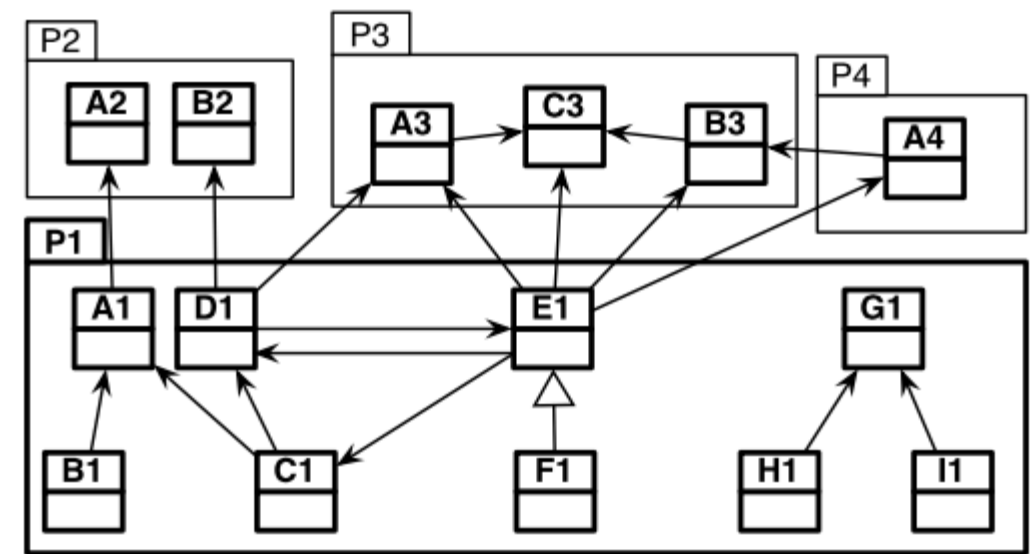
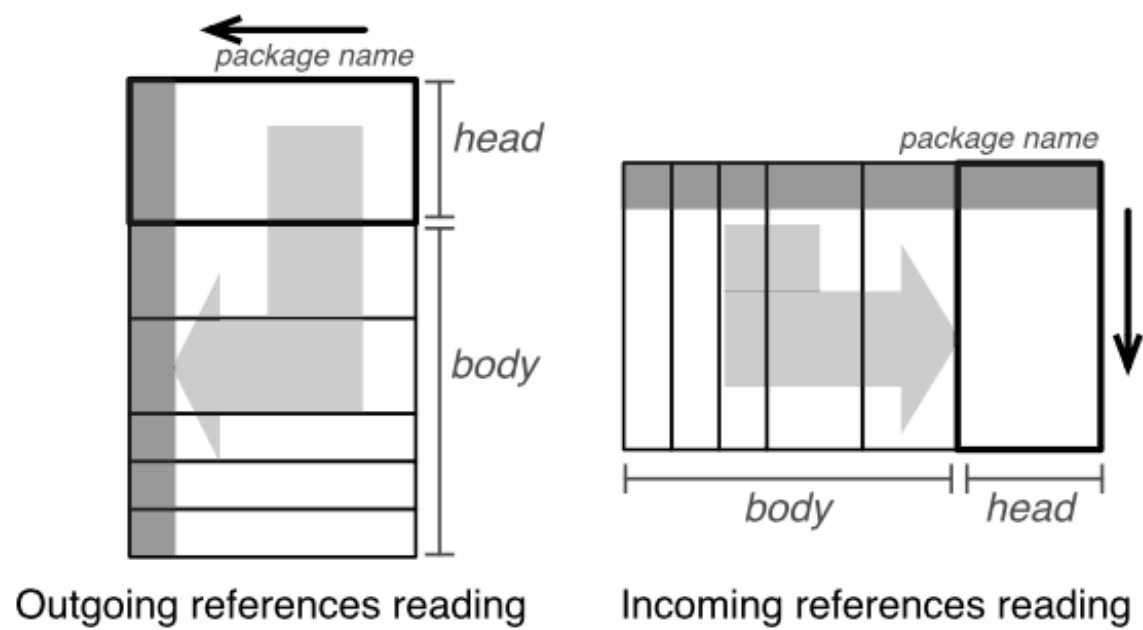
## [PhD Hani Abdeen]

How to understand the fan-in/fan-out of package?

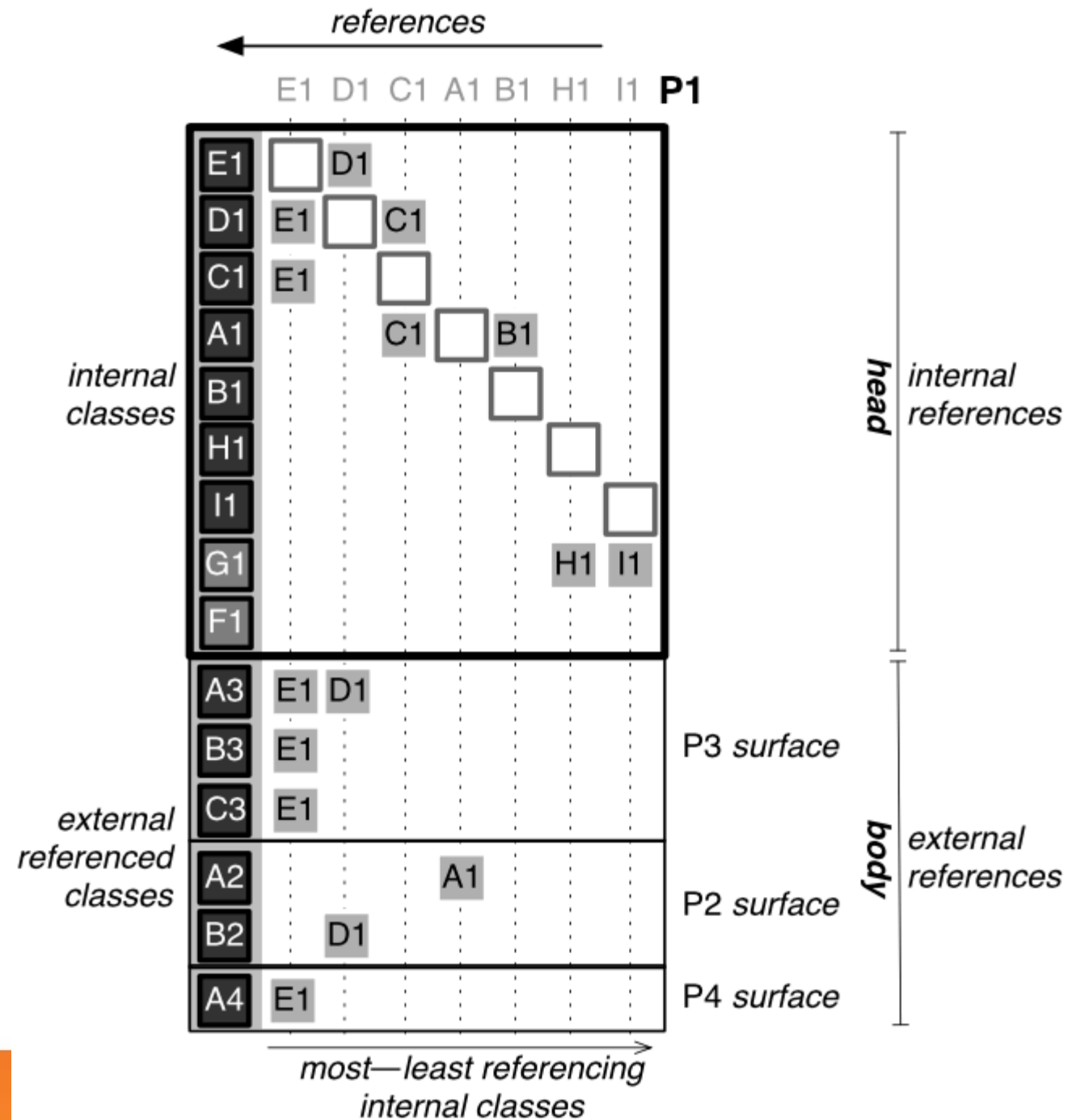
How to help remodularizers?

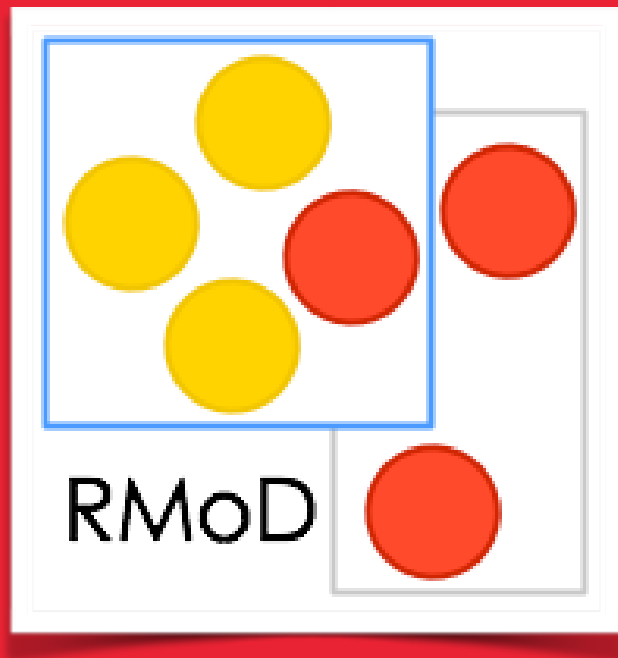
- Is this package cohesive?
- What is the bandwidth with others?
- What is the most used/used by class?
- What is the most important (internal/external view)?





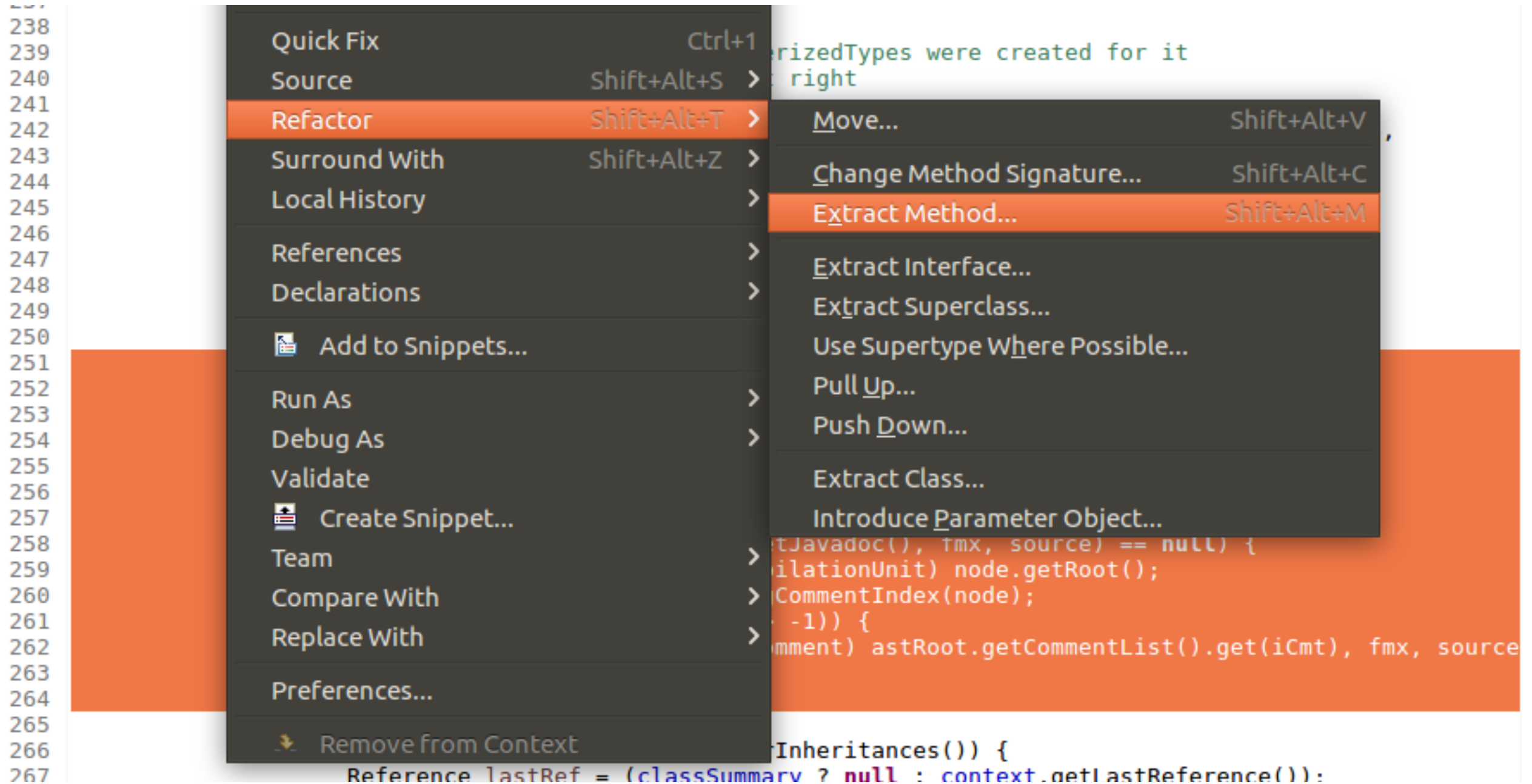
# Package blueprints [ICSME 07]





# Software Evolution in the Large

# Evolution in the small



# Evolution in the small

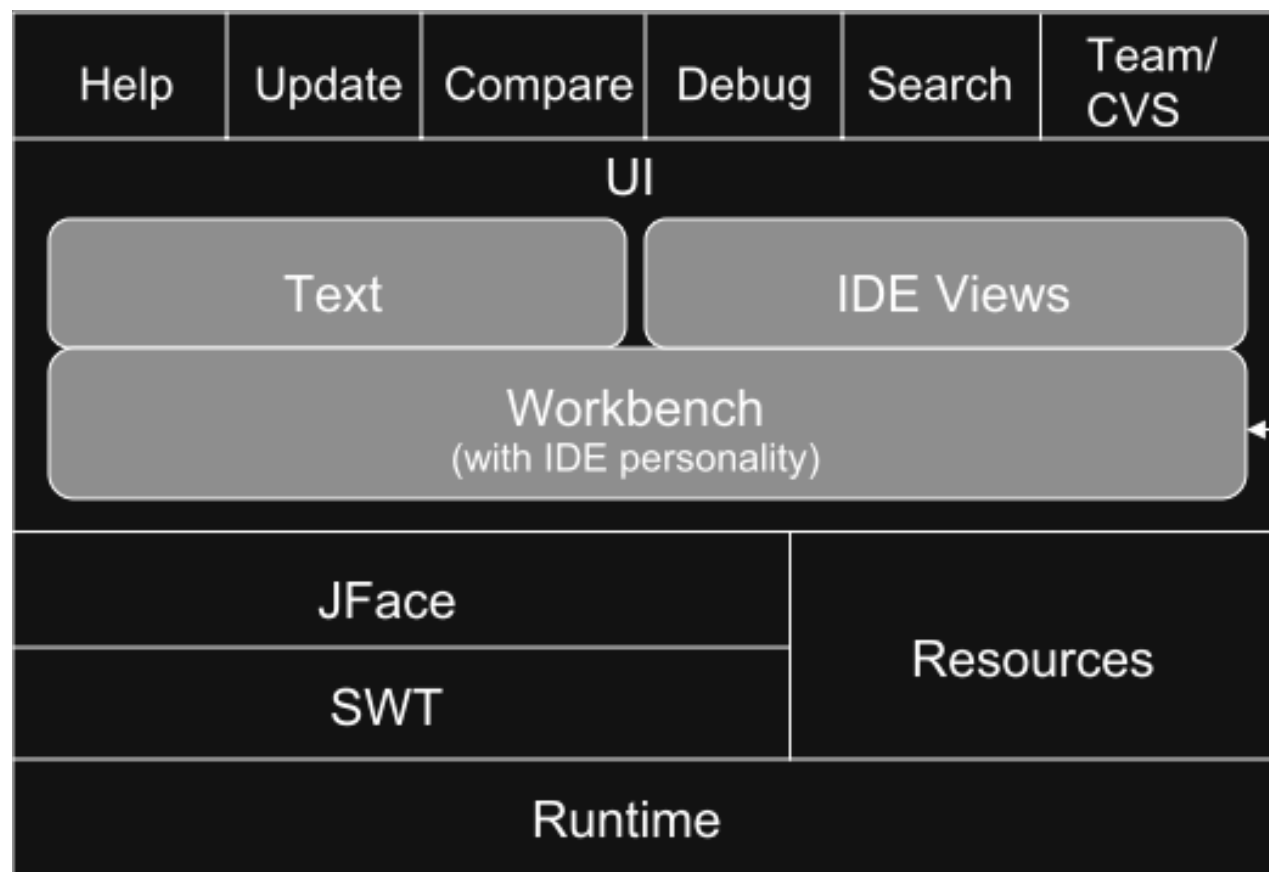
Modification (refactoring)

- On 1 entity
- Small number of parameters
- Well defined behavior
- “Preserve code semantic”
- Generic (constrained only by the programming language)

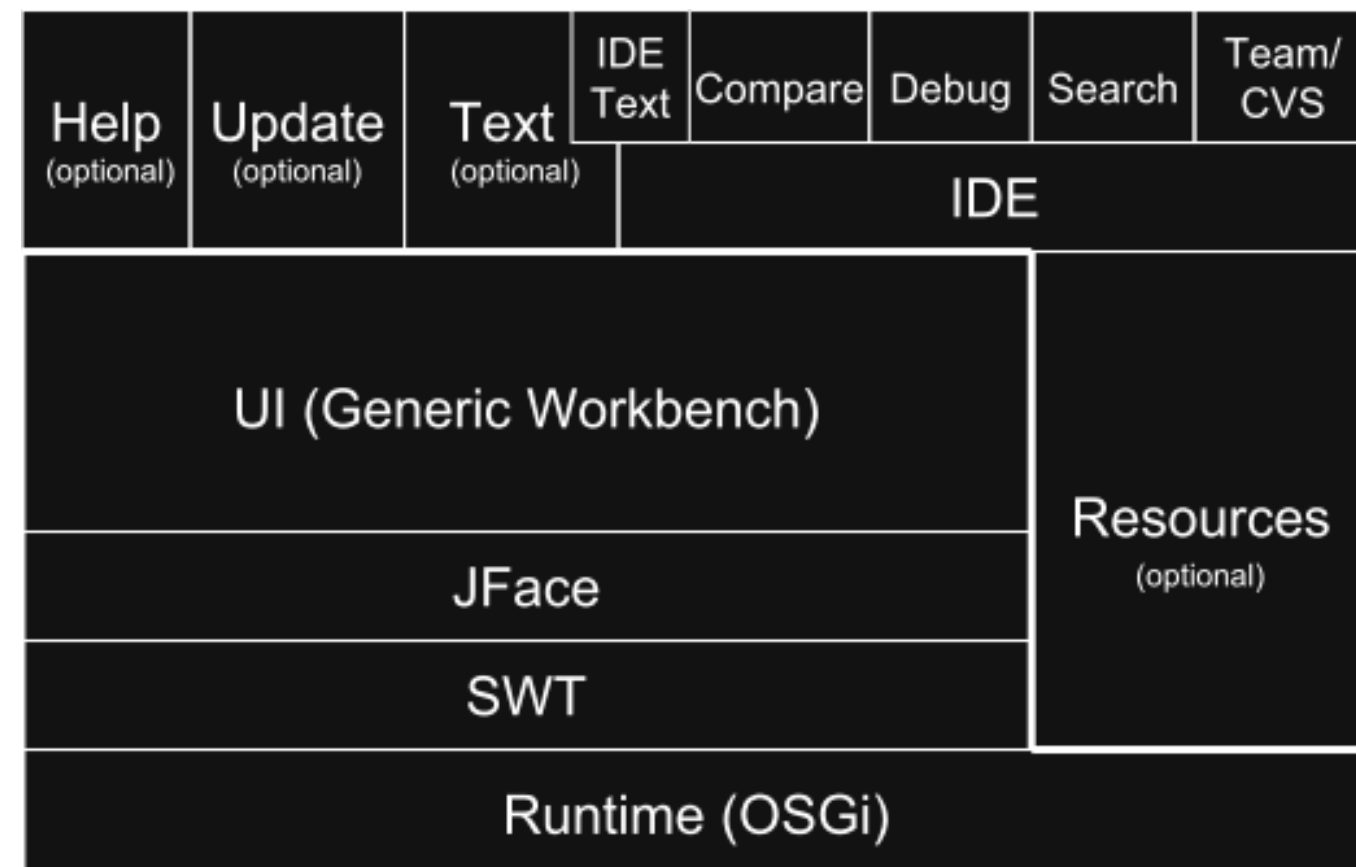


# Evolution in the LARGE

- Eclipse v2.1 → v3.0



v2.1 (Extensible IDE)



v3.0 (Rich Client Platform)

# Evolution in the LARGE

Restructure architecture

Large refactor

Break a big class

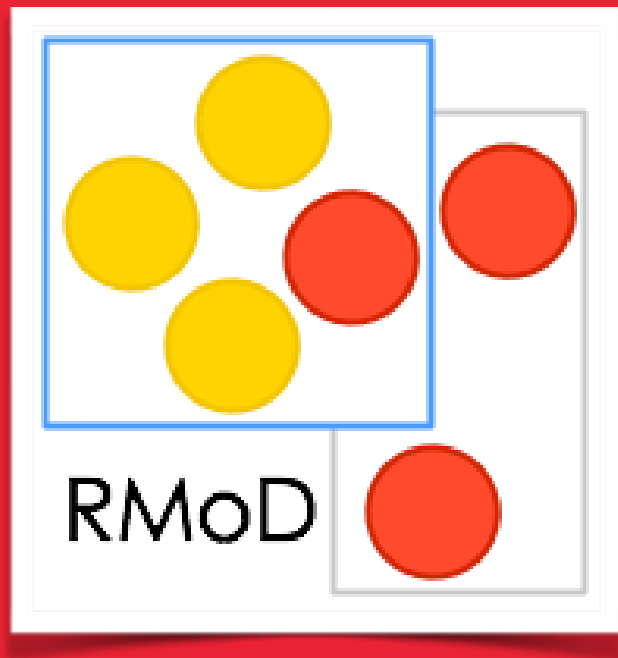
Introduce a design pattern (e.g. MVC,  
Hybernate)

Migrate to a new library version

...

