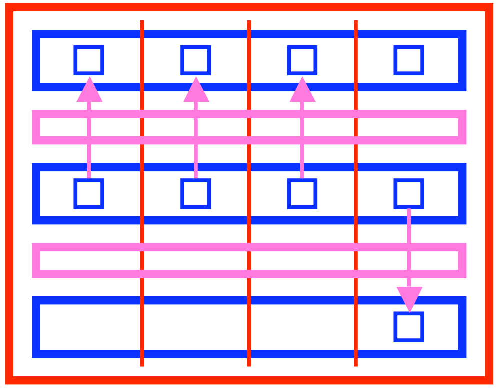
Software Wartung und Evolution

Modeling History with Metamodels

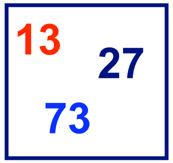
Harald Gall
Institut für Informatik
Universität Zürich
http://seal.ifi.unizh.ch

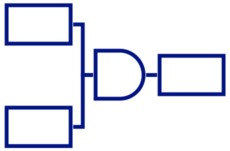
Universität Zürich

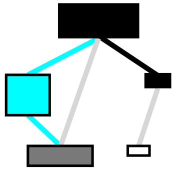
s.e.a.l.
software evolution & architecture lab

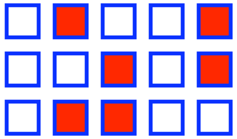


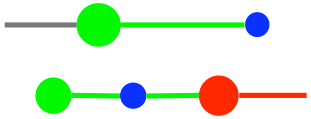
Modeling History to Understand Software Evolution



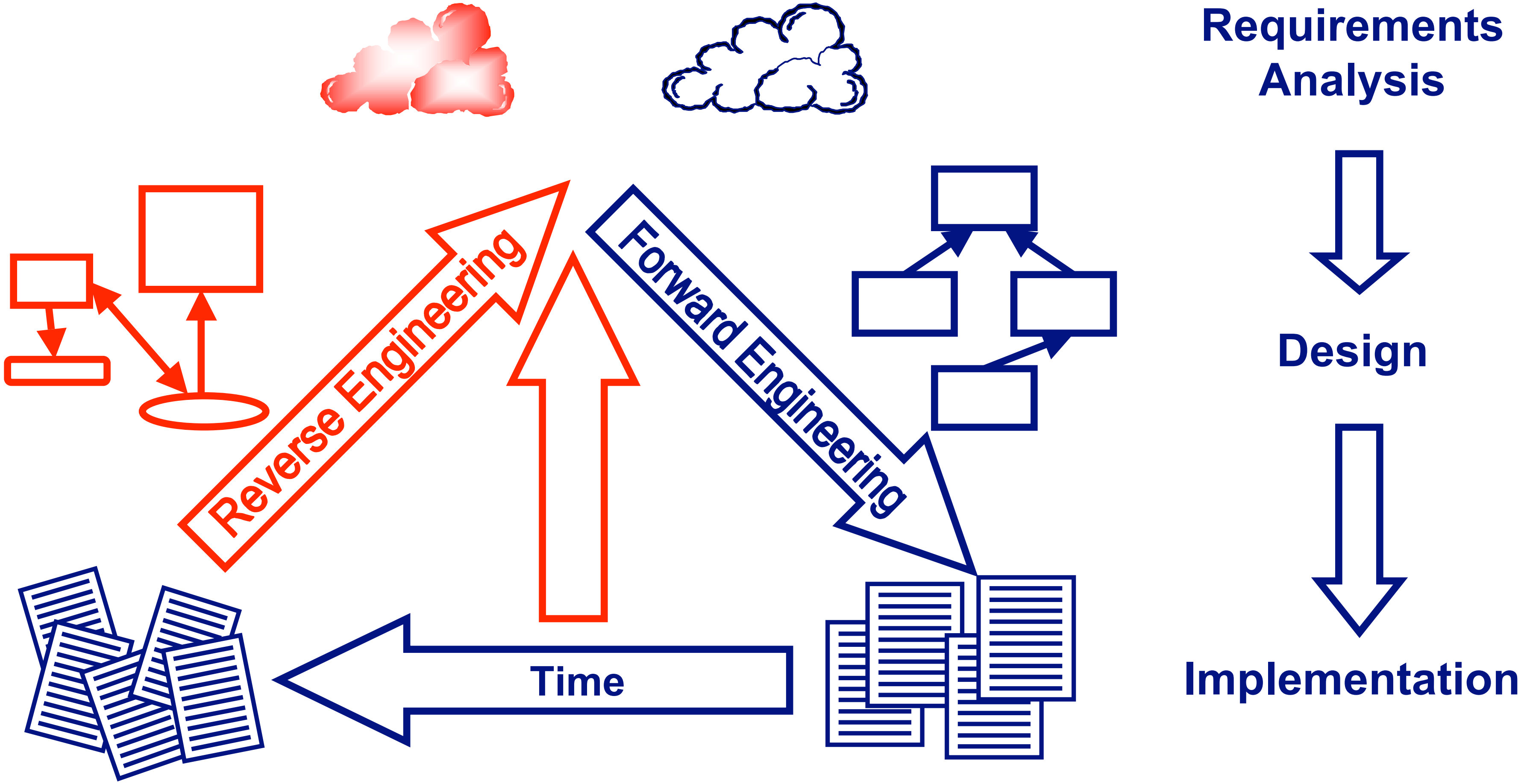








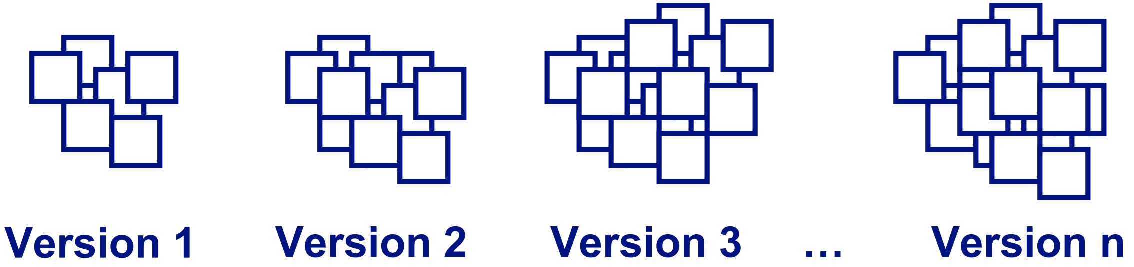
Context: Reverse engineering is creating high level views of the system



Context: History holds useful information for reverse engineering

The doctor always looks at my health file

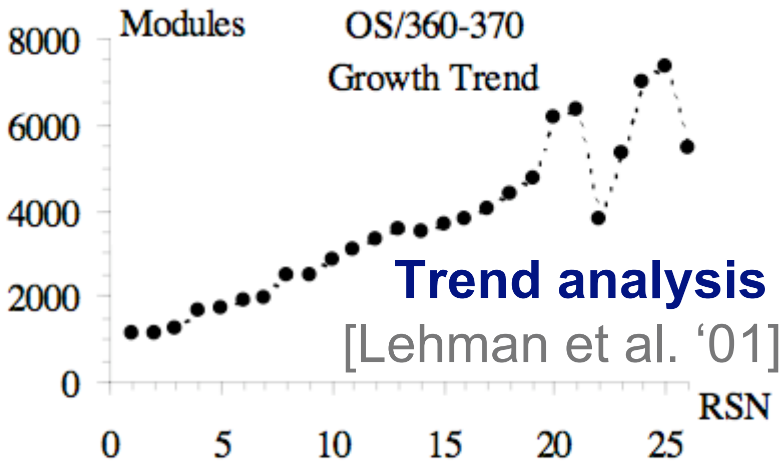
Historical information is useful but, it is hidden among huge amounts of data