# Evolution in the LARGE

- Evolution
  - On several entities
  - Complex behavior
  - Specific to the domain, system, and task
  - May break code semantic temporarily



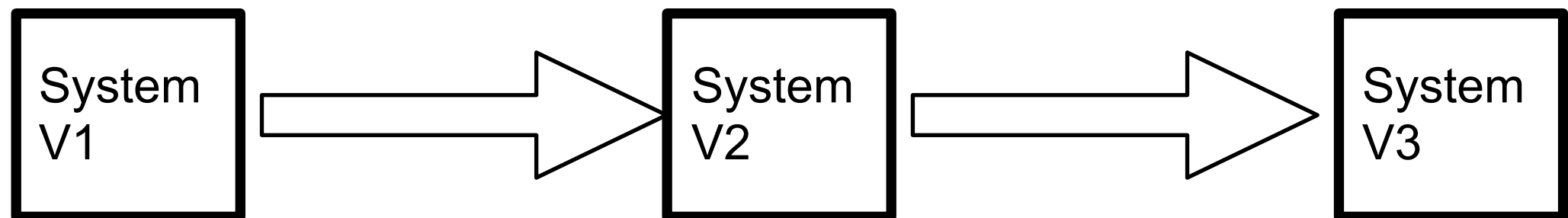
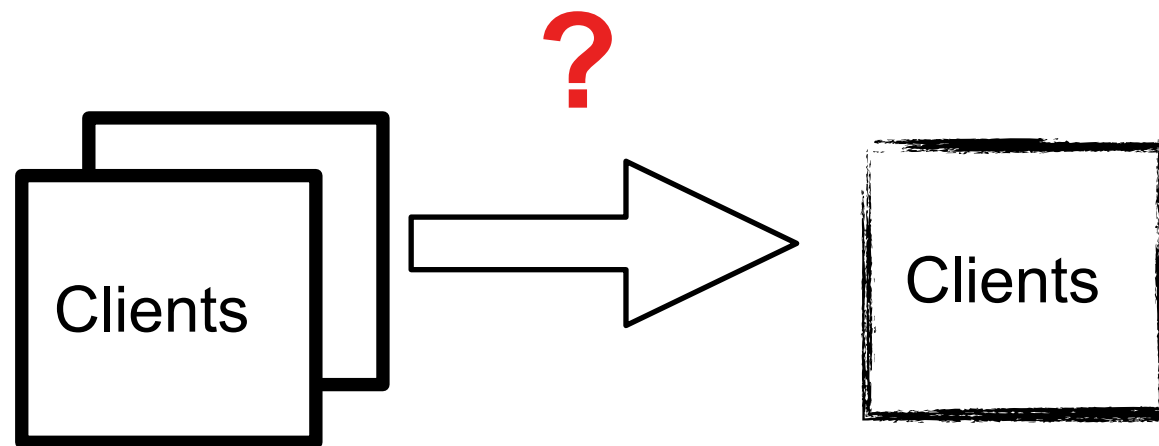
# Evolution in the large: Automated rules to support migration [PhD A. Hora]



# Mining API Change Rules

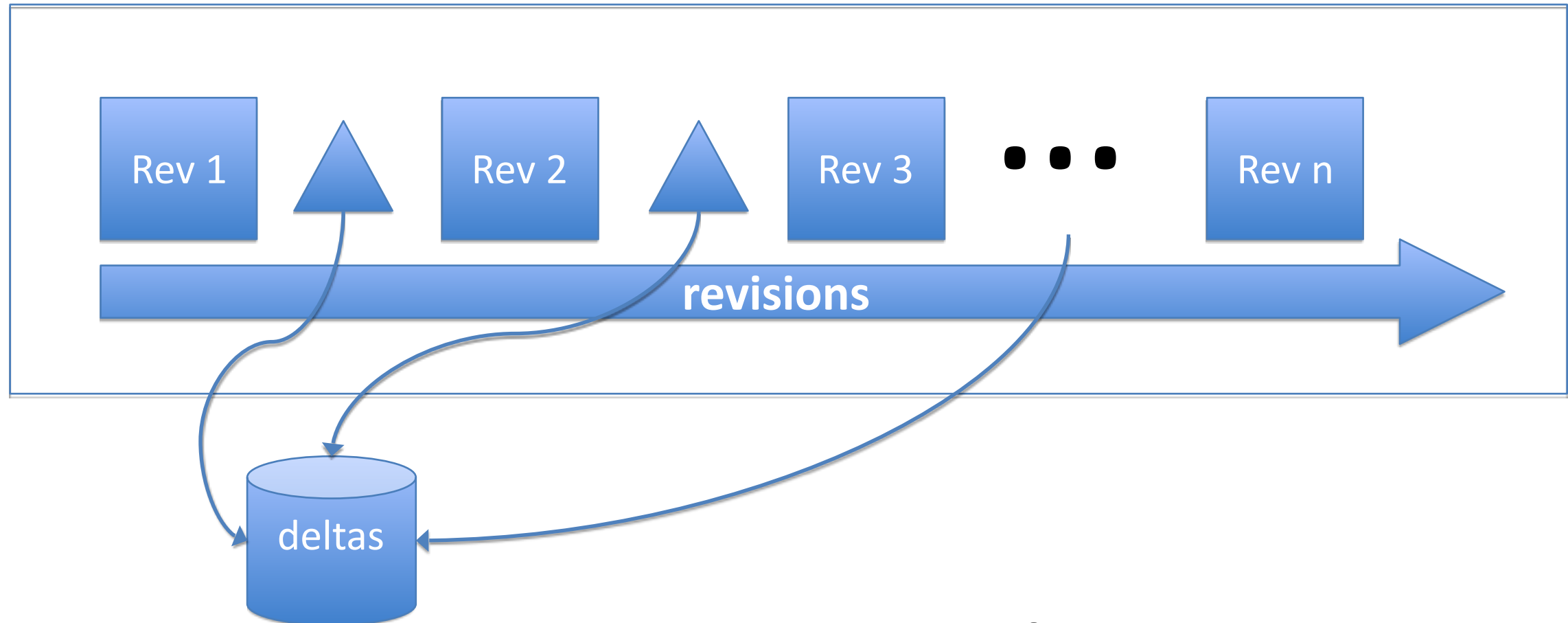
- In Eclipse, 42% of the methods in version 1 were not in v.2
- In Pharo, a simple API change affected thousands of clients
- Clients should not but *do* call internal API

# How to help migrate from version to version?

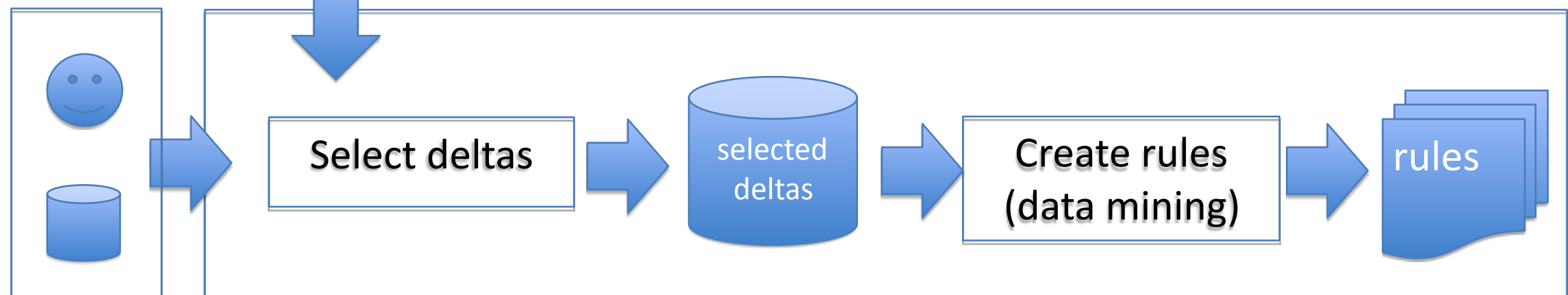


# Mining API Change Rules

## 1. Extracting deltas



## 2. Discovering rules



# Mining API Change Rules

## Format of the changes



deleted-invoc(context-id, signature)

added-invoc(context-id, signature)

## Diff of method foo() between version 1 and 2

```
- self.add(MooseModel.root().add(model));  
+ self.add(model.install());
```

## Formatted changes

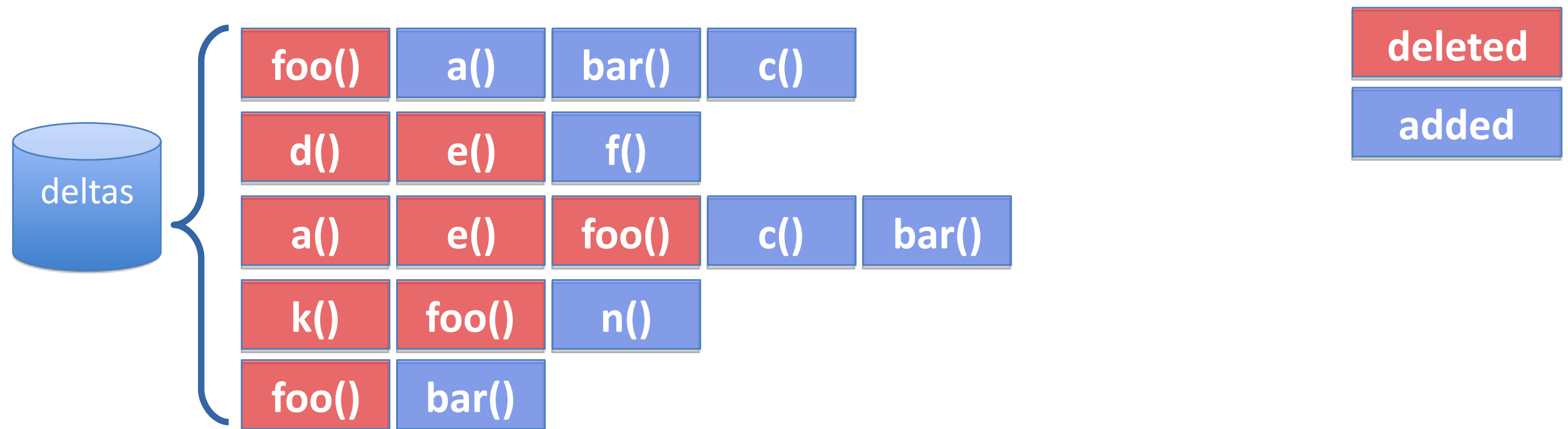
deleted-call("foo()-rev2", "MooseModel.root()")

deleted-call("foo()-rev2", "MooseModel.add(MooseModel)")

added-call("foo()-rev2", "MooseModel.install()")

# Mining API Change Rules

Request: foo()

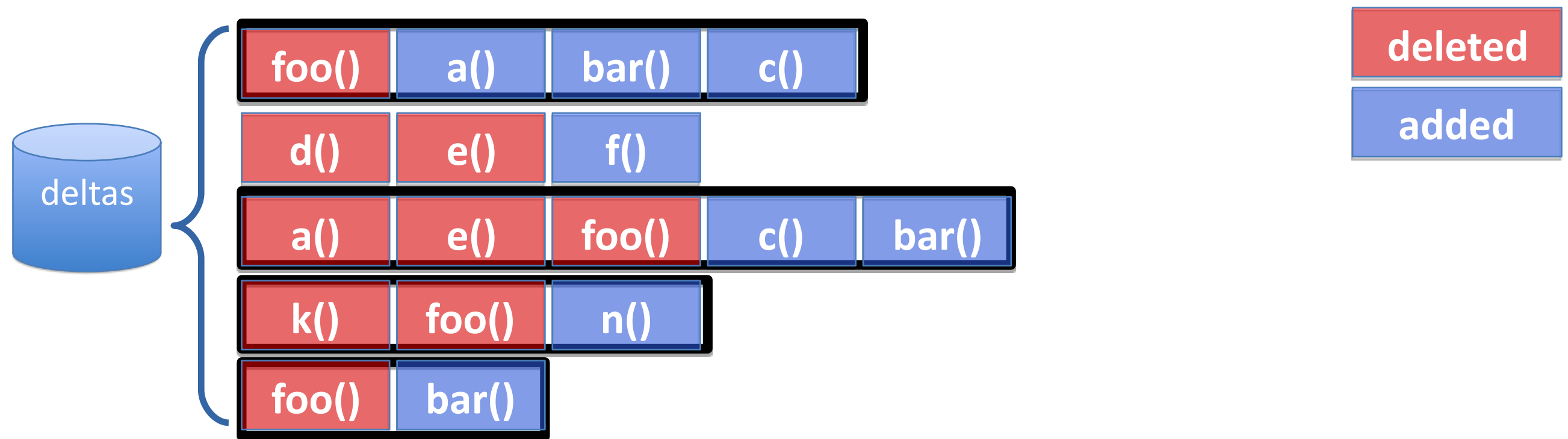




# Mining API Change Rules

Request: foo()

1<sup>st</sup> step: selecting deltas

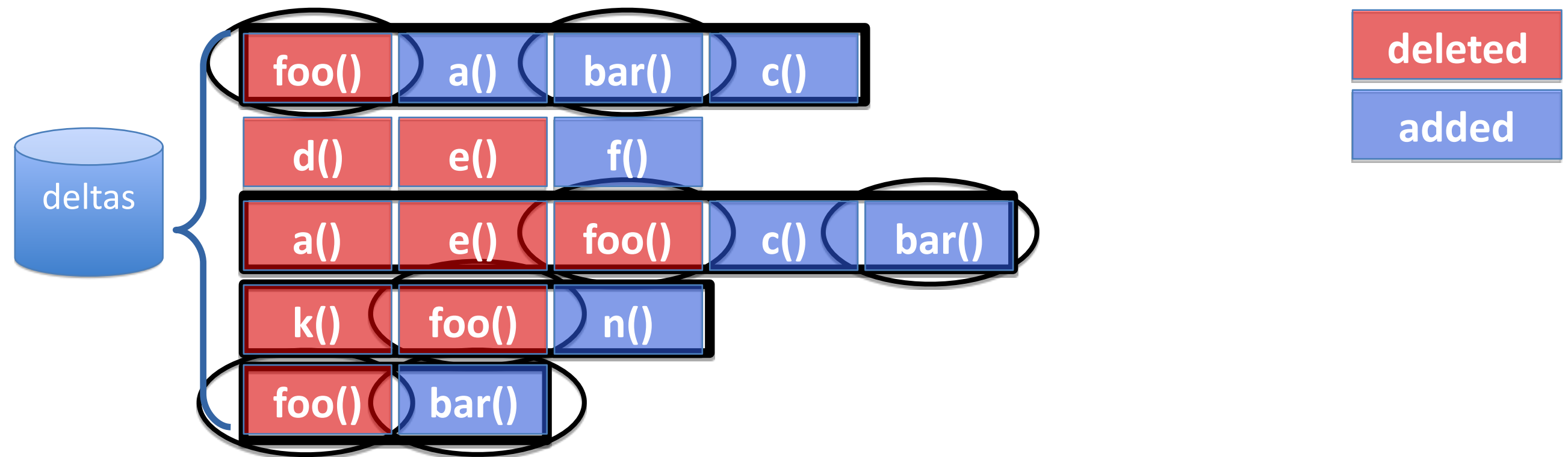


# Mining API Change Rules

Rule: foo() → bar() Confidence = 75%

1<sup>st</sup> step: selecting deltas

2<sup>nd</sup> step: discovering rules



# Mining API Change Rules

`isNil().ifTrue(*) → ifNil(*)`

`keys().do(*) → keysDo(*)`

`intersect(*) → intersectIfNone(*,*)`

`Scanner.new().scanTokens(*) → parseAsLiteralToken()`

`RegisterAsApplication(*) →`

`WAAdmin.registerAsApplicationAt(*,*)`

`Character.cr() → ROPlatform.current().newLine()`

# Rule Validation

Are the generated rules valid to experts?

Yes between 46% to 86%

	Rules	Valid	Invalid	Don't know	Precision
Pharo2	62	17	20	25	<b>46%</b>
Pharo3	95	68	24	3	74%
Moose	50	43	7	0	<b>86%</b>
Glamour	23	15	8	0	65%
Roassal	36	28	8	0	78%
Seaside	76	61	15	-	80%

# Tool Support

**1. Input pane**

- On request rules
- List of rules
- Min support: 0.5
- largeDeltaStrategy
- allTimeStrategy
- Request: ClassOrganizer default

**2. Association rule pane**

- Association Rules
- Association Rules (1)
- Deleted: d ClassOrganizer default
- Added: a Protocol unclassified
- Conf: 1
- Supp: 19
- association rule
- confidence and support of the rule

**3. Delta pane**

- Default displaying: list of deltas
- Deltas (19)
- Map
- NautilusUI,implementSelector: small
- AbstractNautilusUI,asYetUnclassifiedString small
- MethodClassifier.classifyByOtherImplementors: small
- SmalltalkImage,fixUpProblemsWithAllCategory small
- CodeHolder,categoryFromUserWithPromptFor: medium
- AbstractTool,categoryMethodsOf:from: small
- SyntaxError,addClass:codeErrorLocation:debugger:doitFlag: small
- TClassAndTraitDescription,uncategorizedMethods small
- TPureBehavior,updateMethodDictionarySelector: small
- TClassAndTraitDescription,compile:notifying: small
- CodeHolder,categoryFromUserWithPromptFor: medium
- Browser,categoryAllUncategorizedMethods small
- AbstractTool class,protocolSuggestionsFor: medium
- Browser,addCategory medium

**4. Example helper pane**

- Delta description
- Deleted call
- Added call
- Older revision
- Newer revision

```

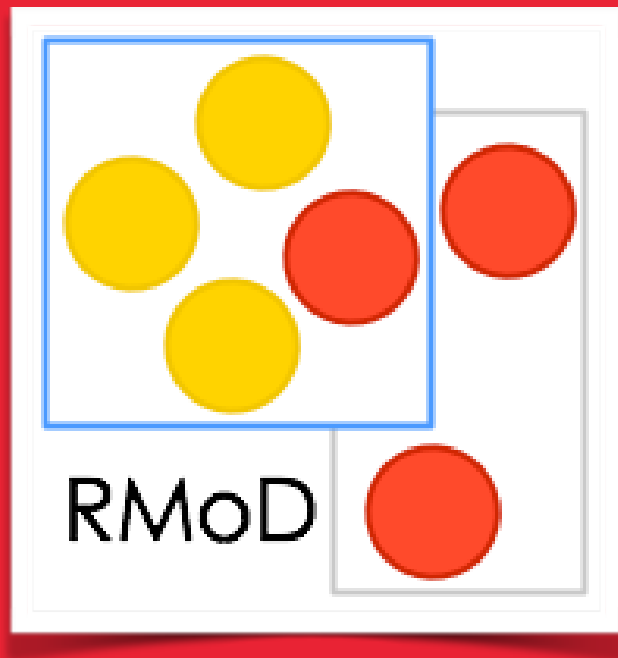
Older revision:
categorizeMethods: aCollection of: aClass from: aCategory
  "Present a choice of categories or prompt for a new category name and add it before the current"

  [ labels reject lines oldIndex newName ]
  aCollection isEmpty: [ ^self ].
  aClass ifNil: [ ^self ].
  labels := OrderedCollection new.
  labels
    addAll: aClass organization categories copy sort;
    add: ClassOrganizer default.
  lines := OrderedCollection new.
  lines add: labels size - 1.
  newName := UIManager default chooseOrRequestFrom: labels lines: lines title: ('Change Protocol
  newName ifNil: [ ^self ].
  newName := newName asSymbol.
  aCollection do: [ item ]
    item methodClass organization
      classify: item selector
      under: newName
      suppressIfDefault: true ].

Newer revision:
categorizeMethods: aCollection of: aClass from: aCategory
  "Present a choice of categories or prompt for a new category name and add it before the current"

  [ labels reject lines oldIndex newName ]
  aCollection isEmpty: [ ^self ].
  aClass ifNil: [ ^self ].
  labels := OrderedCollection new.
  labels
    addAll: aClass organization categories copy sort;
    add: Protocol unclassified.
  lines := OrderedCollection new.
  lines add: labels size - 1.
  newName := UIManager default chooseOrRequestFrom: labels lines: lines title: ('Change Protocol
  newName ifNil: [ ^self ].
  newName := newName asSymbol.
  aCollection do: [ item ]
    item methodClass organization
      classify: item selector
      under: newName
      suppressIfDefault: true ].
  
```





# Tailoring runtimes

[Ph.D. G. Polito ]

# Application extraction example

```
MainApp>>start
  logger := StdoutLogger new.
  logger log: 'Application has started'.
  "do something"
  logger log: 'Application has finished'.
```

```
StdoutLogger>>newLine
  stdout newLine.
```

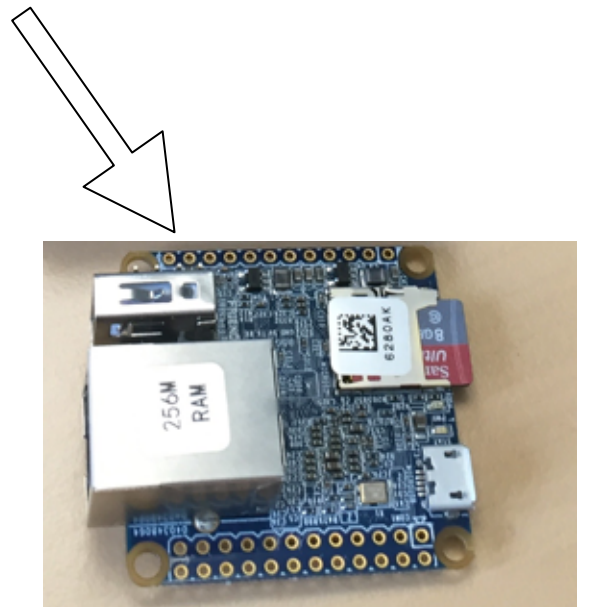
```
StdoutLogger>>log: aMessage
  stdout nextPutAll: Time now printString.
  stdout nextPutAll: aMessage.
  stdout newLine.
```

```
RemoteLogger>>log: aMessage
  | socket |
  socket := self newSocket.
  socket nextPutAll: Time now printString.
  socket nextPutAll: aMessage.
  socket newLine.
```

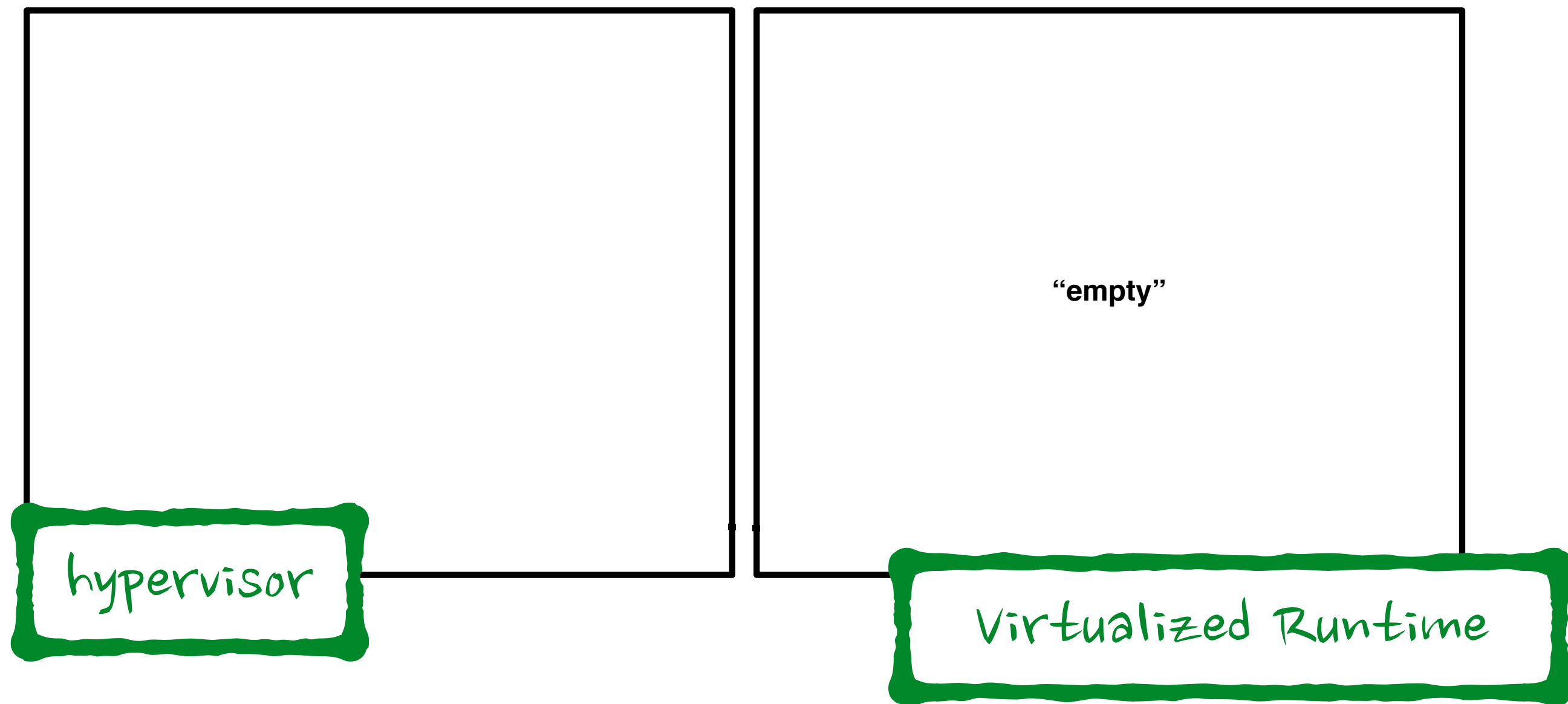
```
RemoteLogger>>newSocket
  "...."
  "creates an instance of socket given some configuration"
```

**Unused  
Code**

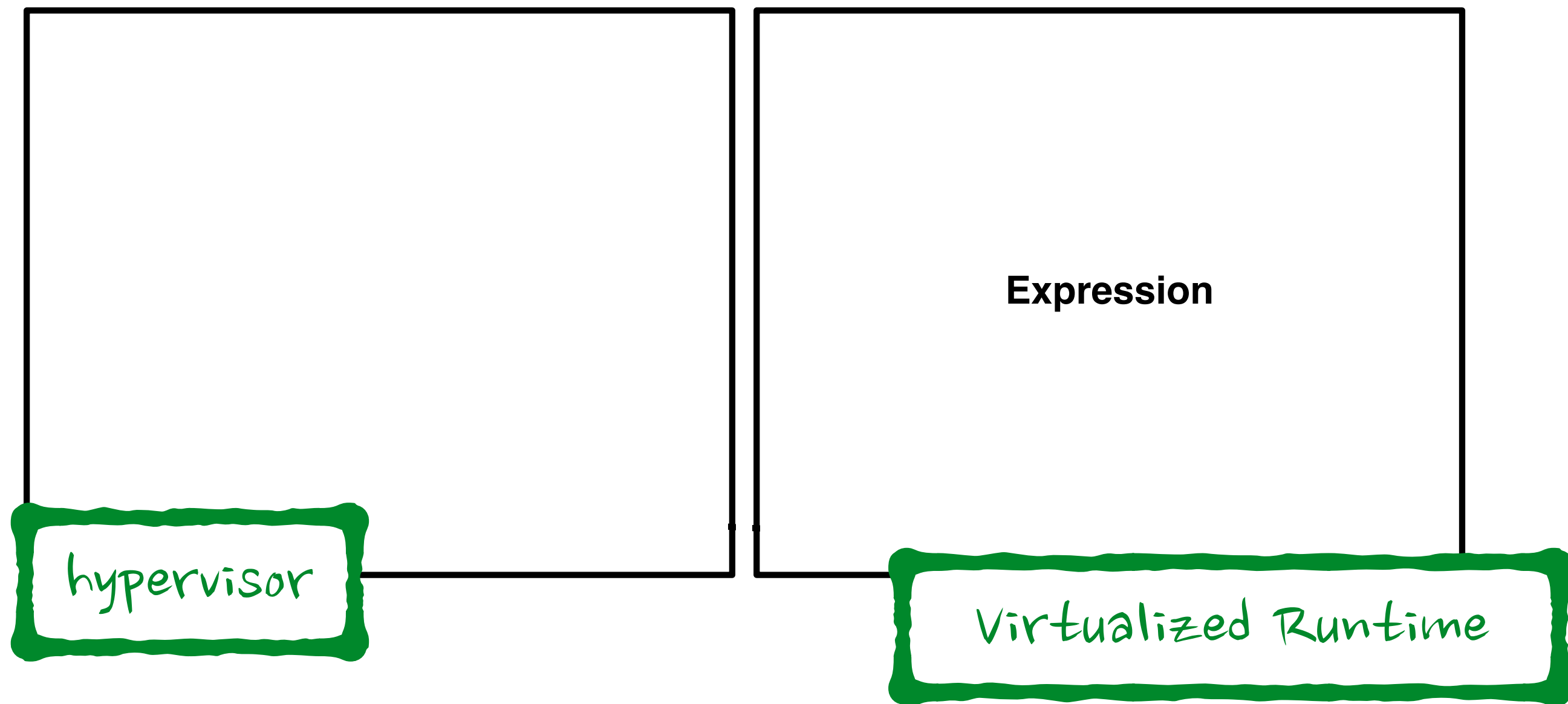
Specialised runtime  
Minimal applications



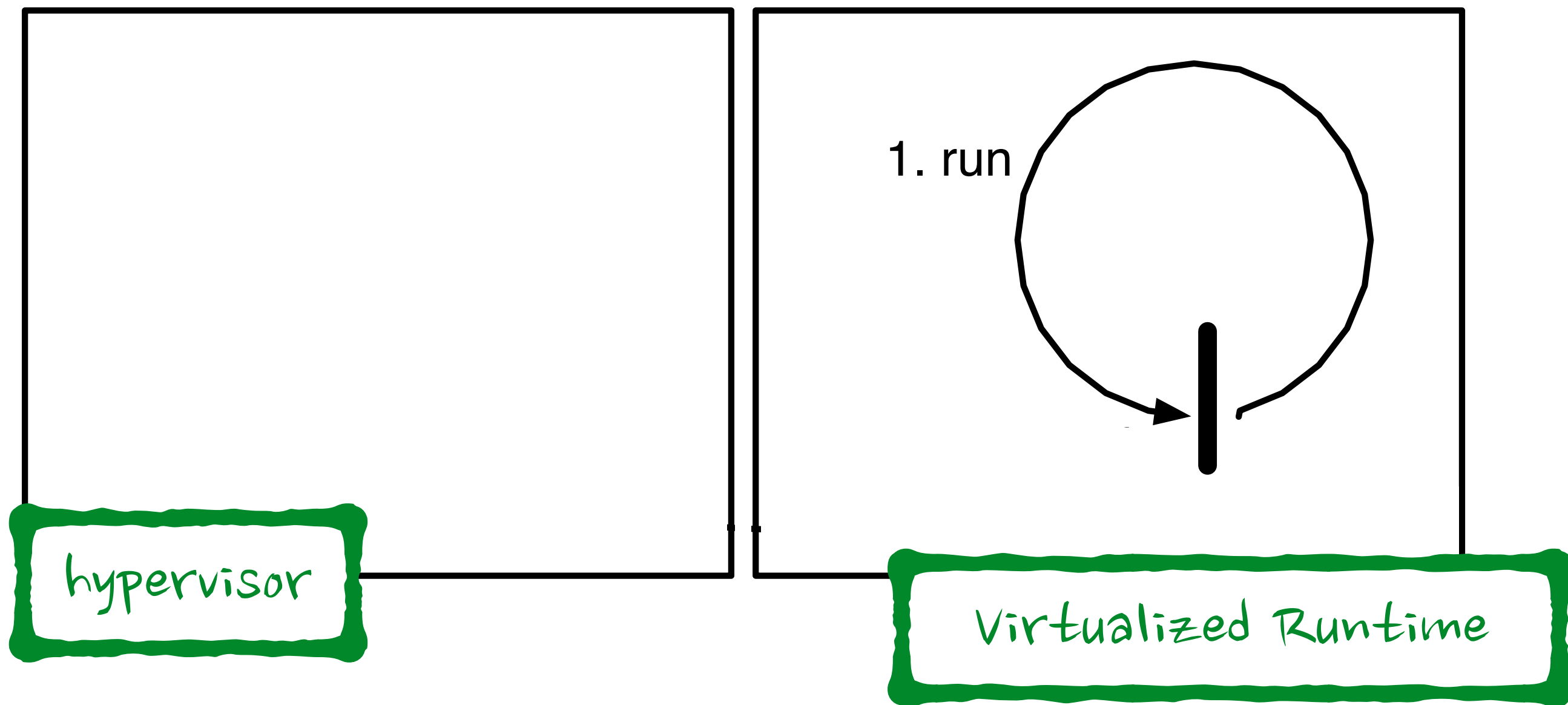
# Our approach: Run-Fail-Grow



# Our approach: Run-Fail-Grow

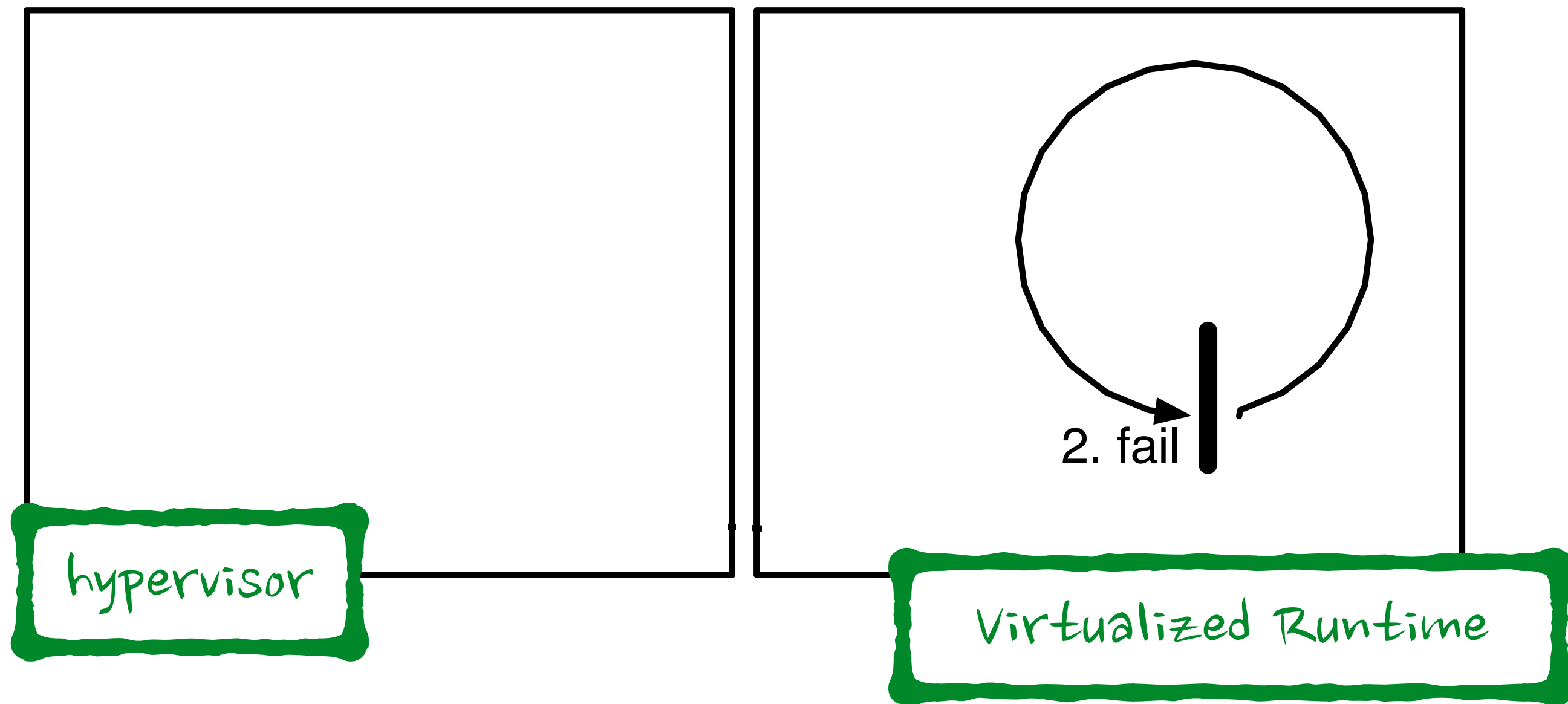


# Run

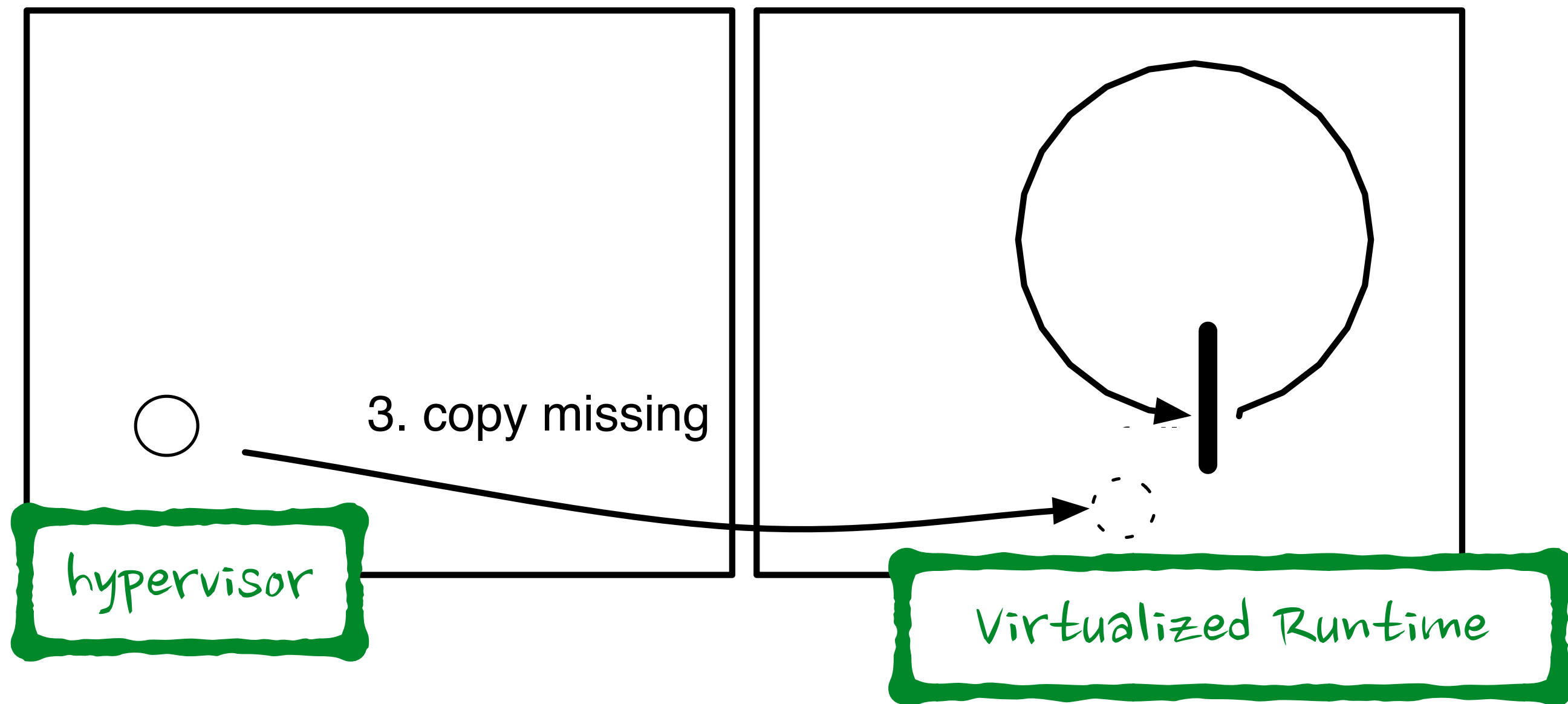




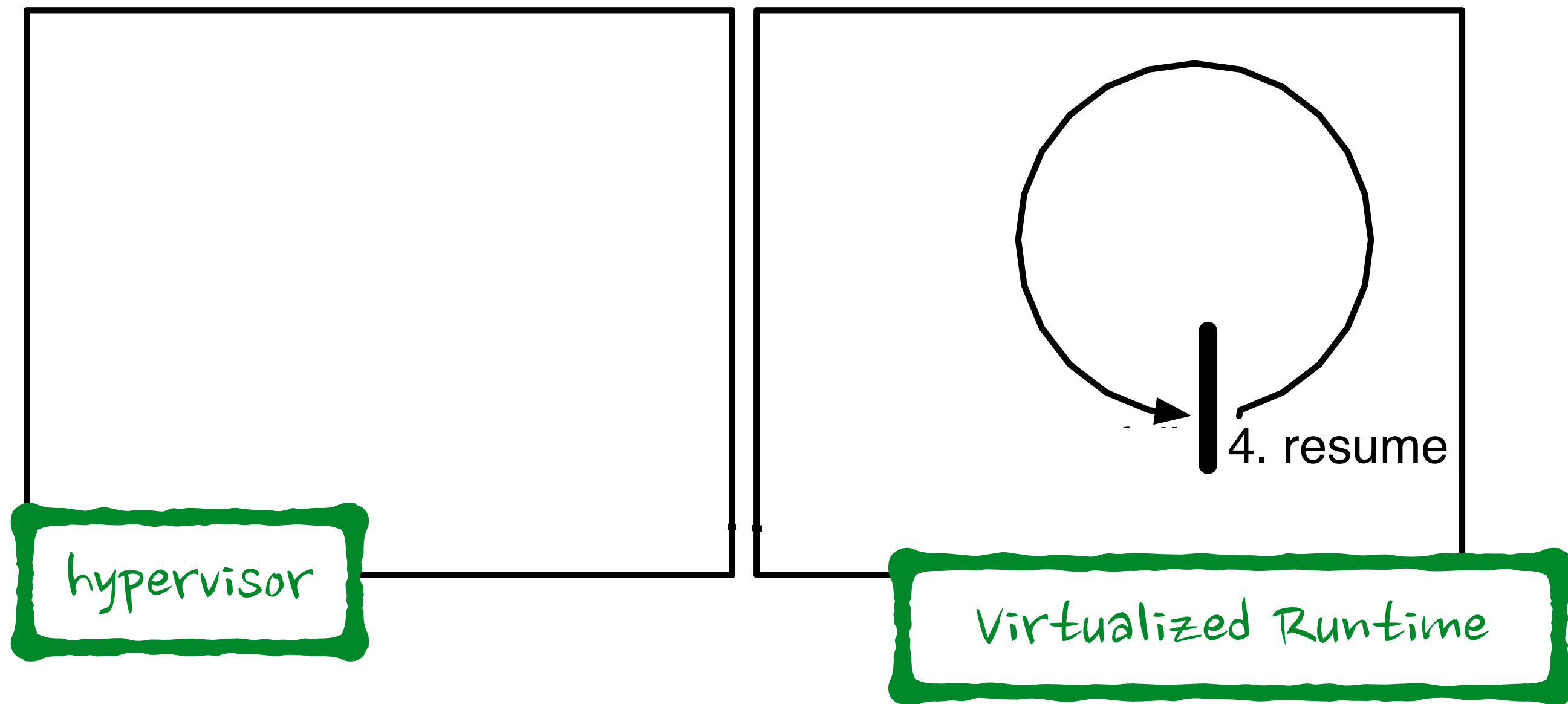
# Run-Fail



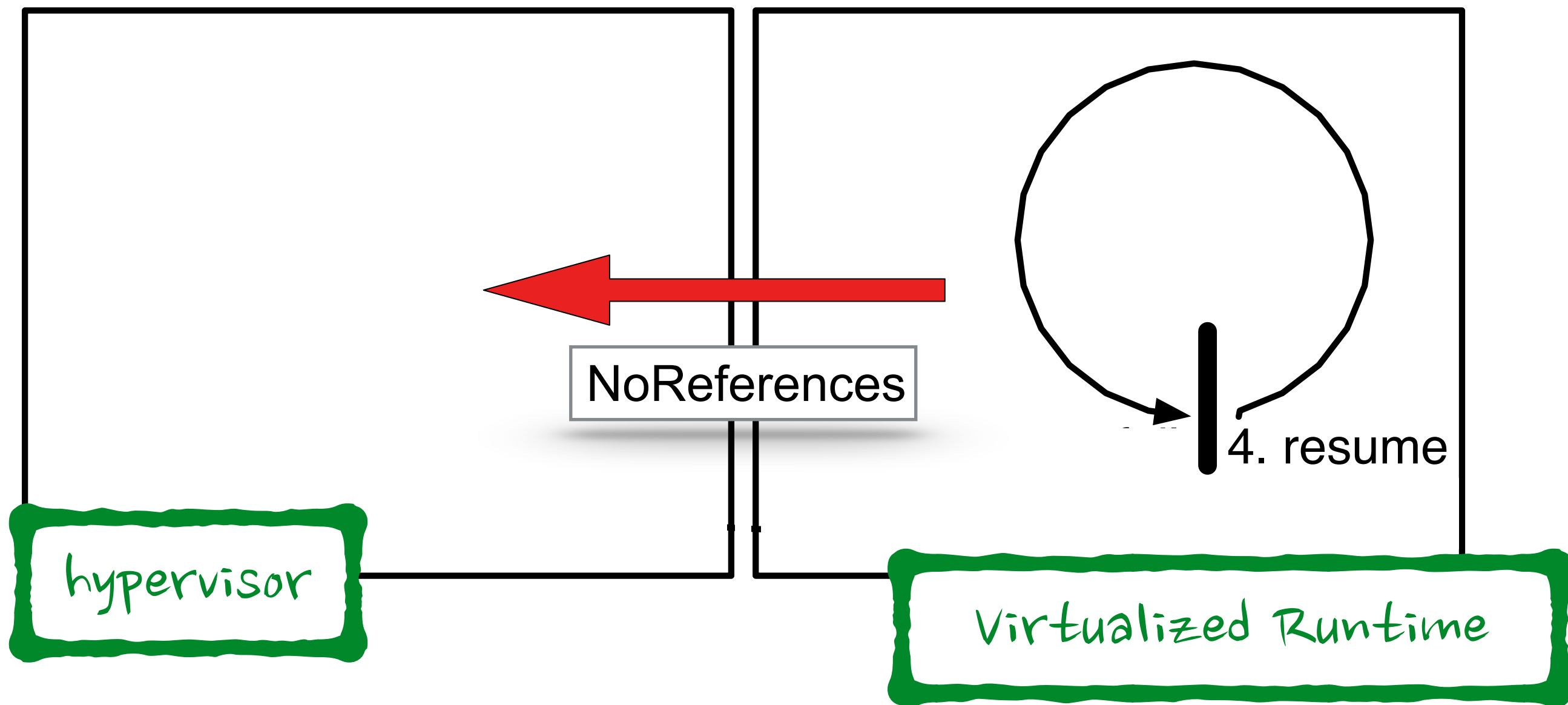
# Run-Fail-Grow



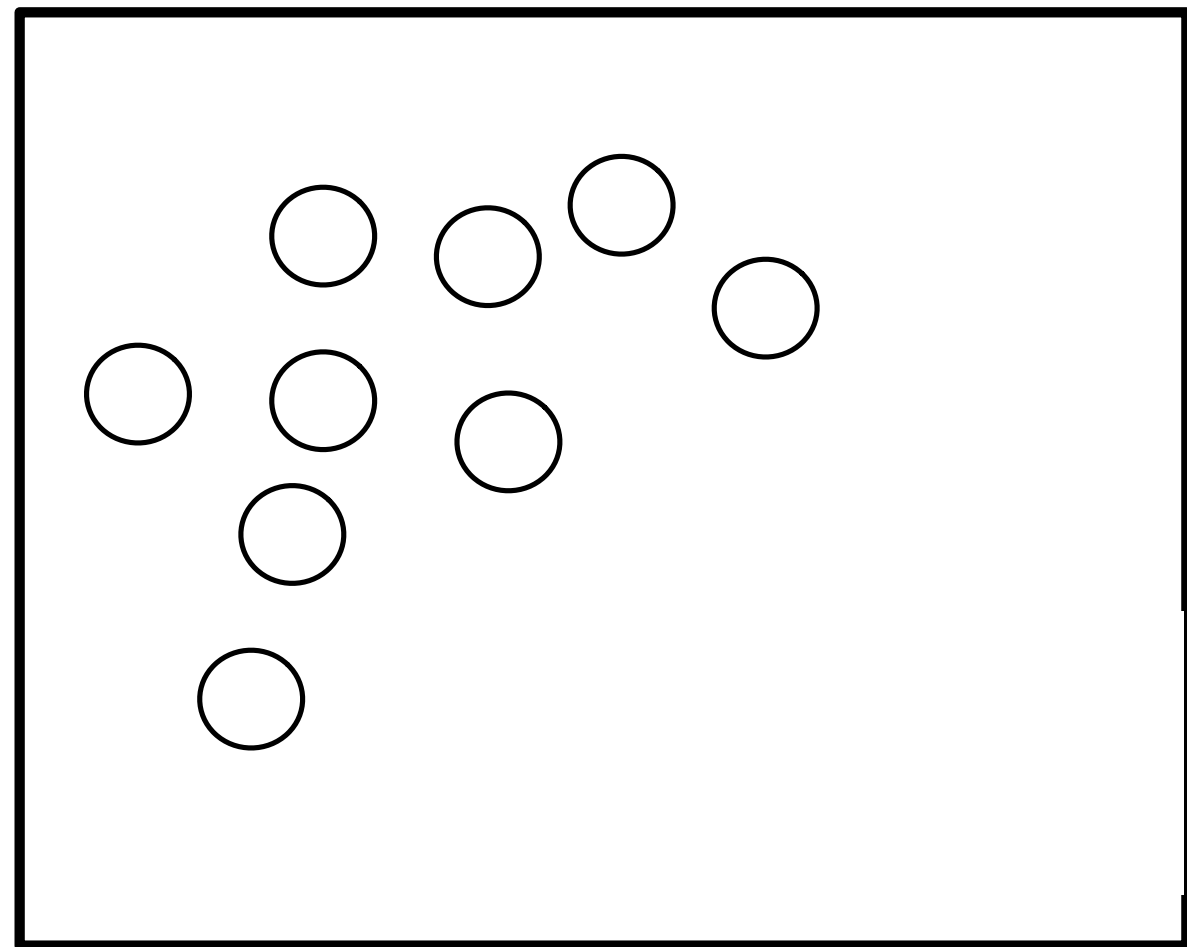
# Run



# No reference from virtualized to hypervisor



# Virtualized is now standalone





# Results

Experiment	Size (KB)	%Saved
Addition	11	99.99%
Simple Reflective Operation	32	99.83%
Factorial 100 + I/O	89	99.39%
Web Simple App	573	96.73%