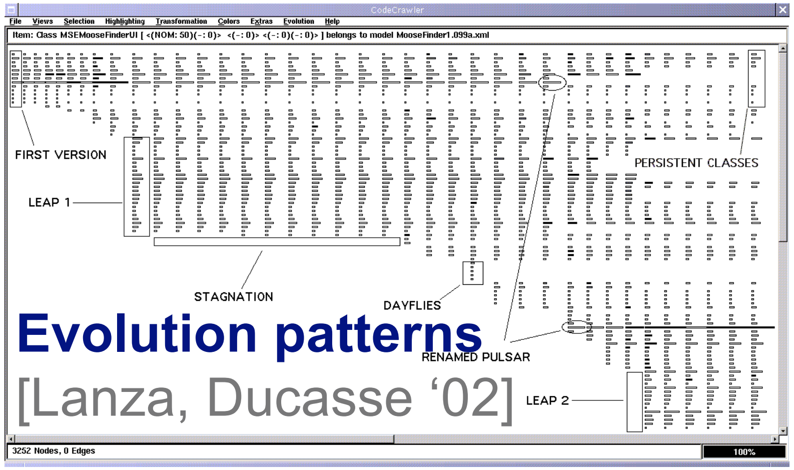
N versions means
N times more data

The more data the more techniques are needed to analyze it

Context: Many techniques were developed



Trend analysis
[Lehman et al. ‘01]

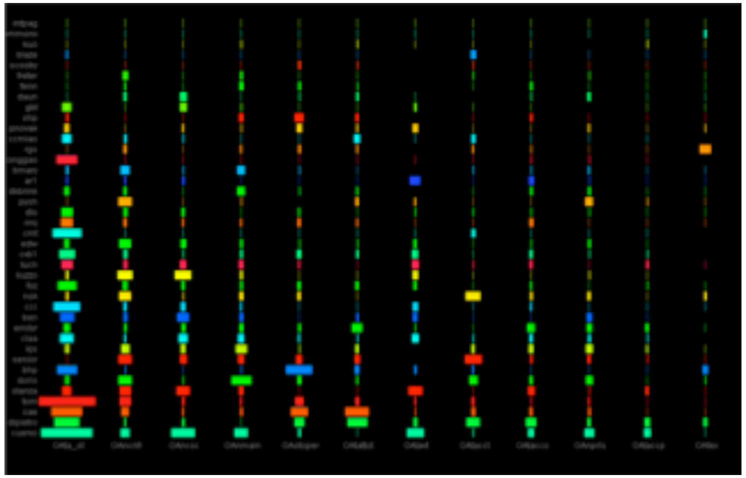


Item: Class MSEMooseFinderUI [<(NOM: 50)(-: 0)> <(-: 0)> <(-: 0)(-: 0)>] belongs to model MooseFinder1.099a.xml

3252 Nodes, 0 Edges

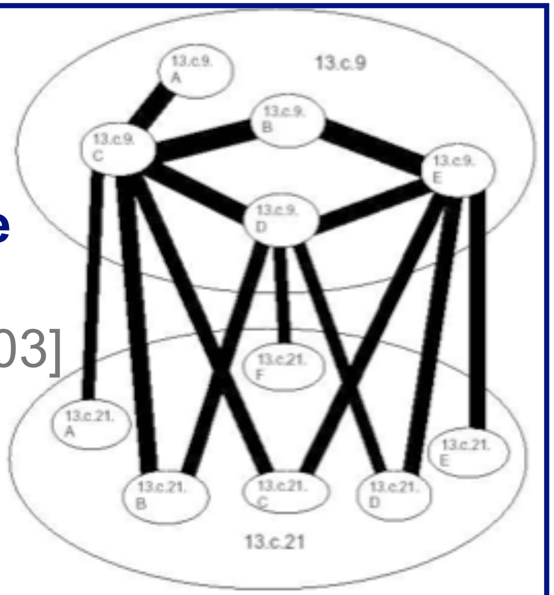
Evolution patterns
[Lanza, Ducasse ‘02]