Experiments with empty seeds

# Object Space

- First-class virtualized runtime
- It's a meta-object! => MOP

## Language Manipulation

e.g., create class

## Scaffolding

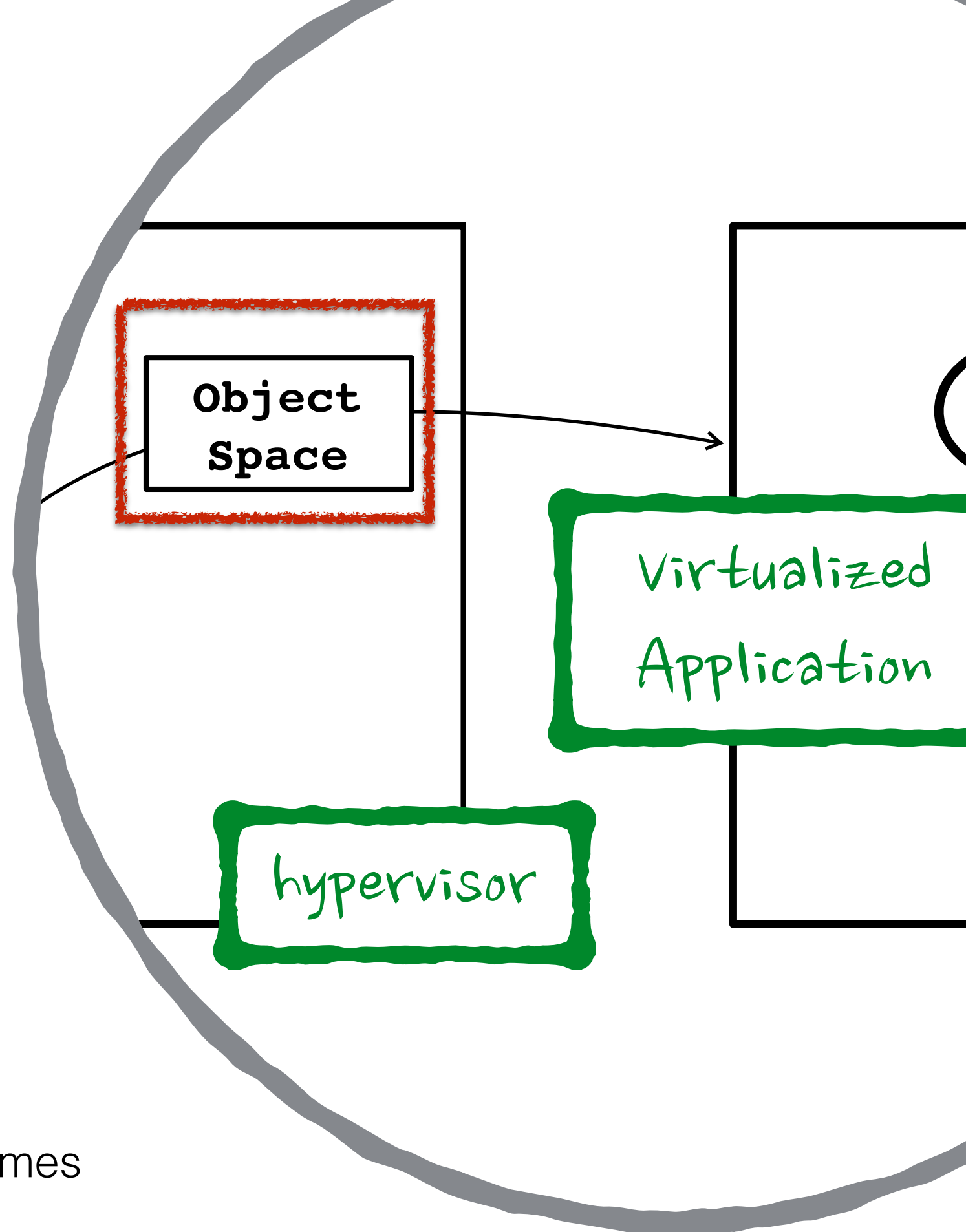
e.g., smart proxies

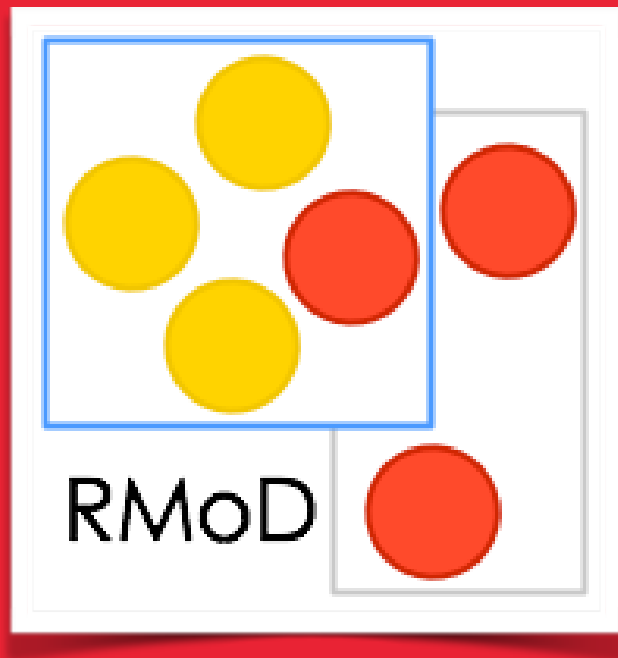
## Object Manipulation

e.g., set slot, get slot

## Execution manipulation

e.g., create process, get stack frames

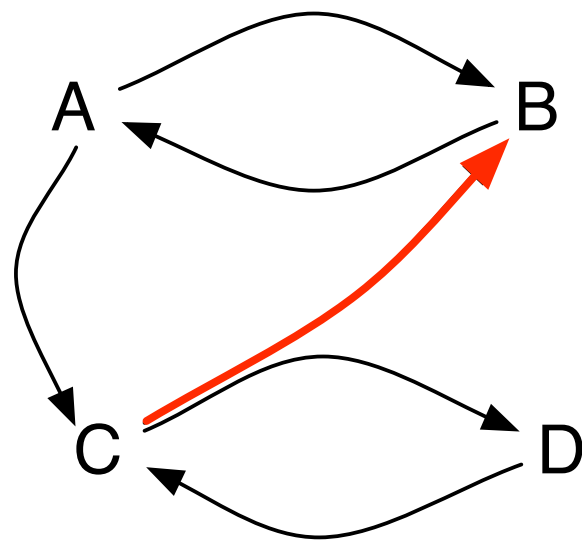




# What about cycles?

[ PhD J. Laval]

# Building a DSM



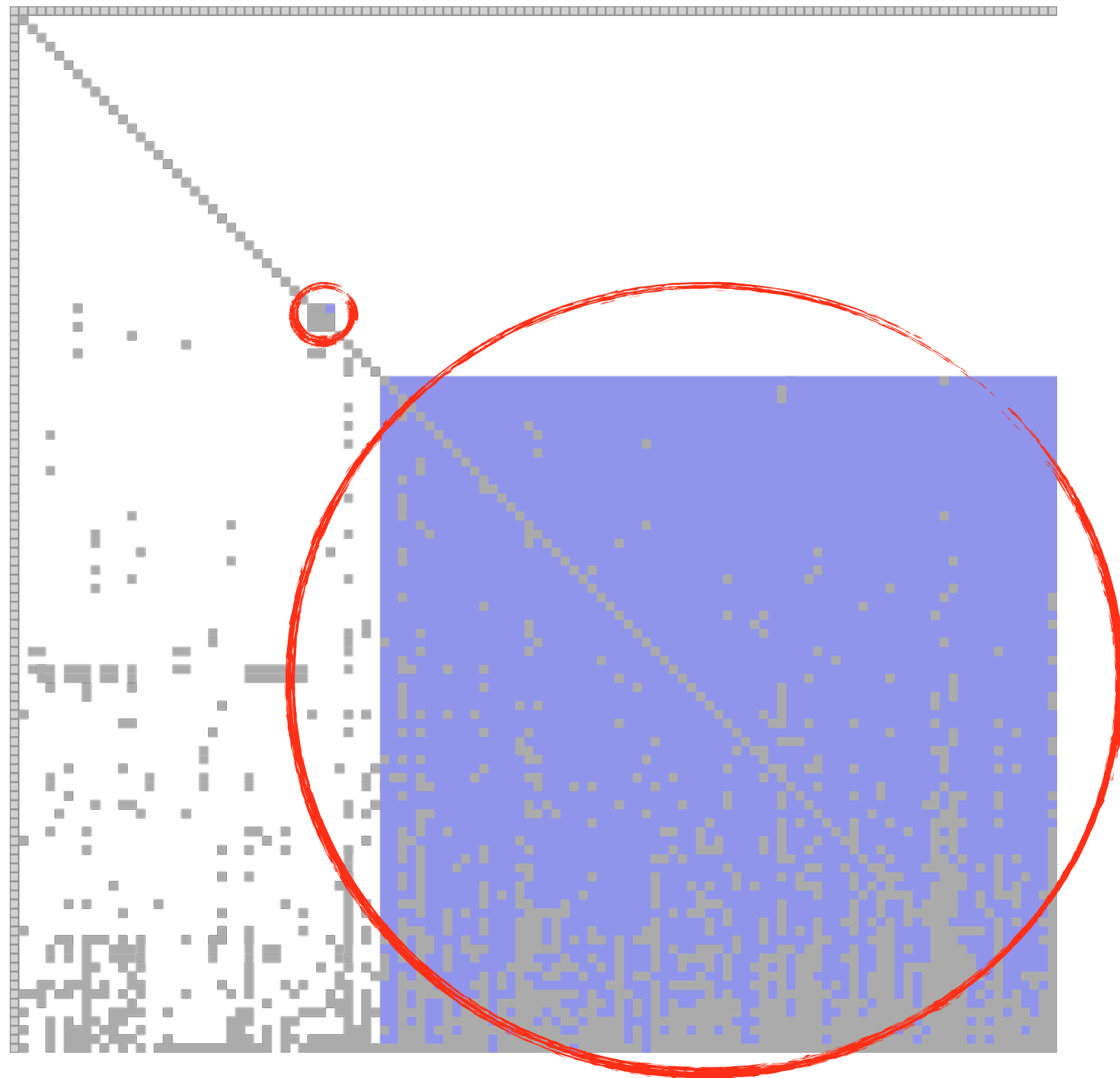
	A	B	C	D
A		X		
B	X		X	
C	X			X
D			X	

	A	B	C	D
A	0	1	0	0
B	1	0	1	0
C	1	0	0	1
D	0	0	1	0

# DSM for software architecture

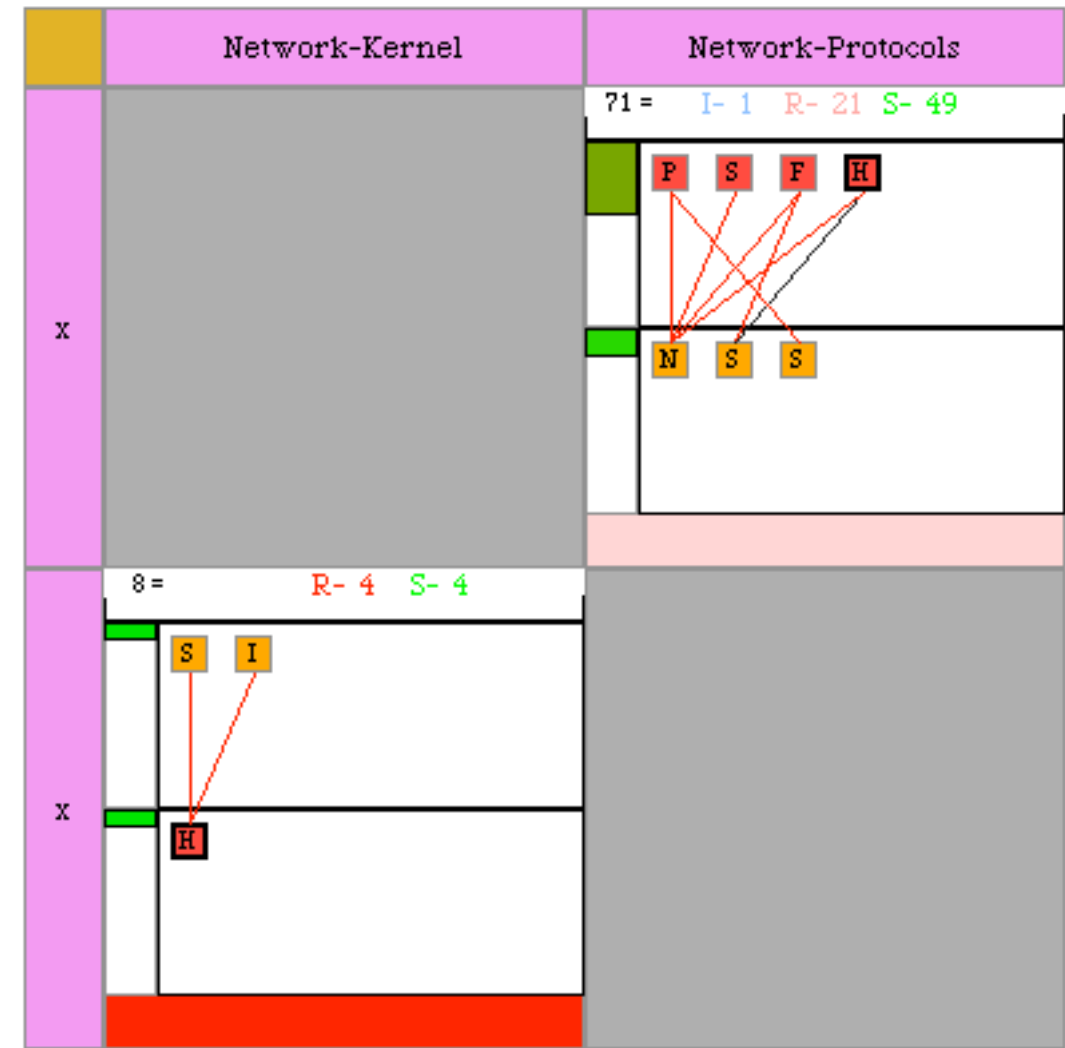
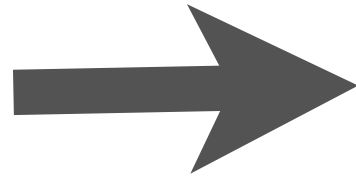
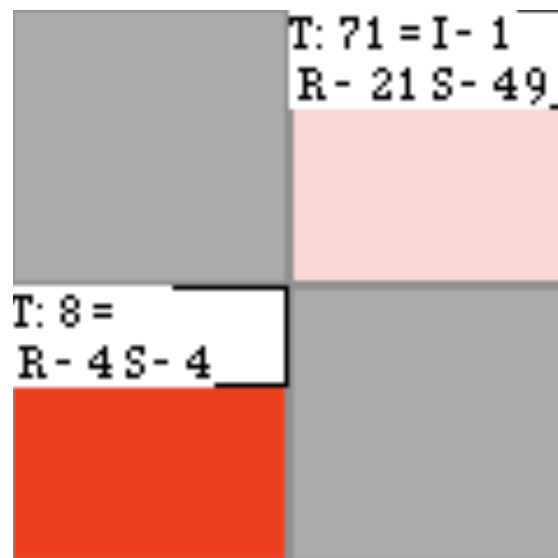
---

*[Sangal 2005]*



PharoCore 1.1, 115 packages - 78 in cycle

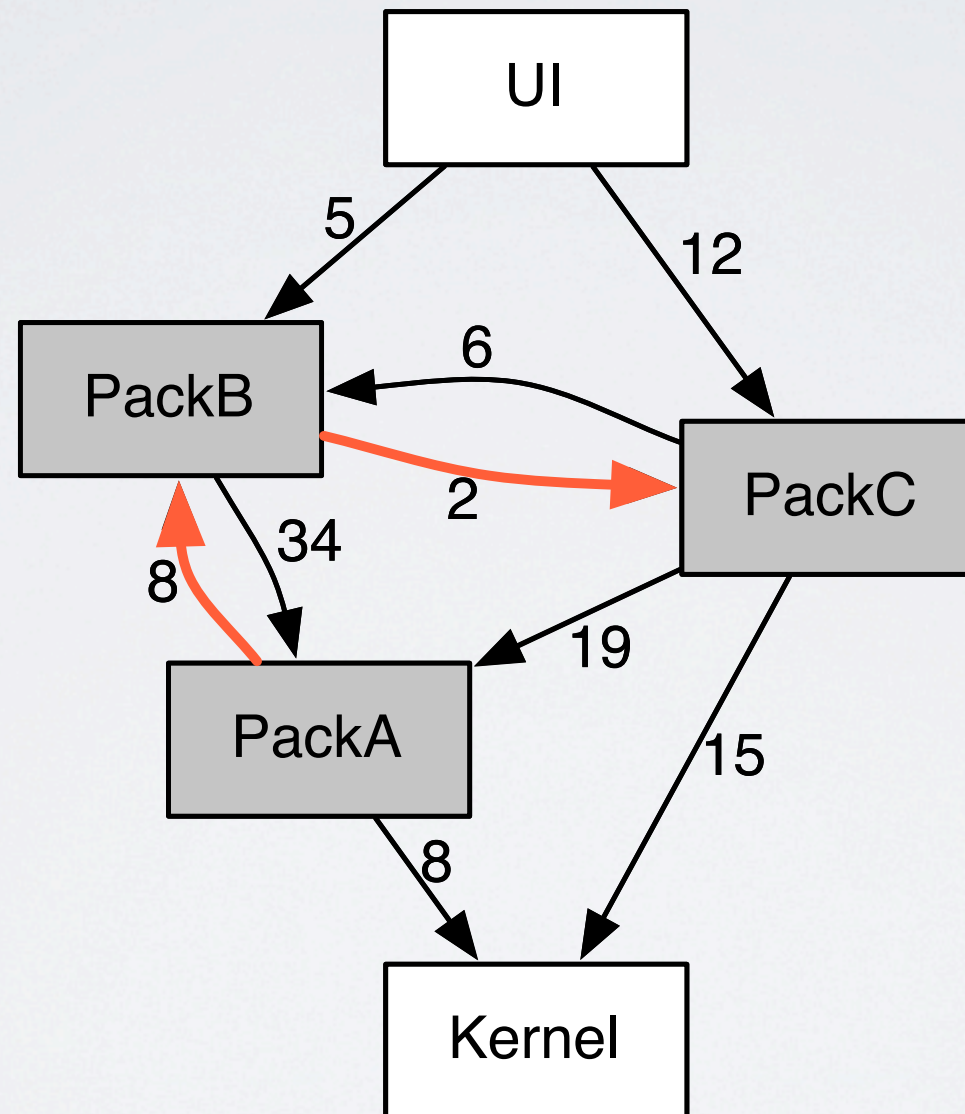
# Causes and distribution







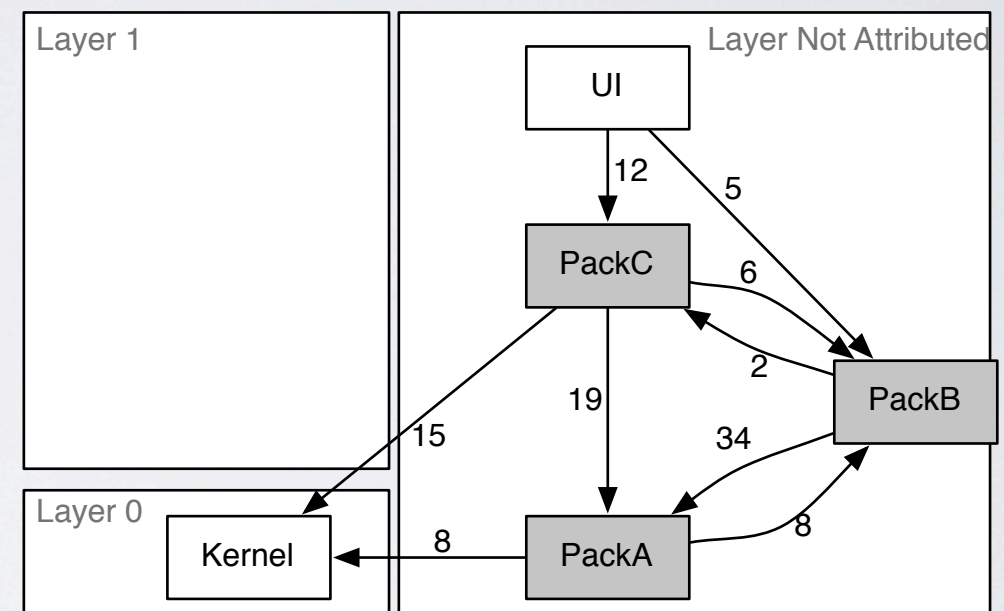
# A Simple Cycle





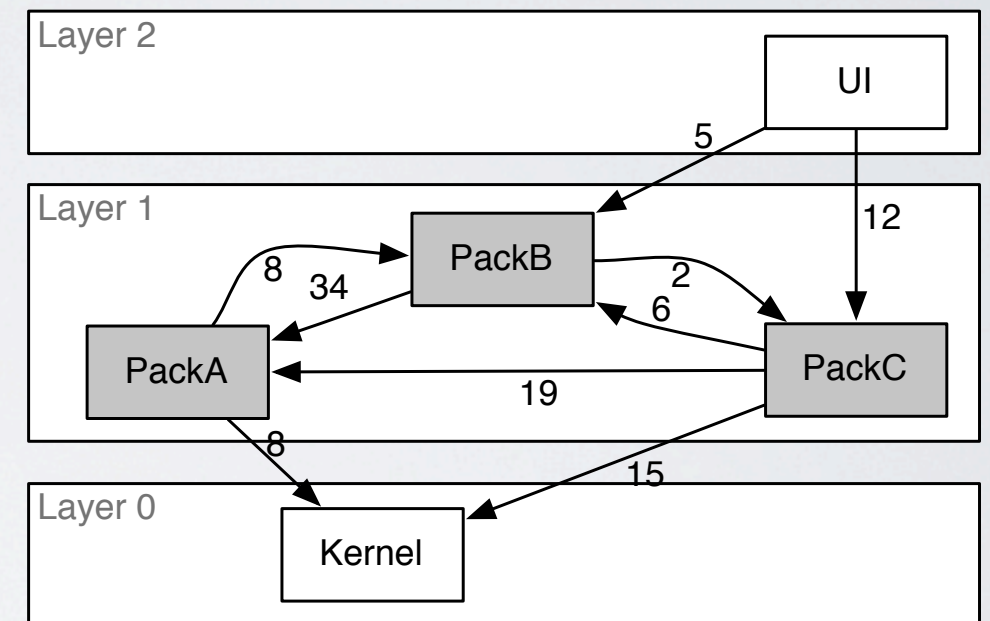
# Possible solutions

- Manual layering
- **Ignore cycles**
- A cycle in one layer
- Remove lightest dependencies



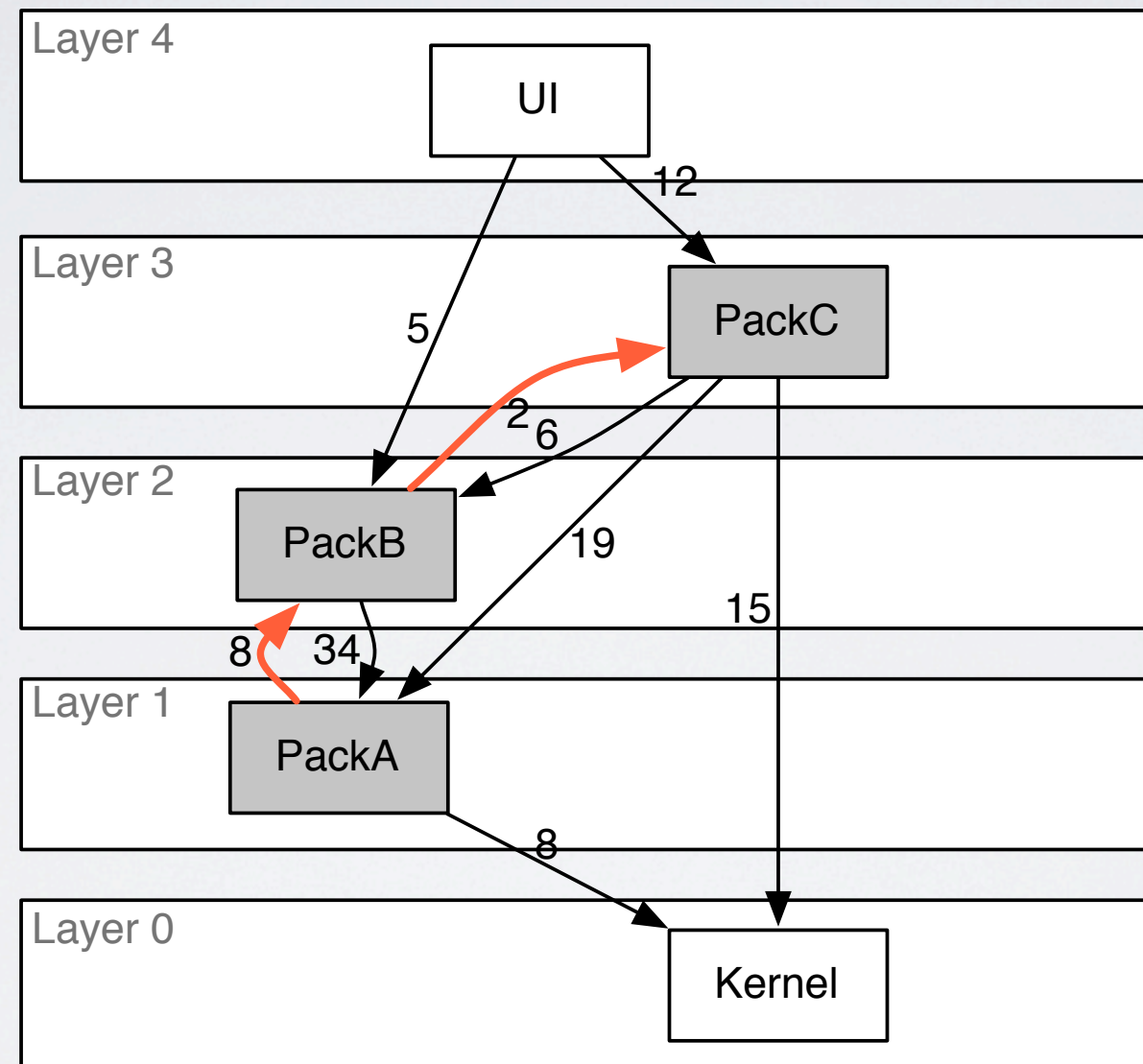
# Possible solutions

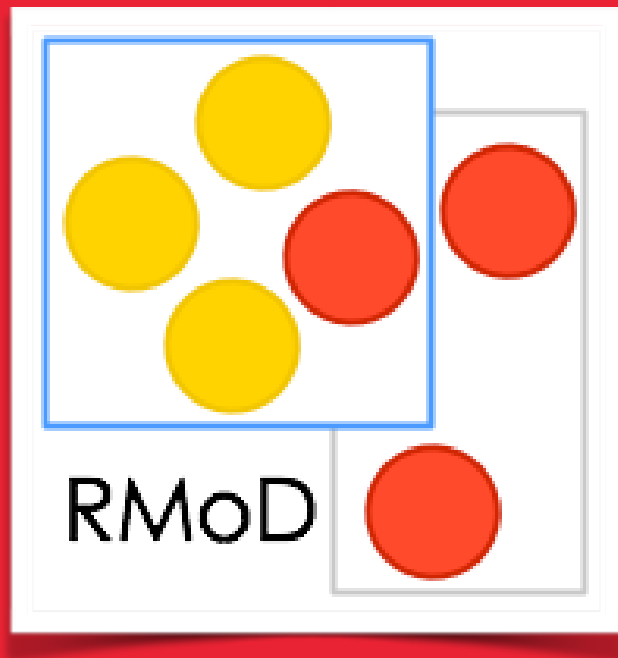
- Manual layering
- Ignore cycles
- **A cycle in one layer**
- Remove lightest dependencies





# Layered Architecture





# Different futures

[ PhD J. Laval]

# How to assess multiple futures?

Before performing a change, can we assess and compare it with multiple other potential futures?

Ok but our models contain millions of entities :)

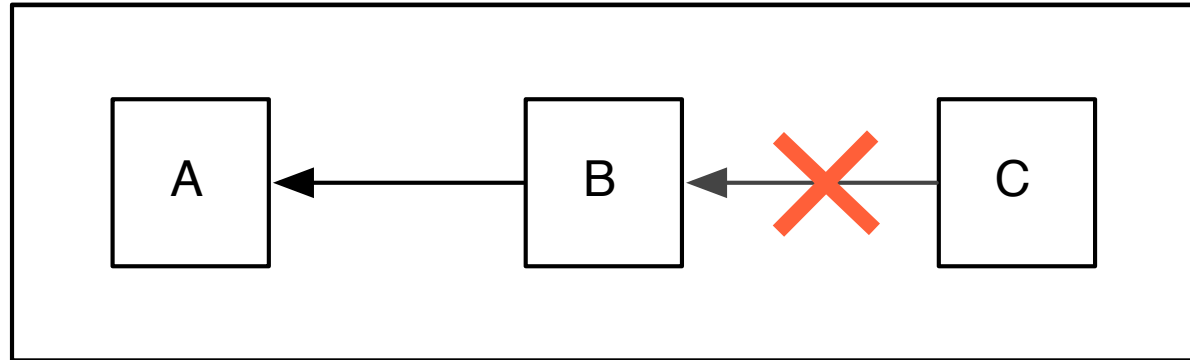
Keeping deltas is tricky



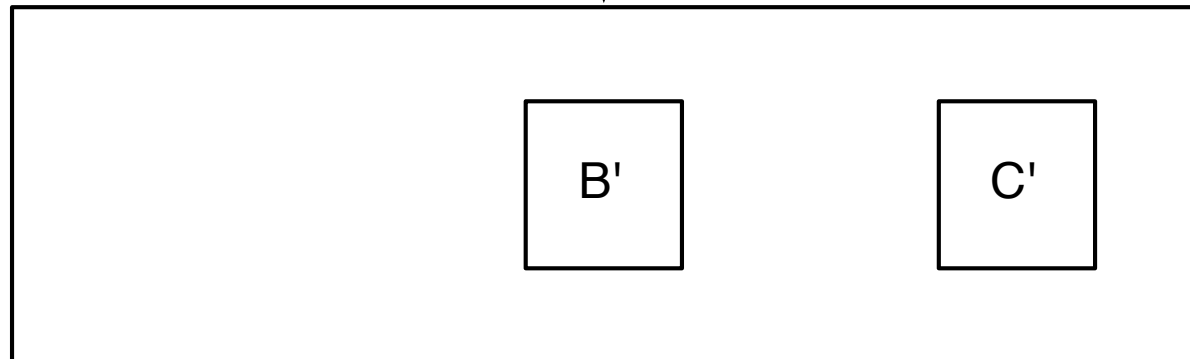
# Orion: Minimal copy

---

v1



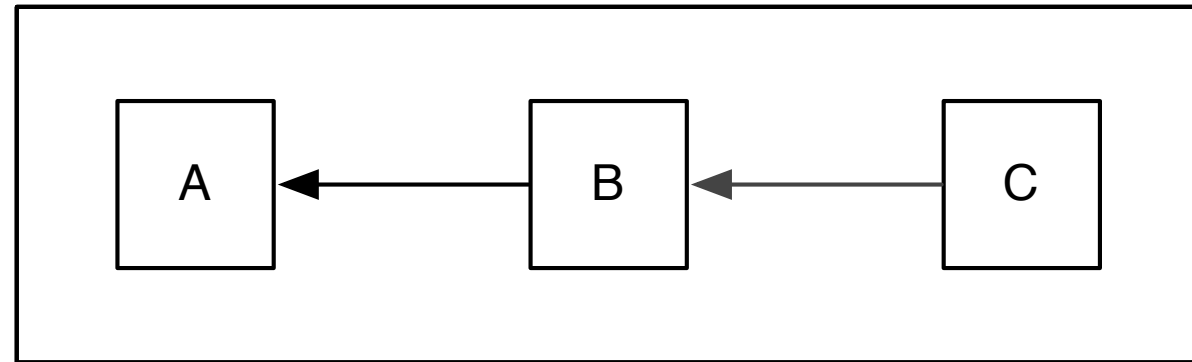
v2



# Orion: Access resolution

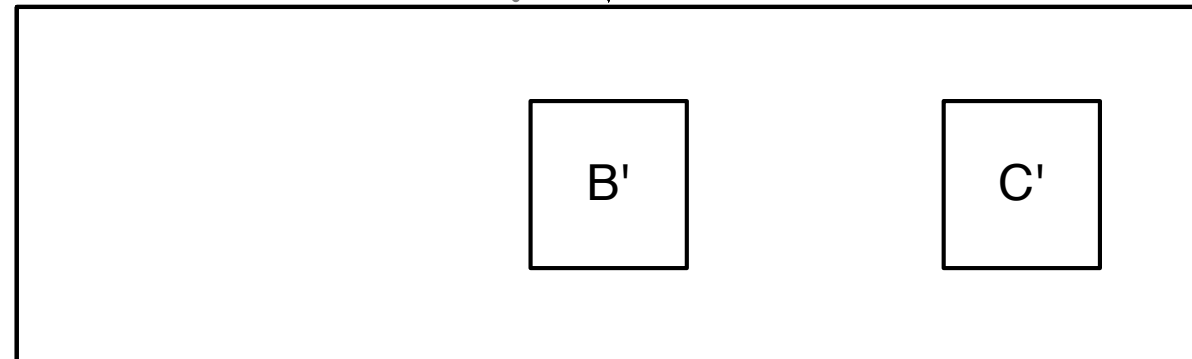
---

v1



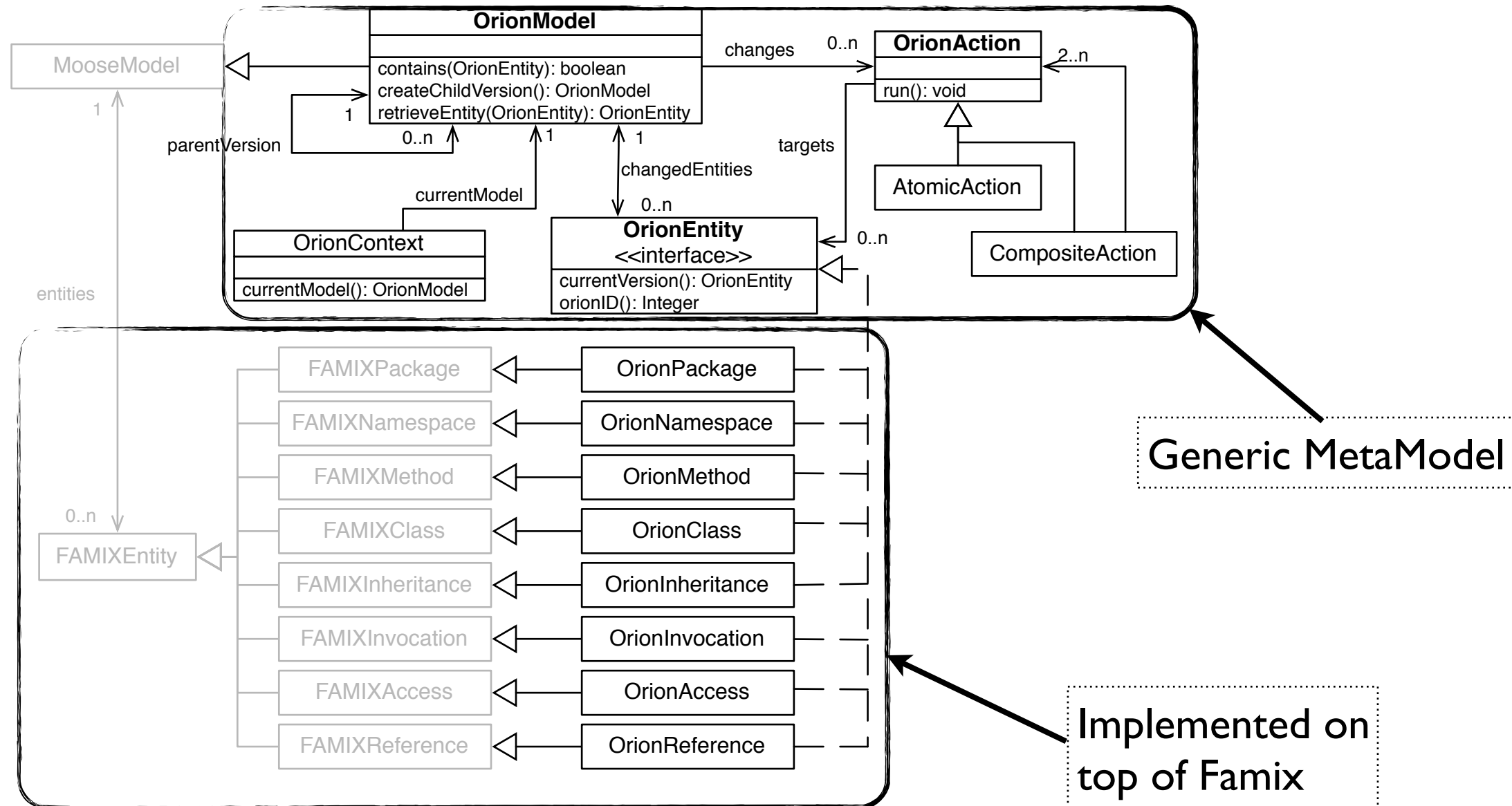
v1: A references references  
 $\Rightarrow \{C\}$

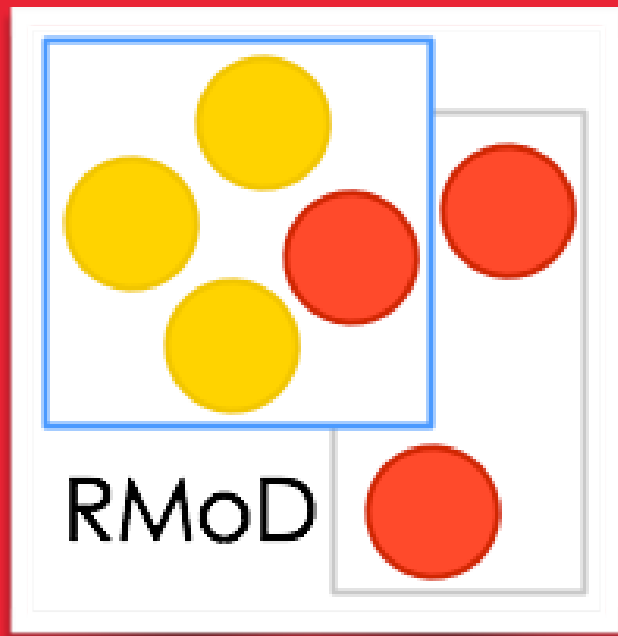
v2



v2: A references references  
 $\Rightarrow \{\}$

# Orion Meta-model





# Evolution in the large: GWT to angular

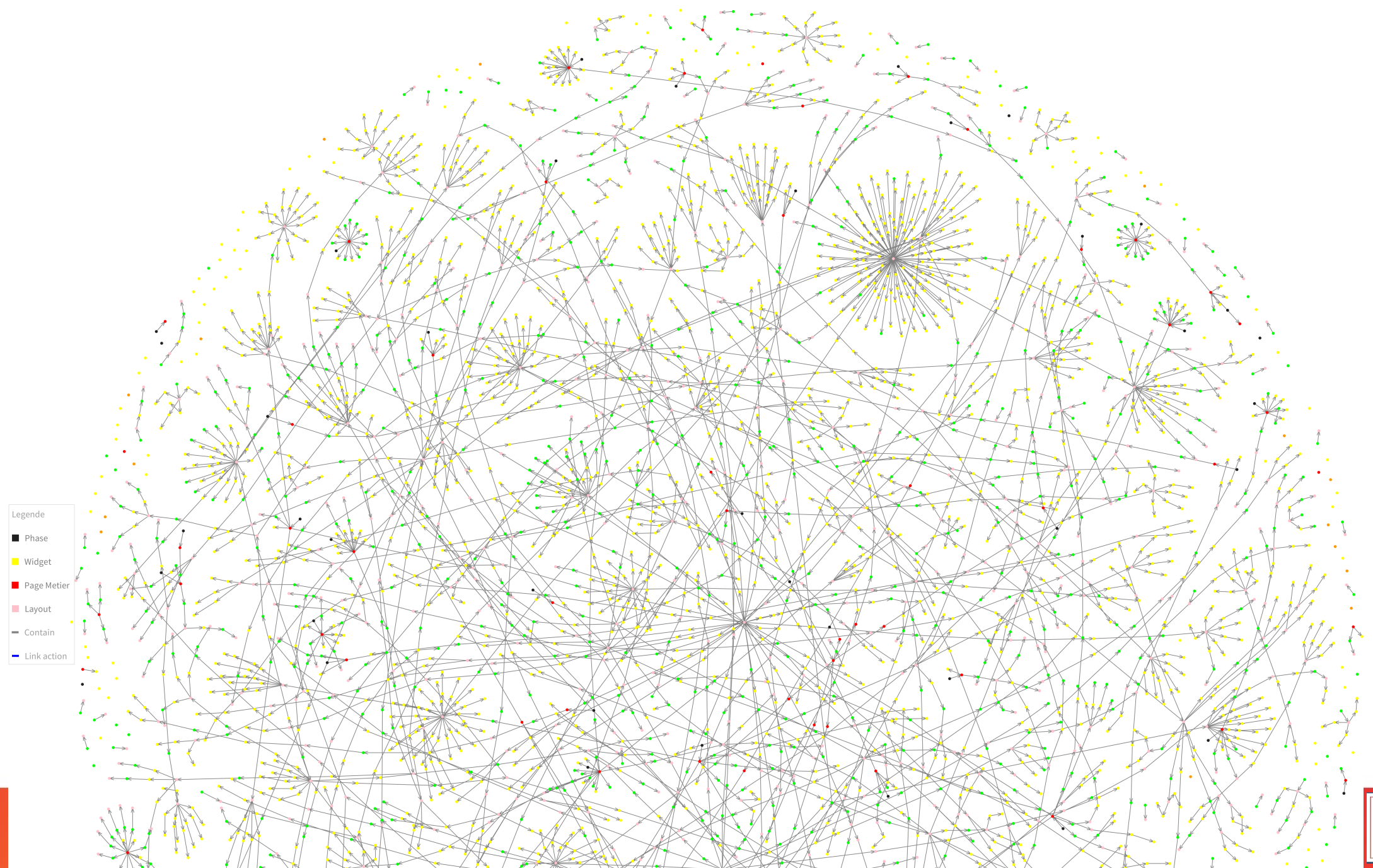
PhD CIFRE of B. Verhaeghe for Berger-Levrault