■ ■ ■

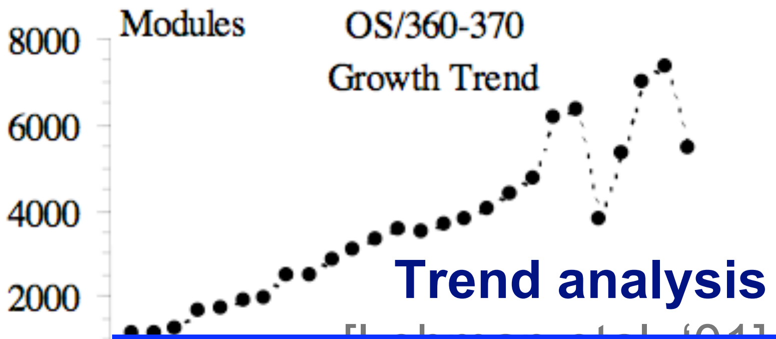


Authors analysis [Eick et al. ‘02]

Co-change analysis
[Gall etal. ‘03]

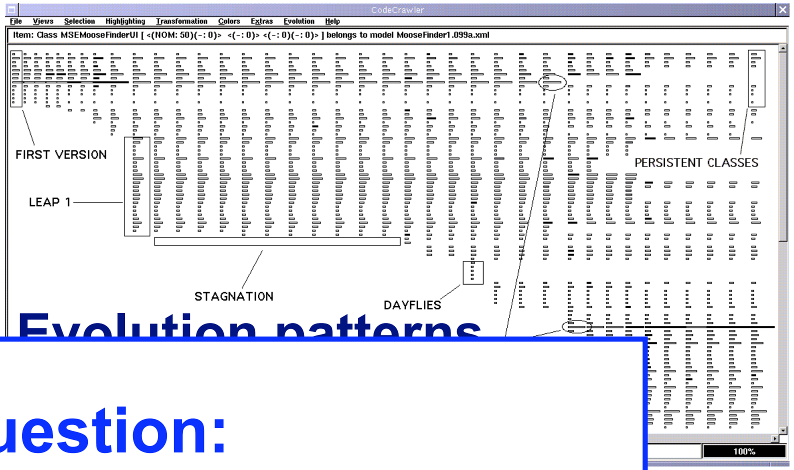


Problem: Current approaches rely on ad-hoc models or on too specific meta-models



Trend analysis

[Lehman etal. '01]

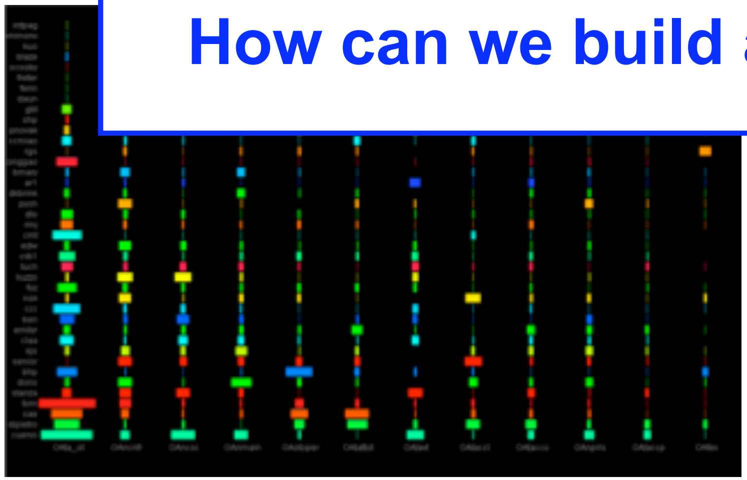


CodeCrawler
File Views Selection Highlighting Transformation Colors Extras Evolution Help
FIRST VERSION
LEAP 1
PERSISTENT CLASSES
STAGNATION
DAYFLIES
100%

Evolution patterns

Research question:

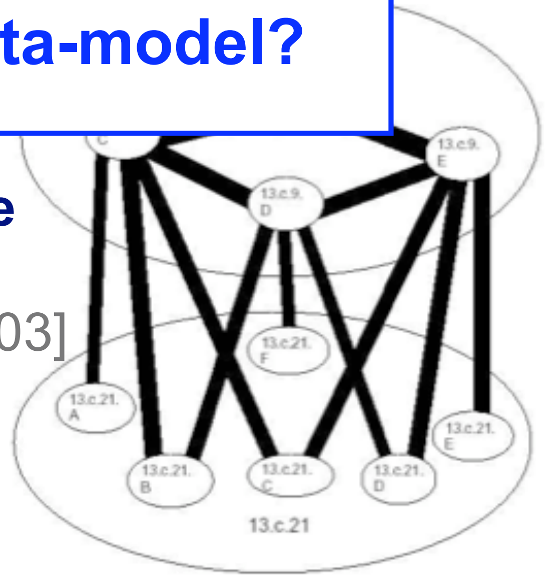
How can we build a generic meta-model?