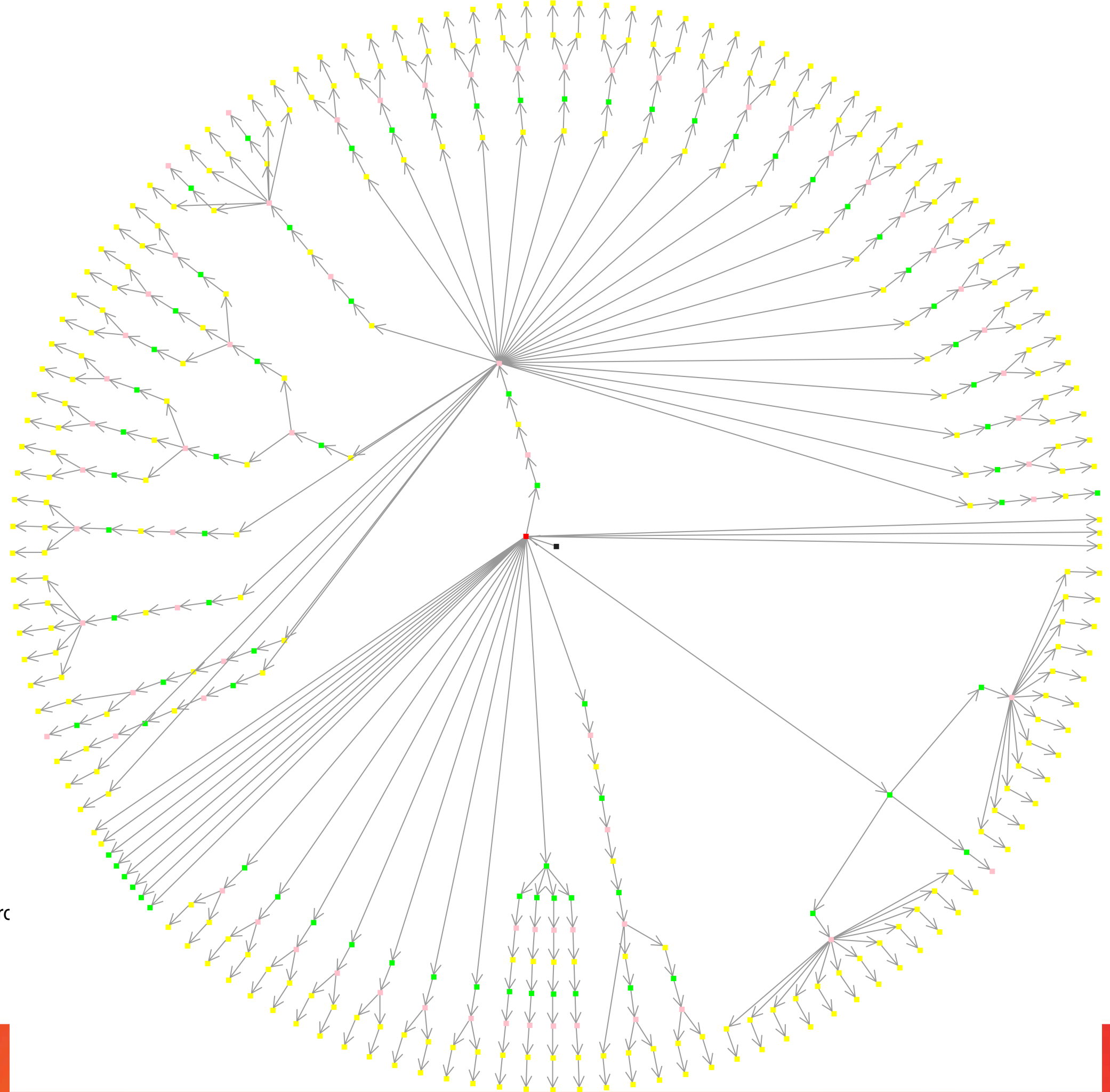
# GWT Typical Application

- 26 000 Classes
- 450 web pages
- 974 000 invocations
- 1 063 163 LOC (UI)

# GWT big kitchen (bac a sable)

bigKitchen ~ 8979 éléments

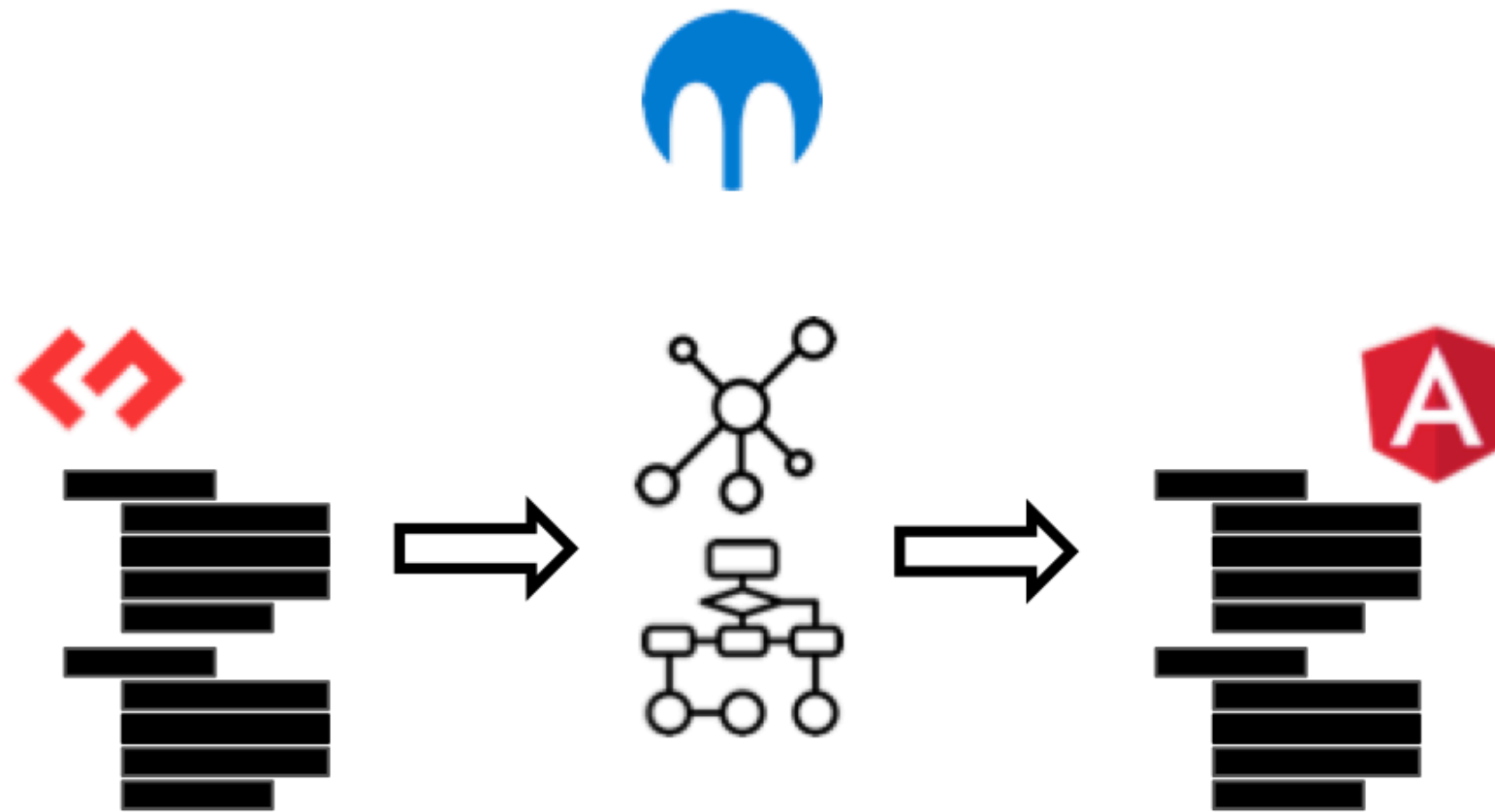




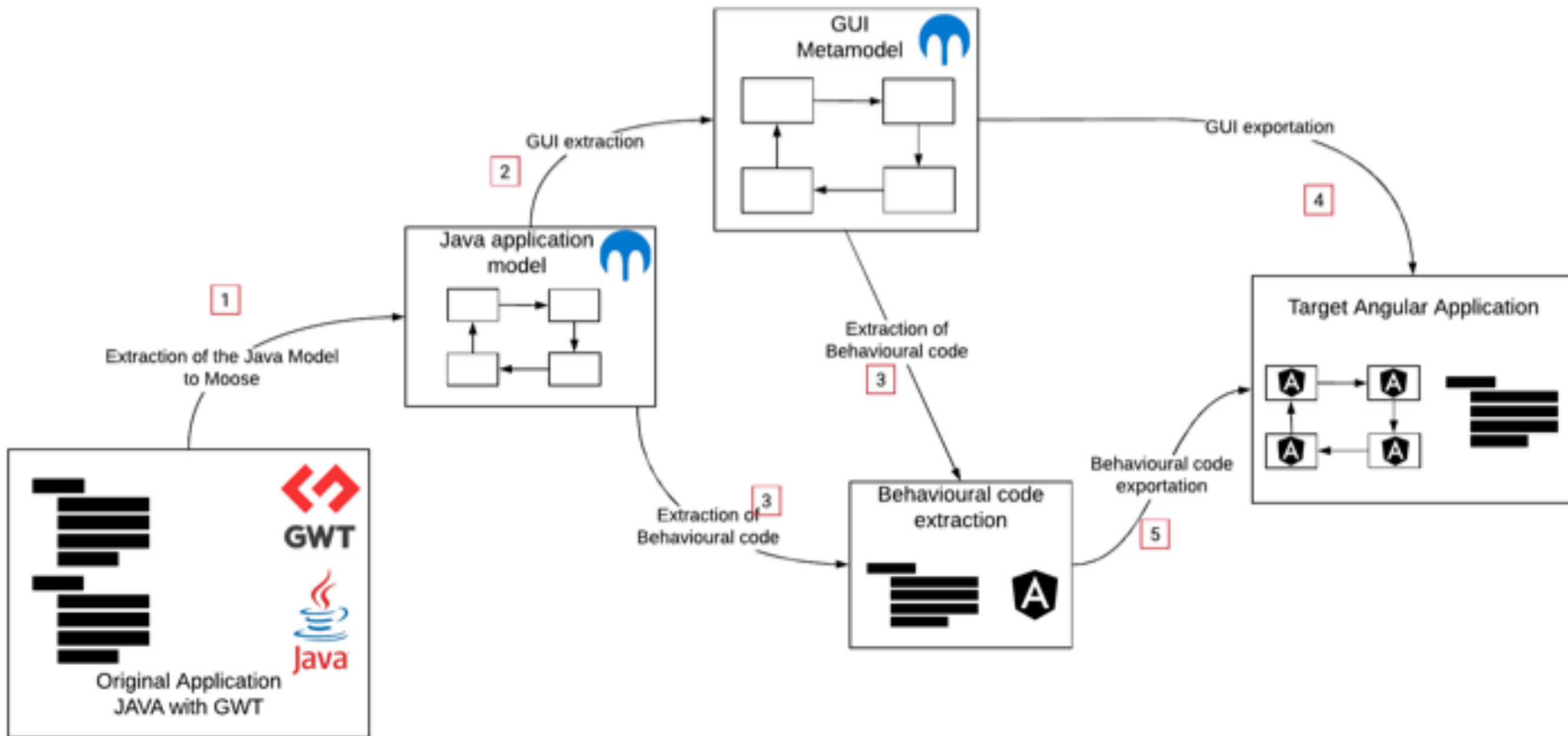
Noir-> des phase/page web  
Rouge -> Business Page (grc  
Gris -> Layout  
Vert -> Widget container  
Jaune -> Widget leaf



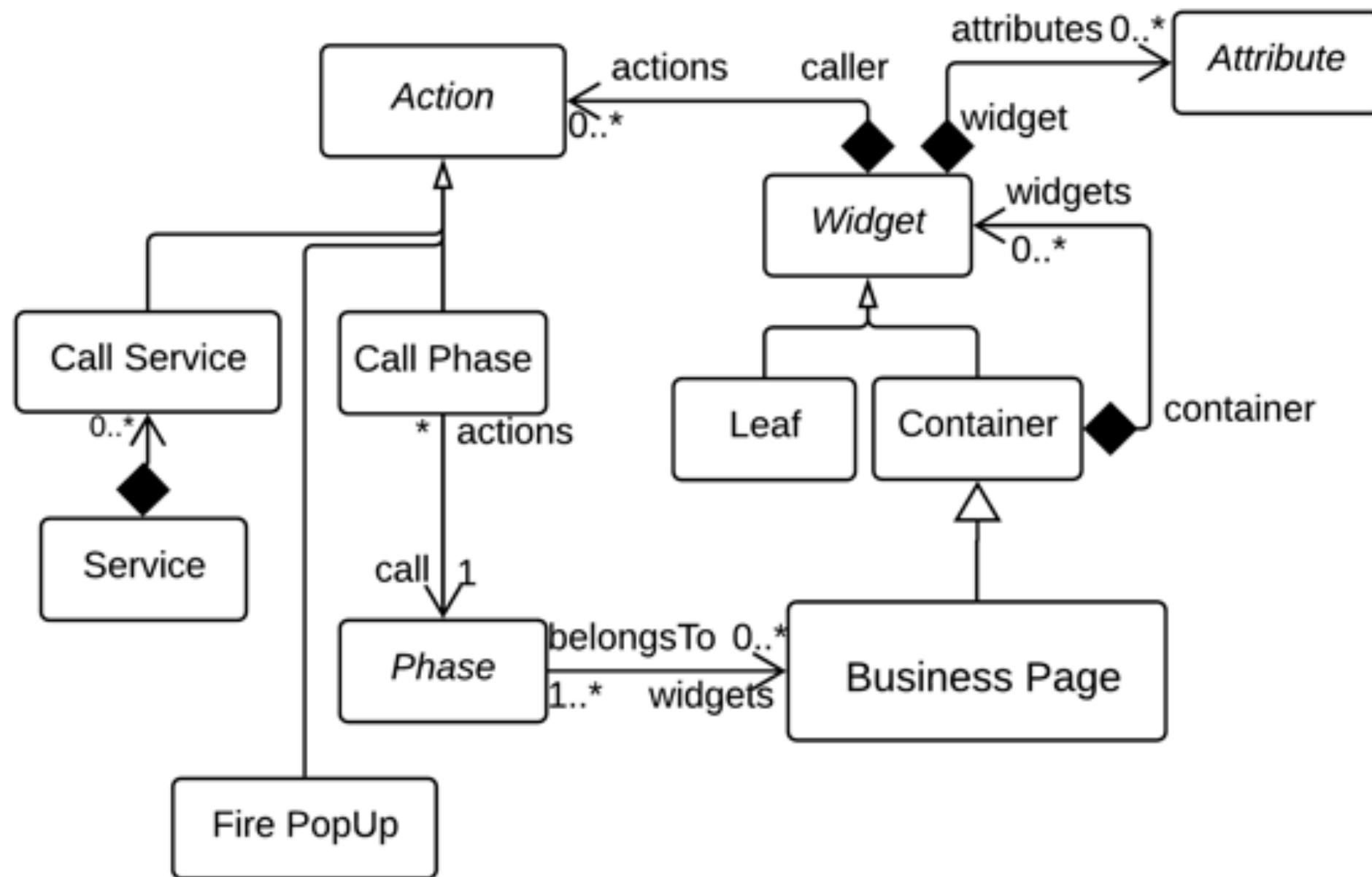
# GWT -> Models -> Angular \*



# GWT -> Models -> Angular \*



# Metamodel



# Results

## Export



### Navigation par phase

- ▶ Ouvrir un onglet
- ▶ Ouvrir une boîte de dialogue modale
- ▶ Ouvrir une boîte de dialogue non modale
- ▶ Ouvrir une boîte de dialogue unique non modale

### Boîtes de dialogue (hors phase)

- ▶ Boîte de dialogue modale
- ▶ Boîte de dialogue non modale
- ▶ Message d'information
- ▶ Message d'avertissement
- ▶ Message d'erreur (métier)
- ▶ Message de confirmation
- ▶ Message d'erreur (exception)



### Navigation par phase

- ▶ Ouvrir un onglet
- ▶ Ouvrir une boîte de dialogue modale
- ▶ Ouvrir une boîte de dialogue non modale
- ▶ Ouvrir une boîte de dialogue unique non modale

### Boîtes de dialogue (hors phase)

- ▶ Boîte de dialogue modale
- ▶ Boîte de dialogue non modale
- ▶ Message d'information
- ▶ Message d'avertissement
- ▶ Message d'erreur (métier)
- ▶ Message de confirmation
- ▶ Message d'erreur (exception)

# Results

## Export



### Etiquettes formatées (pour les listes)

Montant :	10 000 000,00 €
Pourcentage :	1,50%
Booléen (1) :	Oui
Booléen (0) :	Non
Date :	26/06/2018
Durée :	3 ans, 2 mois et 1 jour
Énumération :	jour(s)
Entier :	999999999
Entier long :	999999999999999999
Entier short :	999

### Etiquettes formatées (pour les listes)

Montant :Pourcentage :Booléen (1) :Booléen (0) :Date :Durée :Énumération :Entier :Entier long :Entier short :



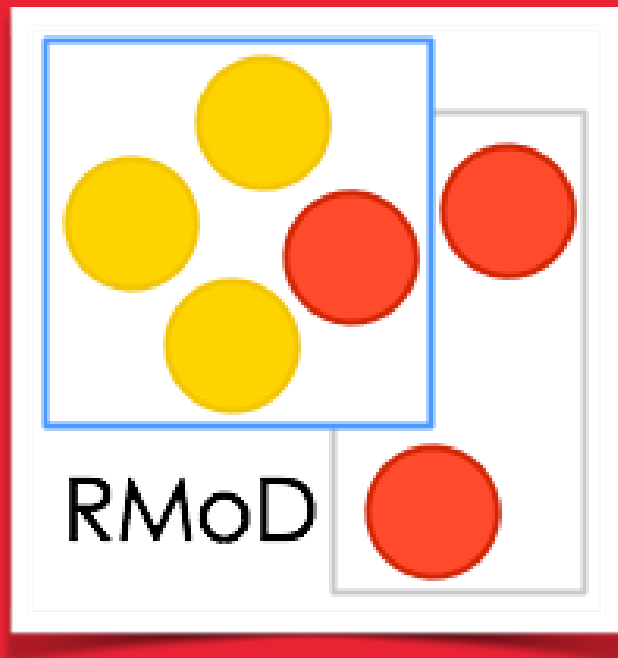
# Future work

Behavior code model

Specific representation of layout

Extract complex structure (such table data structure)

How to evaluate GUI migration?



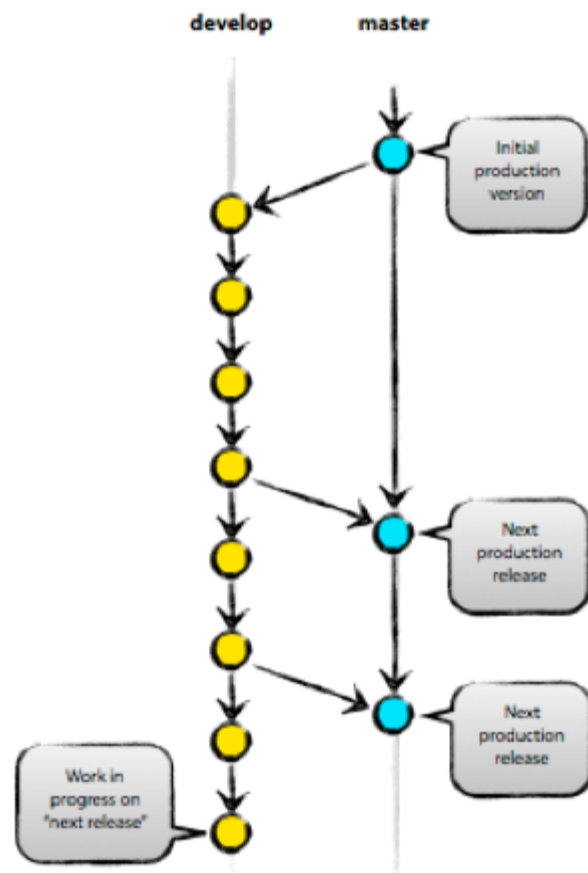
# How can we help merging?