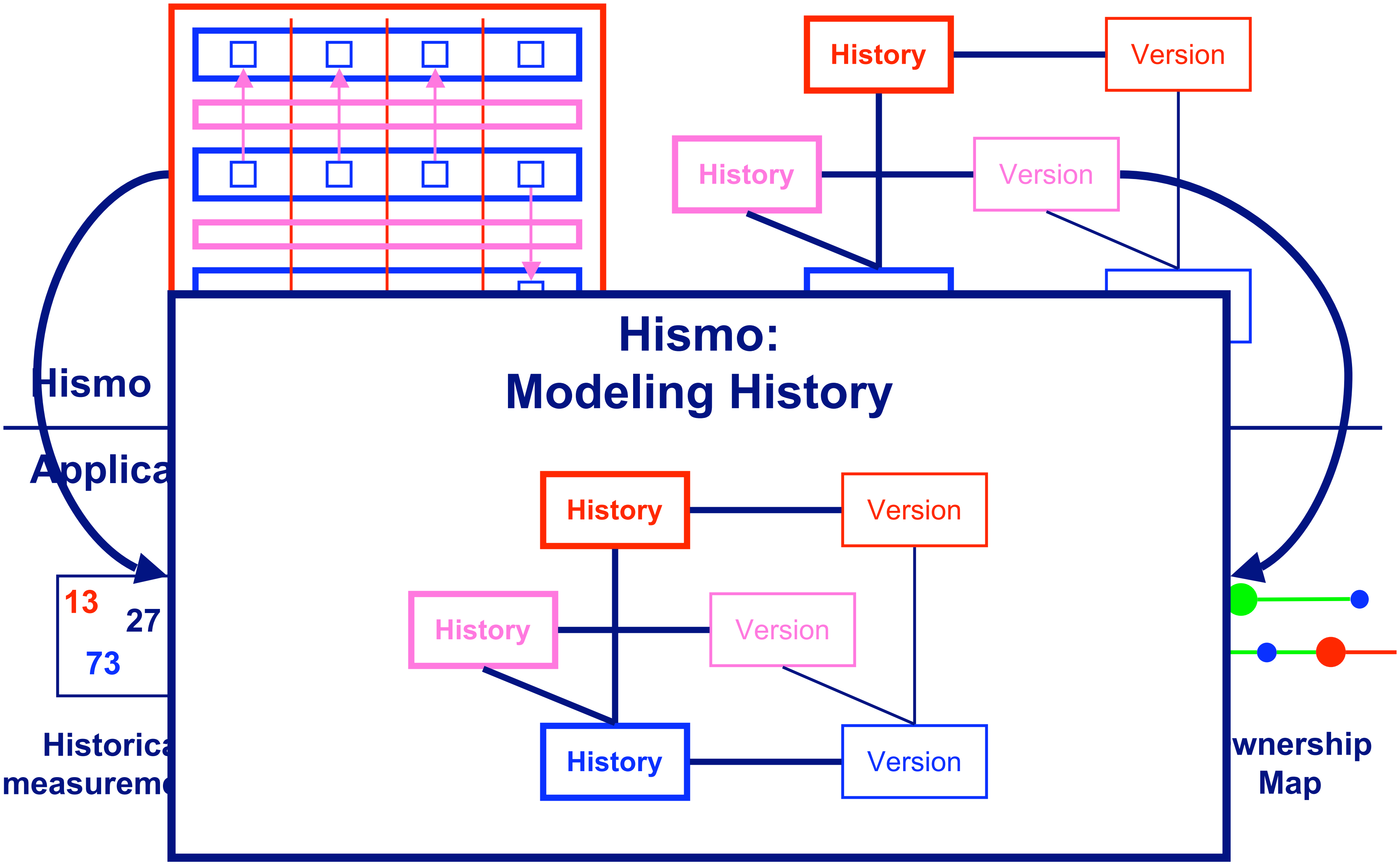
Authors analysis [Eick etal. '02]

Co-change analysis

[Gall etal. '03]



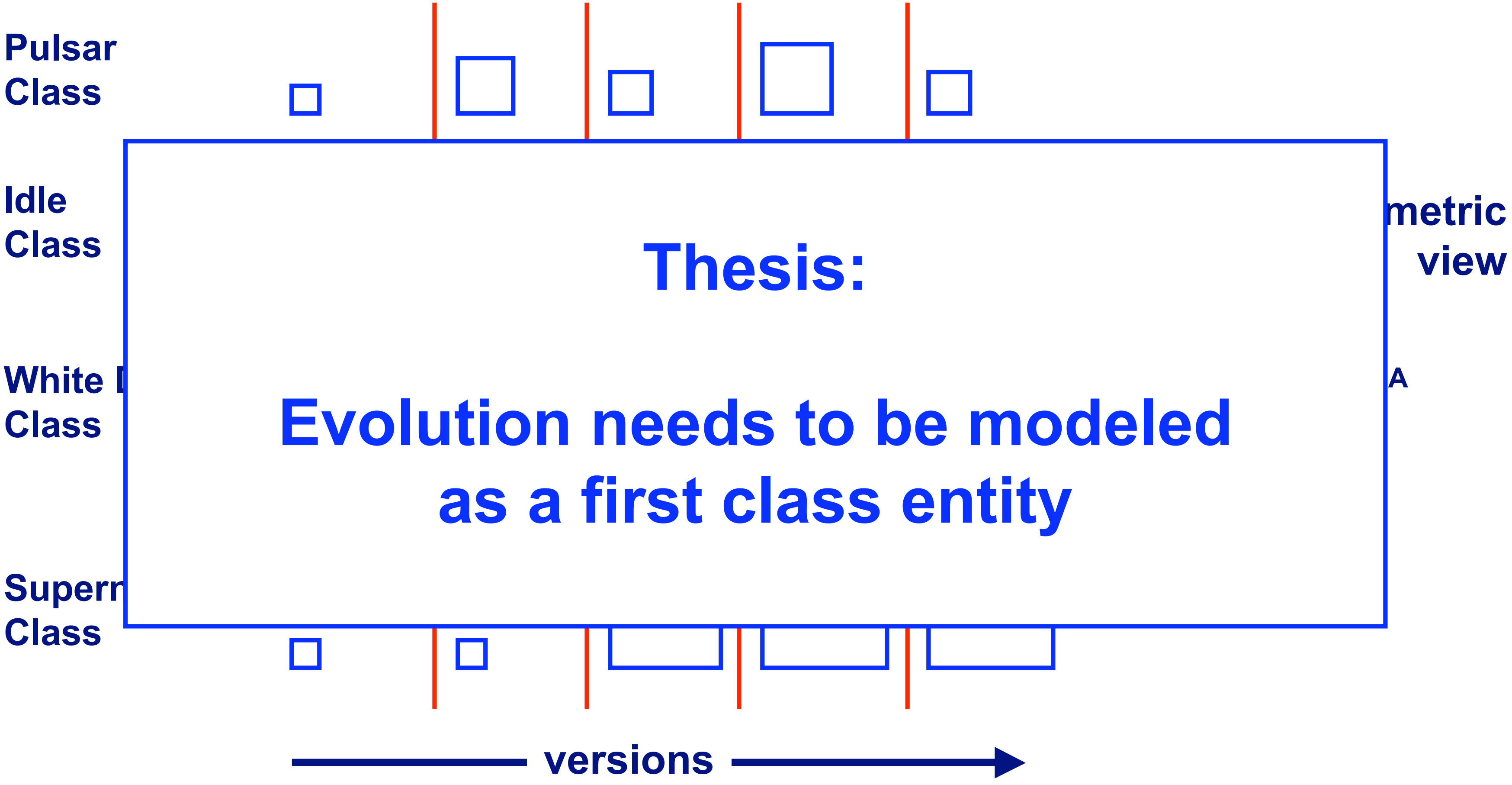
Overview



Hismo: Modeling History