## What is the impact of a change?

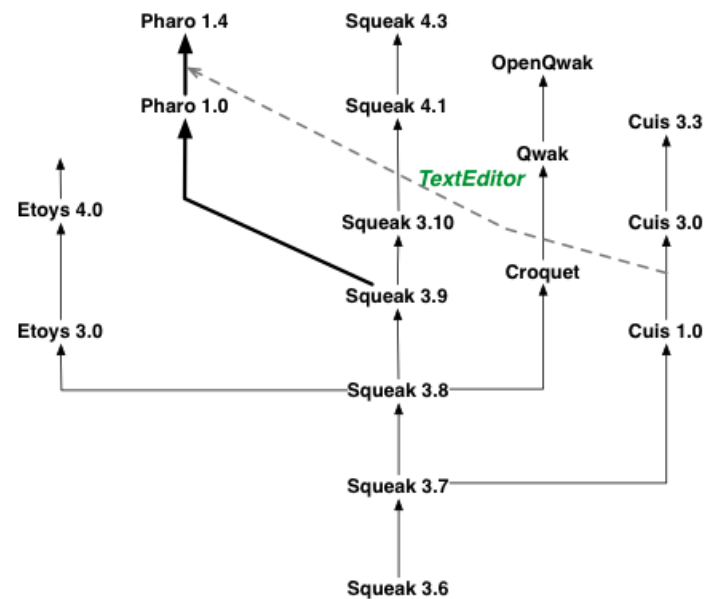
### PhD of V. Uquillas-Gomez (Paid VUB)



# How to support merging branches?



**Git**



**Forks**

**Manual tasks are needed**

**Dependencies between changes**

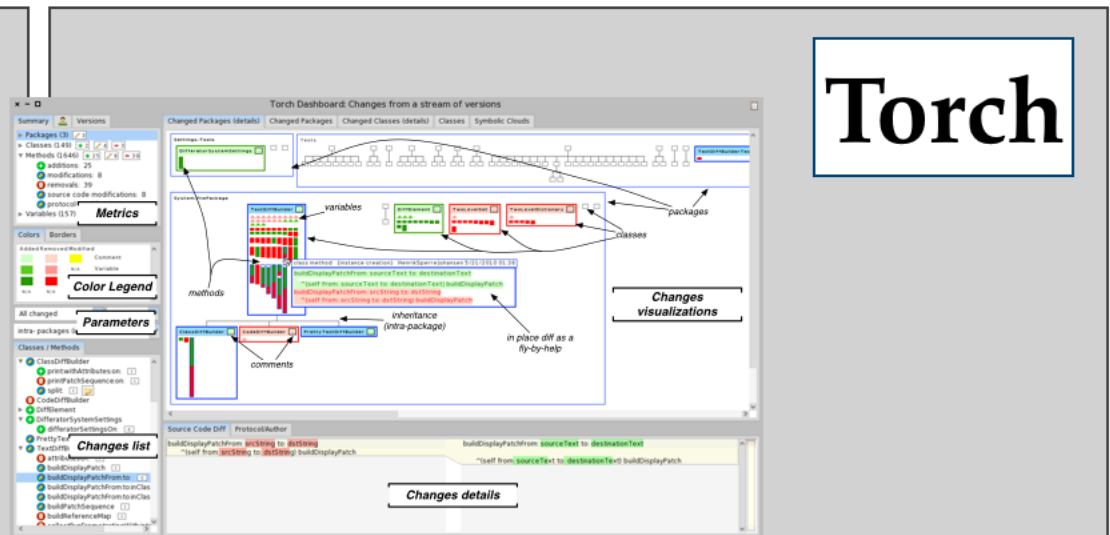
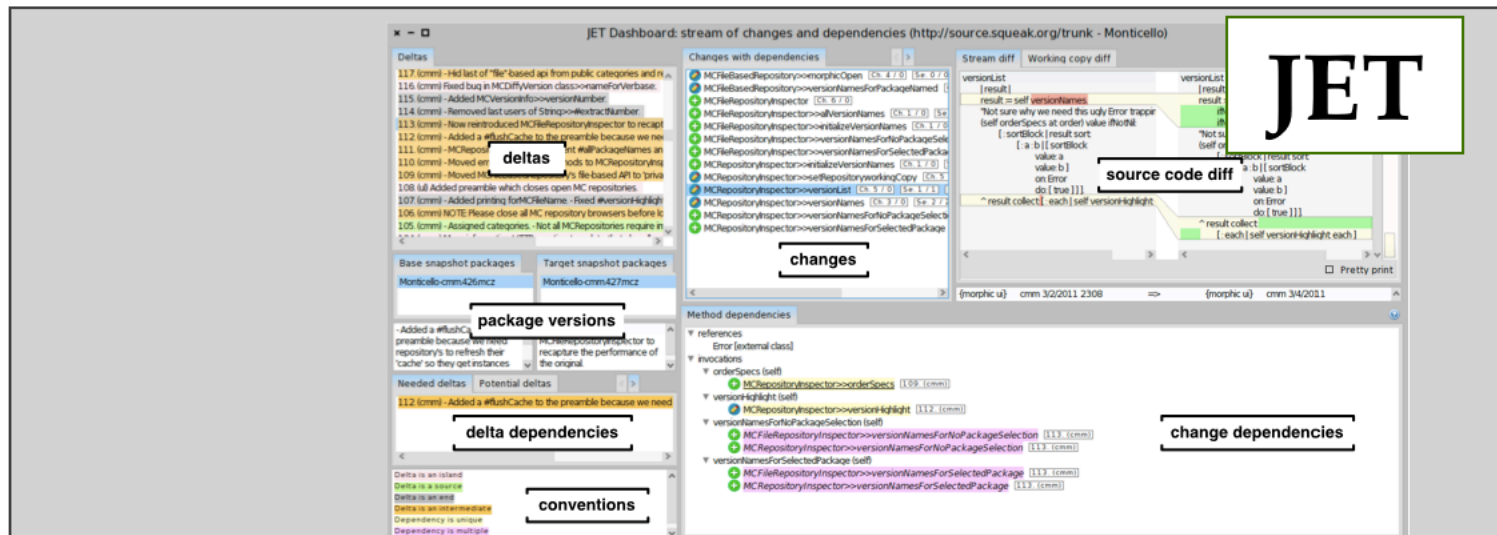
**Integrator is not the author of the changes**

**No guarantee that the system will work**

# Changes and branches

Stream of changes (chains of commits)

Single delta (commit)



Change & Dependency Meta-Model and Analyses (RingC)

History Meta-Model and Analyses

Single-delta Change Meta-Model and Analyses (RingS)

Source Code Meta-Model (Ring)

# Understanding a change

Torch Dashboard: Changes from SLICE-Issue1709-EnhancedTextDiffBuilder (ancestor) to SLICE-Issue1709-EnhancedTextDiffBuild

Change Summary

- Packages (32) 3
- Classes (149) 2 4 3
- Methods (1646) 25 8 39
- Variables (157) 9 18
  - + additions: 9
  - removals: 18

Colors Borders

Added	Removed	Modified	
Light Green	Light Orange	Yellow	Comment
Green	Pink	N/A	Variable
Dark Green	Red	Blue	Method source
N/A	N/A	Orange	Class' package

Viz. Class Status: All changed

Viz. Width: 900

Viz. Relationships: intra- packages

Classes / Methods

- PrettyTextDiffBuilder
- TextDiffBuilder
  - attributesOf: [i]
  - buildDisplayPatch [i]
  - buildDisplayPatchFrom:to: [i]
  - buildDisplayPatchFrom:to:inC [i]
  - buildDisplayPatchFrom:to:inC [i]
  - buildPatchSequence [i]
  - buildReferenceMap [i]
  - collectRunFrom:startingWith: [i]
  - destString: [i]
  - detectShiftedRuns [i]
  - findMatches [i]
  - formatLine: [i]
  - from:to: [i]

Changed Packages (details)

- Tests-System
  - TextDiffBuilderTest
- Settings-Tools
  - DifferatorSystemSettings
- System-FilePackage
  - TextDiffBuilder
  - TwoLevelSet
  - DiffElement
  - TwoLevelDictionary

Source Code Diff

Source Code

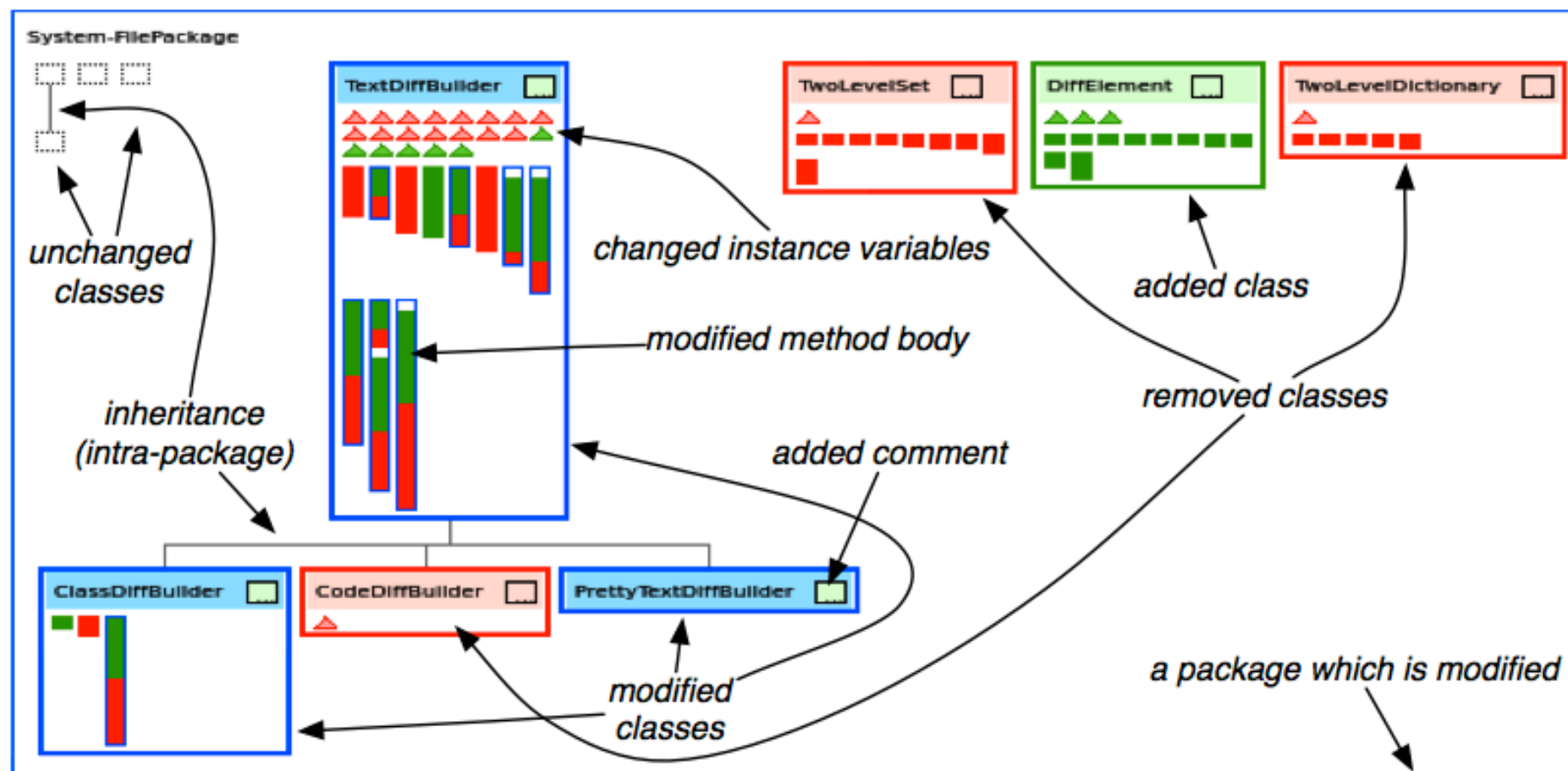
Protocol/Author

```
buildDisplayPatch
^Text streamContents:[ :stream |
  self printPatchSequence: self buildPatchSequence on: stream
]
```

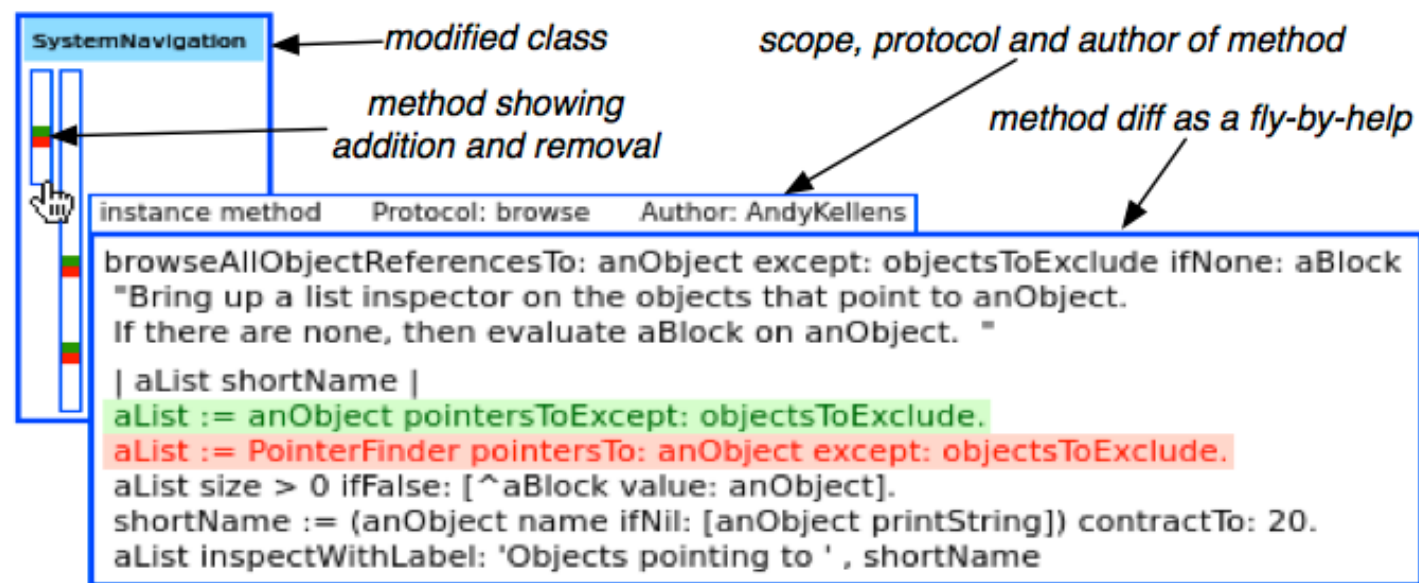
```
buildDisplayPatch
^Text streamContents: [ :stream |
  self
    patchSequenceDofMatch: [ :string |
      self print: string withAttributes: NormalTextAttributes
    ]
    ifInsert: [ :string |
      self print: string withAttributes: InsertTextAttributes
    ]
    ifRemove: [ :string |
      self print: string withAttributes: RemoveTextAttributes
    ]
  ]

```

# Package Structure



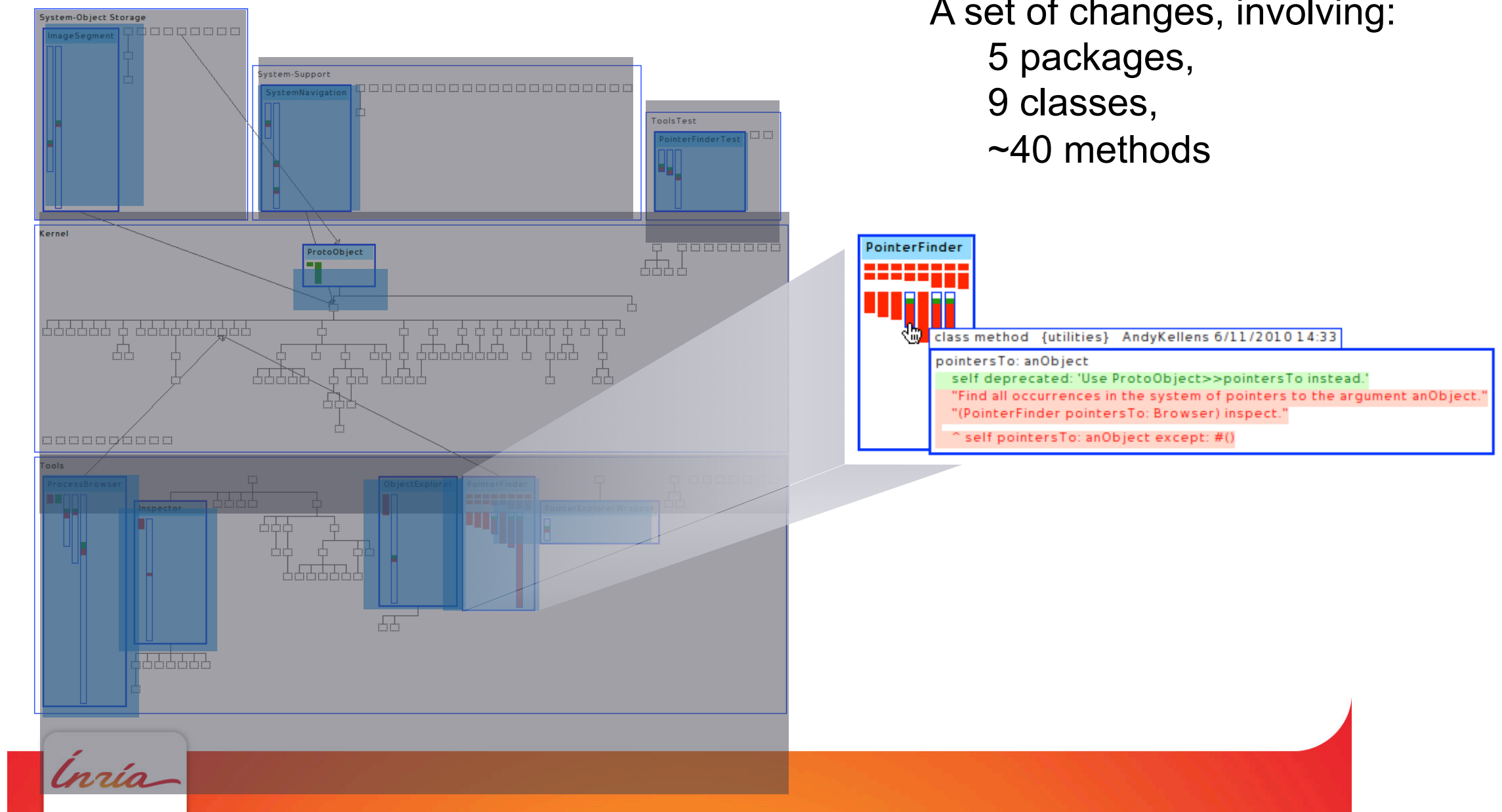
# Omnipresent source code



# Torch: Which changes?

## Where? Who? What?

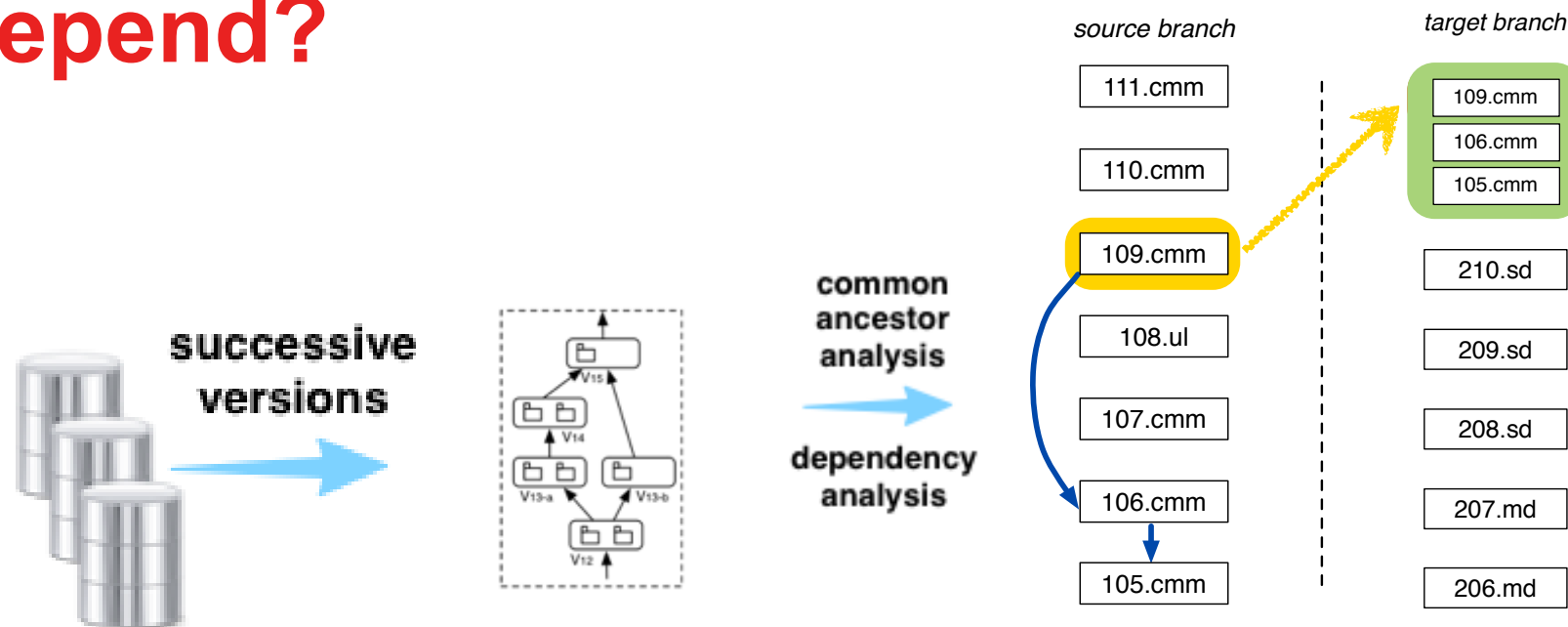
A set of changes, involving:  
5 packages,  
9 classes,  
~40 methods





# Streams of Changes:

## On what other changes does this change depend?

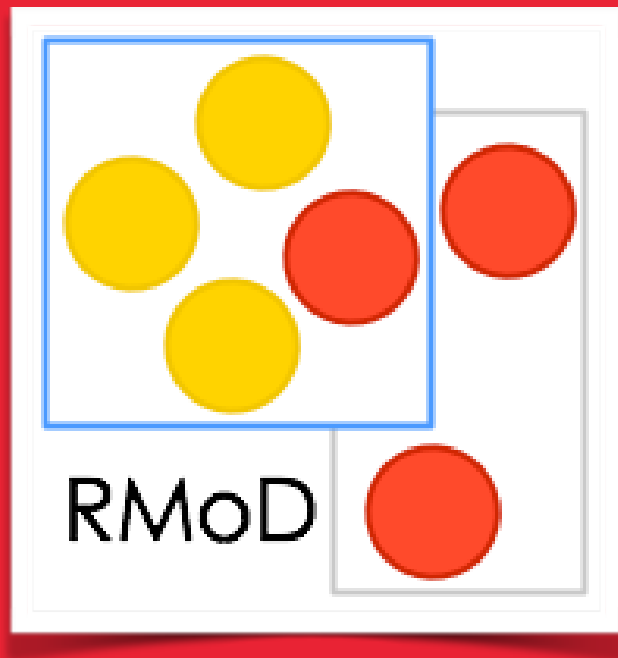


characterization  
of dependencies  
and deltas

The screenshot shows the JET Dashboard interface for a stream of changes and dependencies. The main window is titled "JET Dashboard: stream of changes and dependencies (http://source.squeak.org/trunk - Monticello)". It features several panels:

- Changes with dependencies:** A list of changes, each with a green circular icon and a description. Examples include "MCFileBasedRepository>>morphicOpen" and "MCFileBasedRepository>>versionNamesForPackageNamed".
- Stream diff / Working copy diff:** A panel showing a comparison between two versions of a file, with a "source code diff" view.
- Method dependencies:** A panel showing a list of method dependencies, including "references" and "invocations".
- package versions:** A panel showing a list of package versions, including "Base snapshot packages" and "Target snapshot packages".
- delta dependencies:** A panel showing a list of delta dependencies, including "Needed deltas" and "Potential deltas".
- conventions:** A panel showing a list of conventions, including "Delta is an island", "Delta is a source", "Delta is an intermediate", "Dependency is unique", and "Dependency is multiple".





# Other Software Evolution Challenges

Blockchain, test selection,

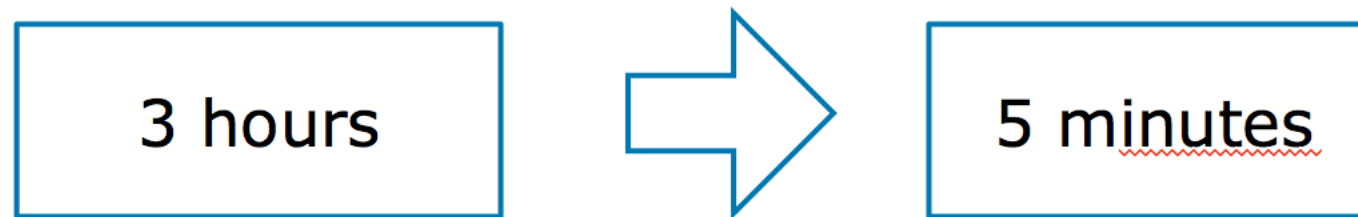
# Which tests to rerun?

## [PhD CIFRE ATOS V. Blondeau]

### Test Selection

---

#### Automatic Test Selection



Select only the tests related to the changes  
and run only those ones

# Blockchain modelisation

[PhD - S. Bragagnolo - Berger-Levrault]

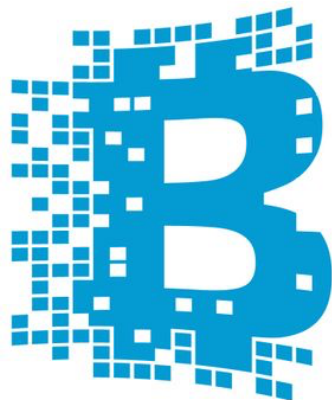
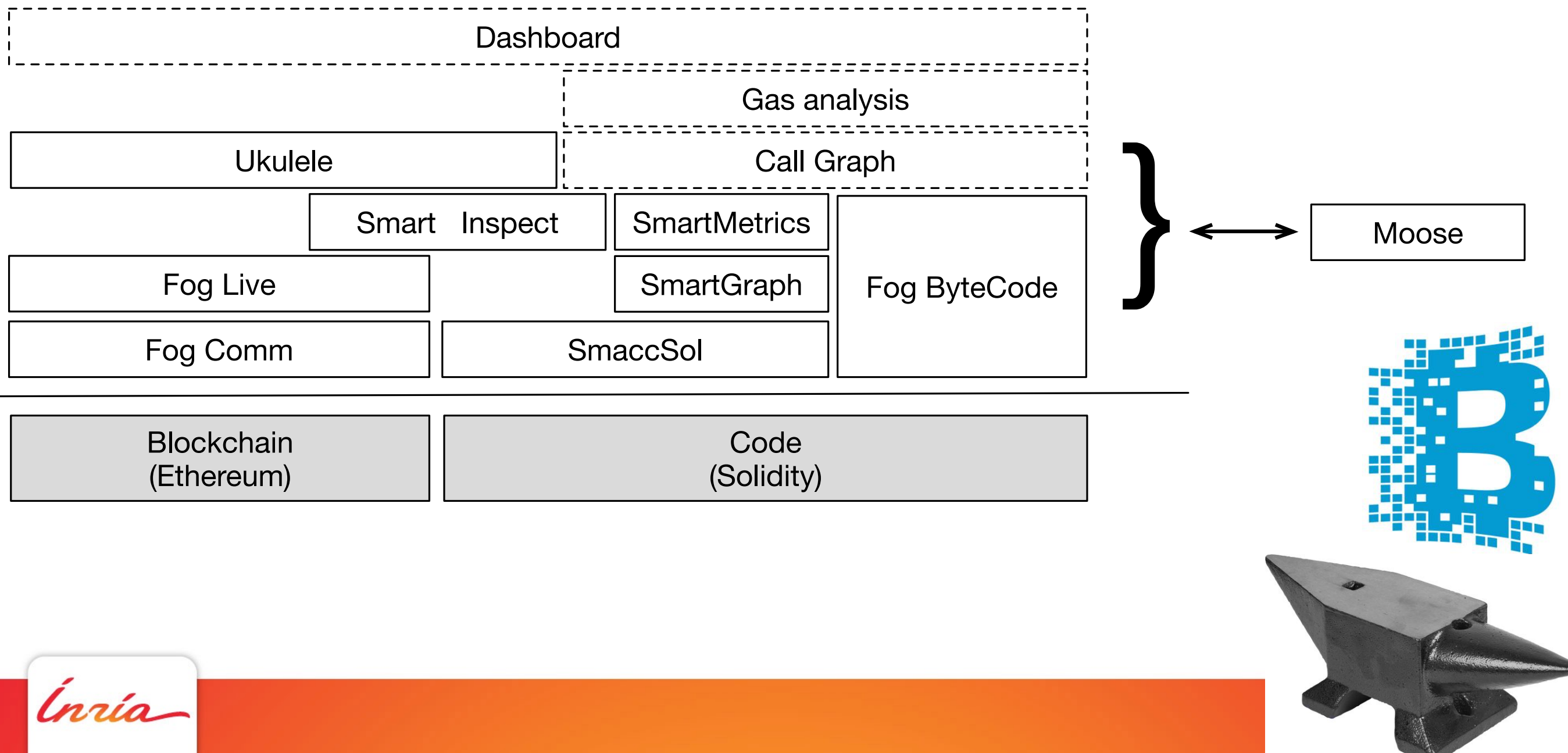
How to capture and model existing process?

How to model and simulate domain of trust and possible architecture?

What are the tools for business blockchain engineers?

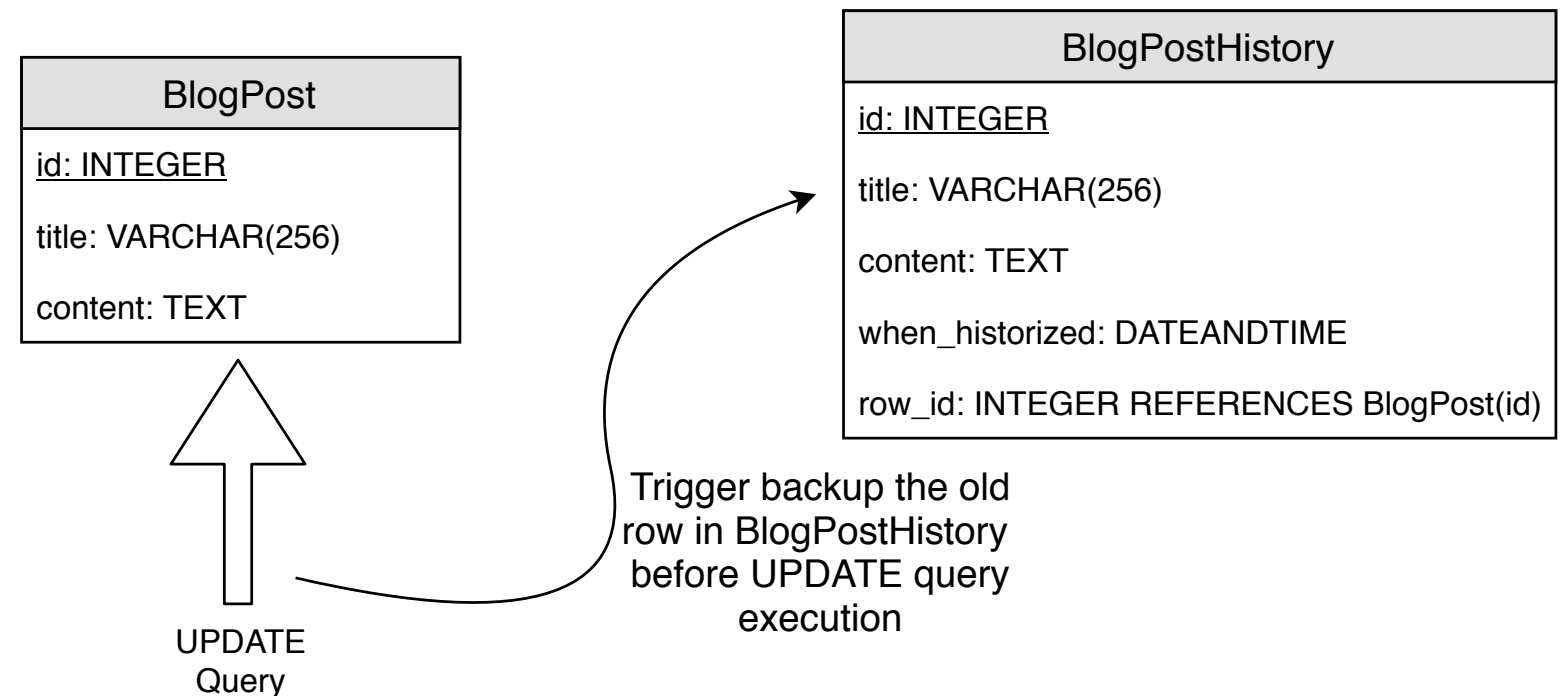


# SmartAnvil: Tools for SE-Blockchains



# And embedded procedures?

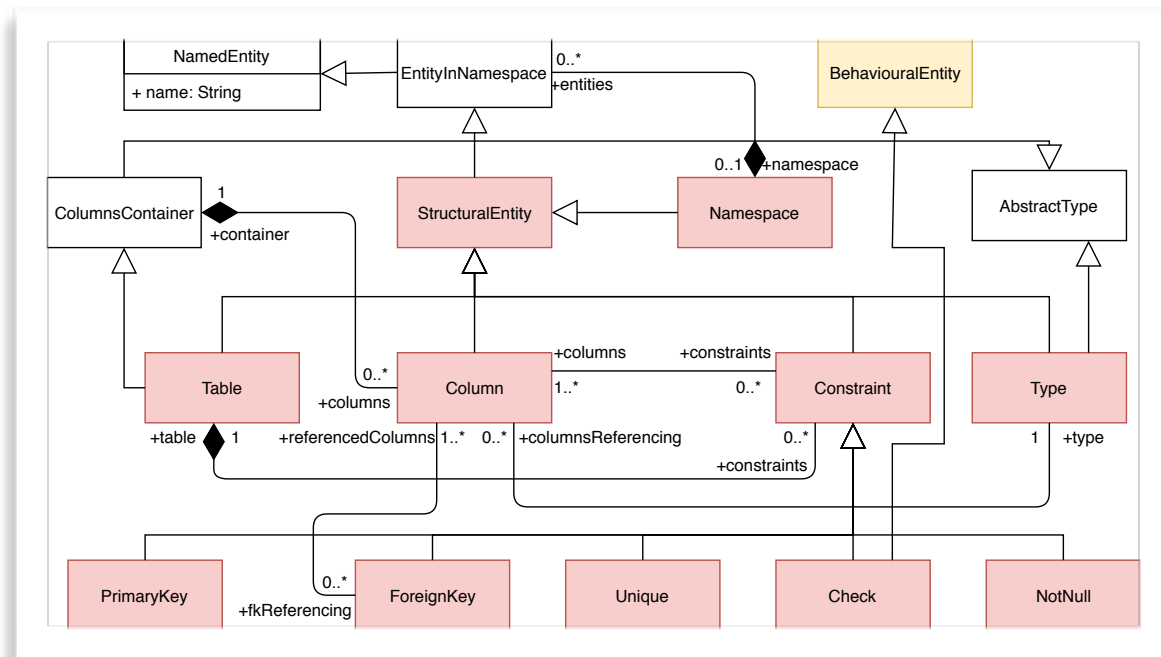
## [PhD J. Delplanque]



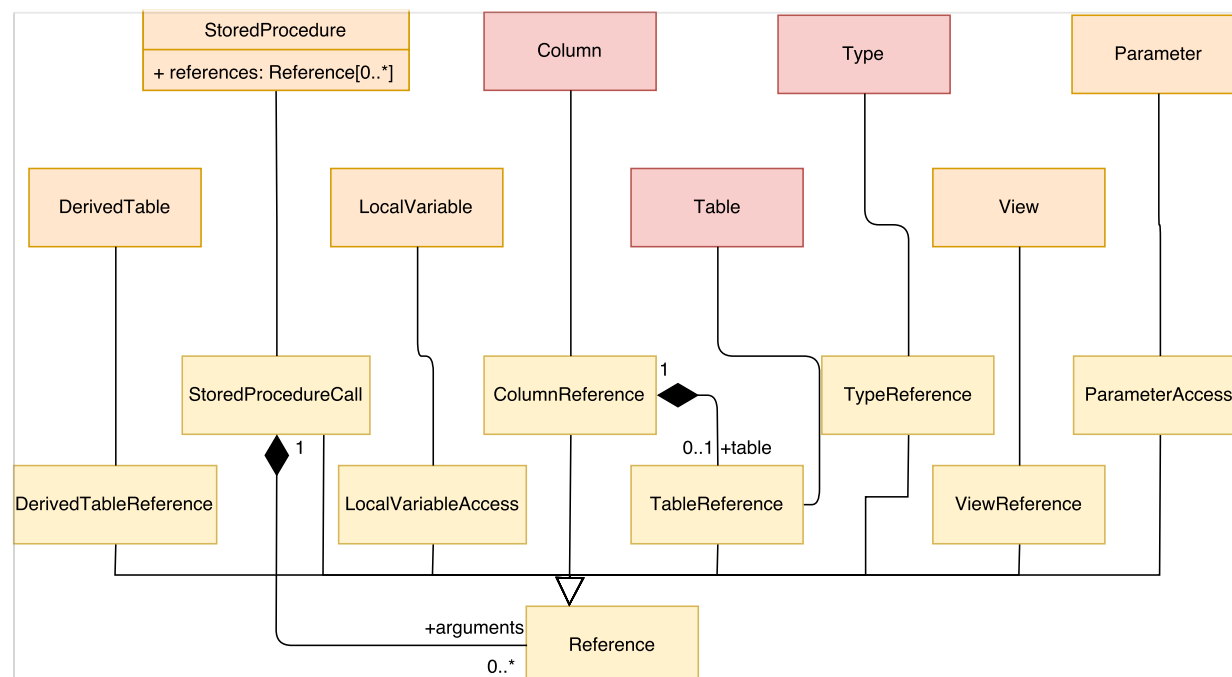
- **Dependency Analysis** — *How to know which entities potentially need to be changed when an entity is changed?*
- **Recommandations** — *How to adapt an entity to a change on a entity it depends on?*
- **Embedded Behaviour Management** — *How to keep the behaviour embedded inside the database working after a change?*

# “Refactorings” for Embedded procedures

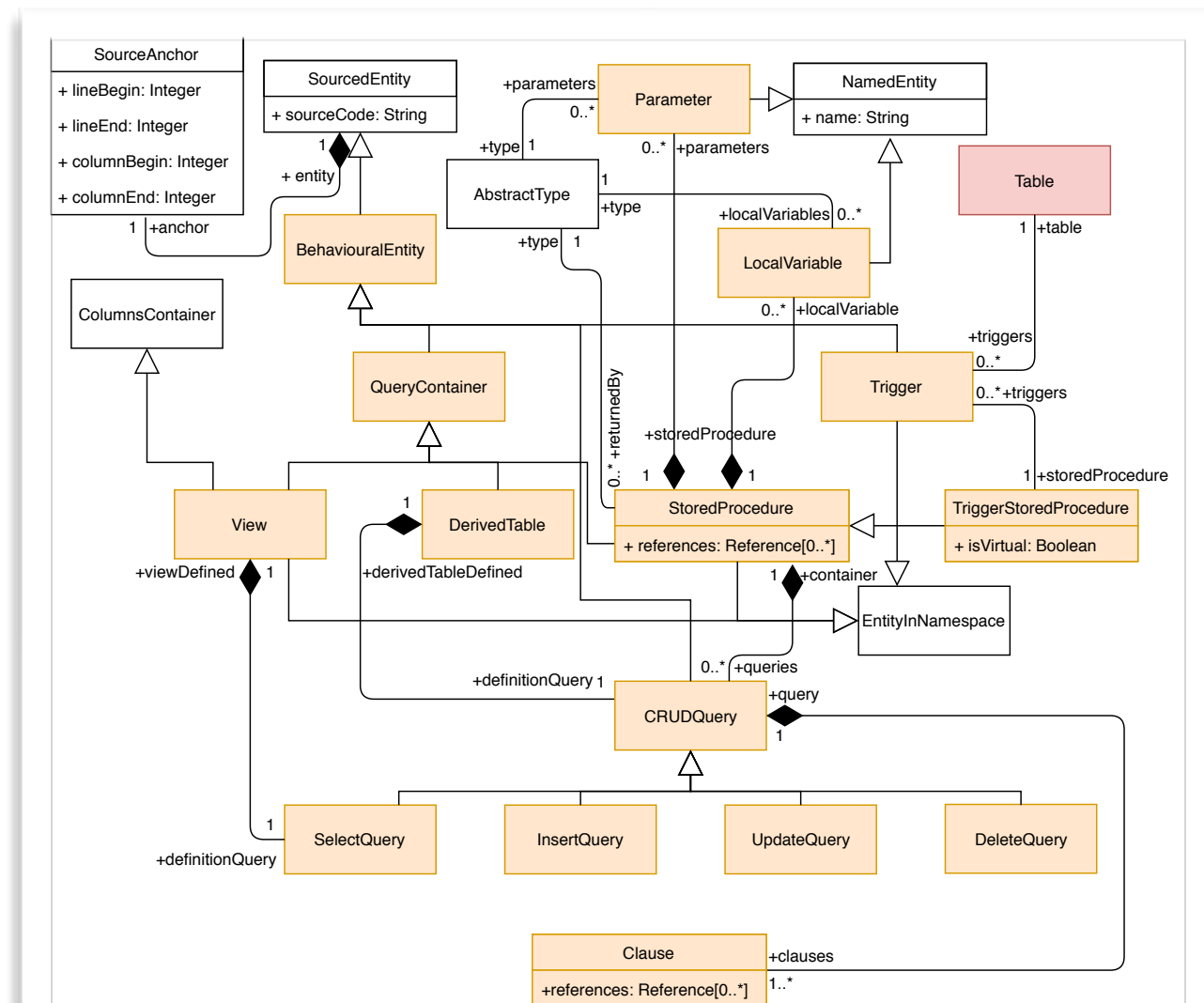
## SQL meta-model



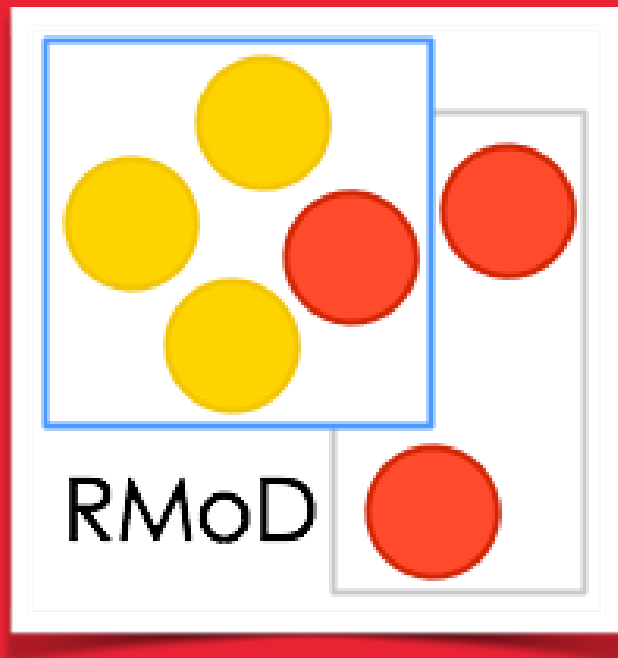
Structural Entities



References Entities



Behavioural Entities



# Recent works



# [ICSE'19] ROTTEN GREEN TEST IS...

---

- A test *passing* (*green*)
- A test that contains at least one *assertion*
- One or more assertions is *not* executed when test runs

# [ICSE'19] A LITTLE SKETCH OF A ROTTEN GREEN TEST

---

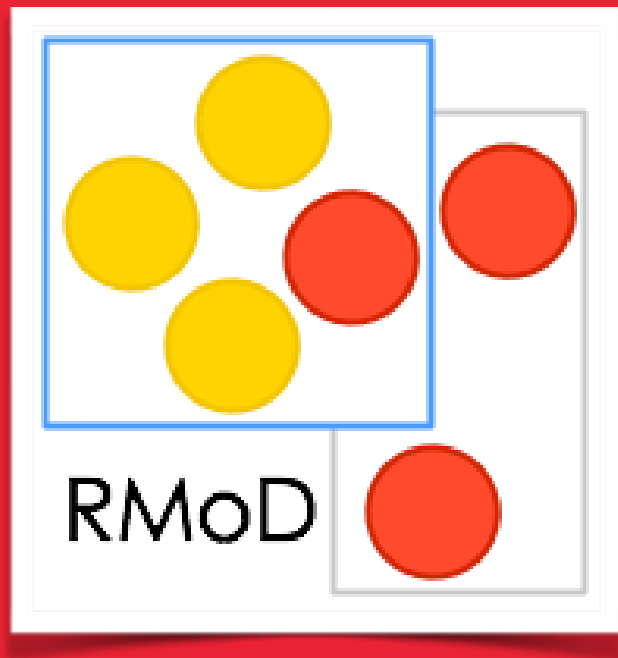
```
class RottenTest {  
    method testABC {  
        if (false) then {self.assert(x)}  
    }  
}
```

# [ICSE'19] THEY DO EXIST

---

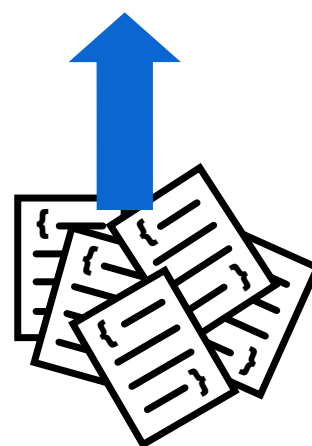
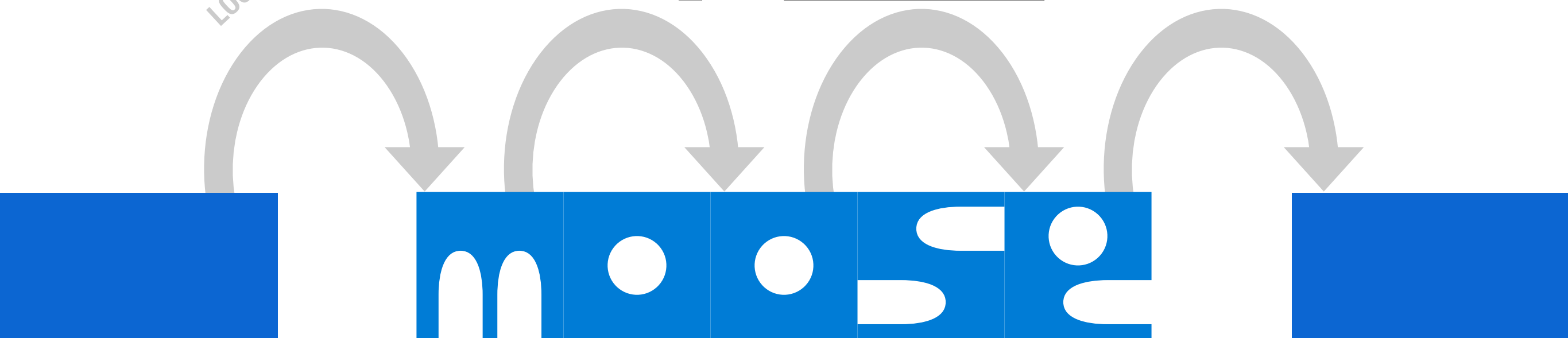
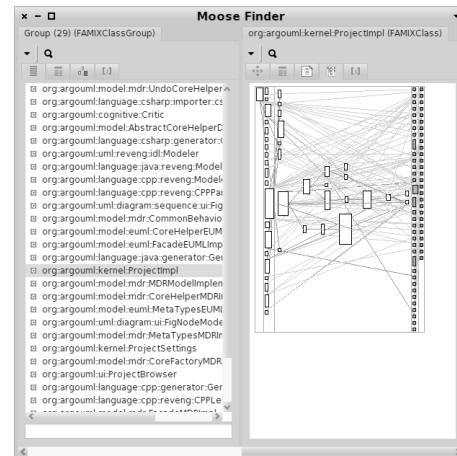
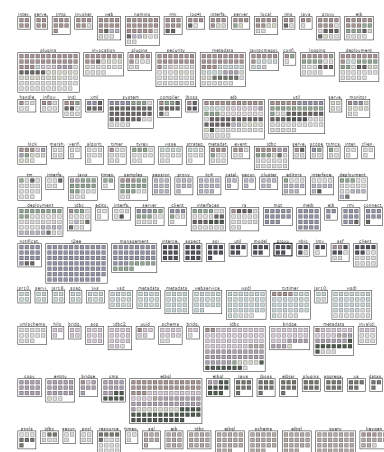
Found originally in Pharo

Already found some in Java



# Stepping back

classes select: #isGod  
McCabe = 21  
LOC = 753,000



# Encouraging feedback loop

*a real problem  
+ an idea*

Data →

Model



Analysis

*a validated better idea  
and many more new ones*





# New Generation Moose

New metamodel generation

- Designing/Reusing traits
- Composing metamodels out of traits

Now

- Working on new IDE
- New tools
- You can join and have fun with us



# Ready to collaborate

Interested by challenges

- migration help
- sorting bugs
- assessment
- rule extraction
- software map
- rearchitecturing
- service/micro service identification
- blockchainisation :)