Example: Evolution Matrix reveals different evolution patterns

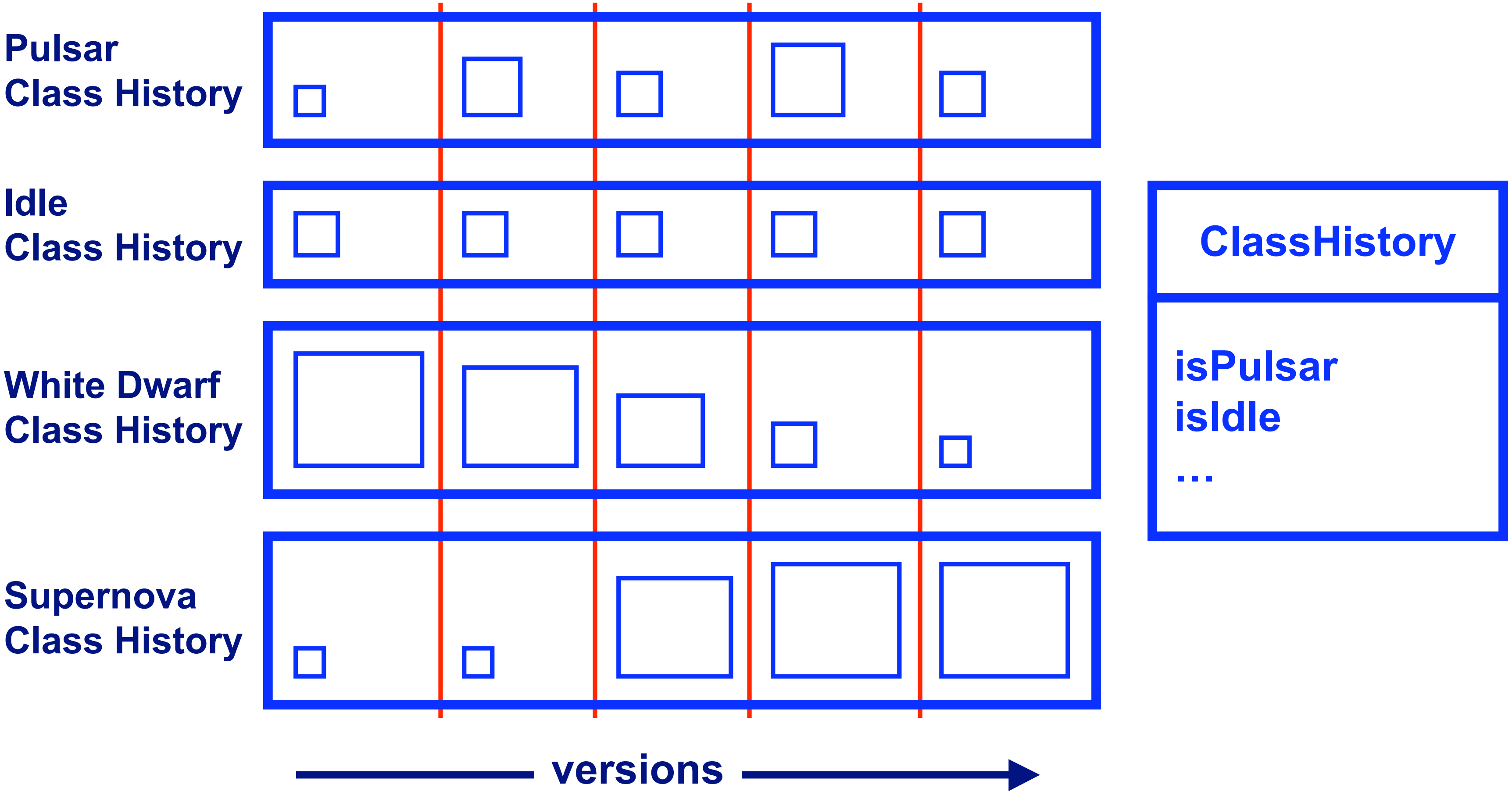
[Lanza, Ducasse '02]



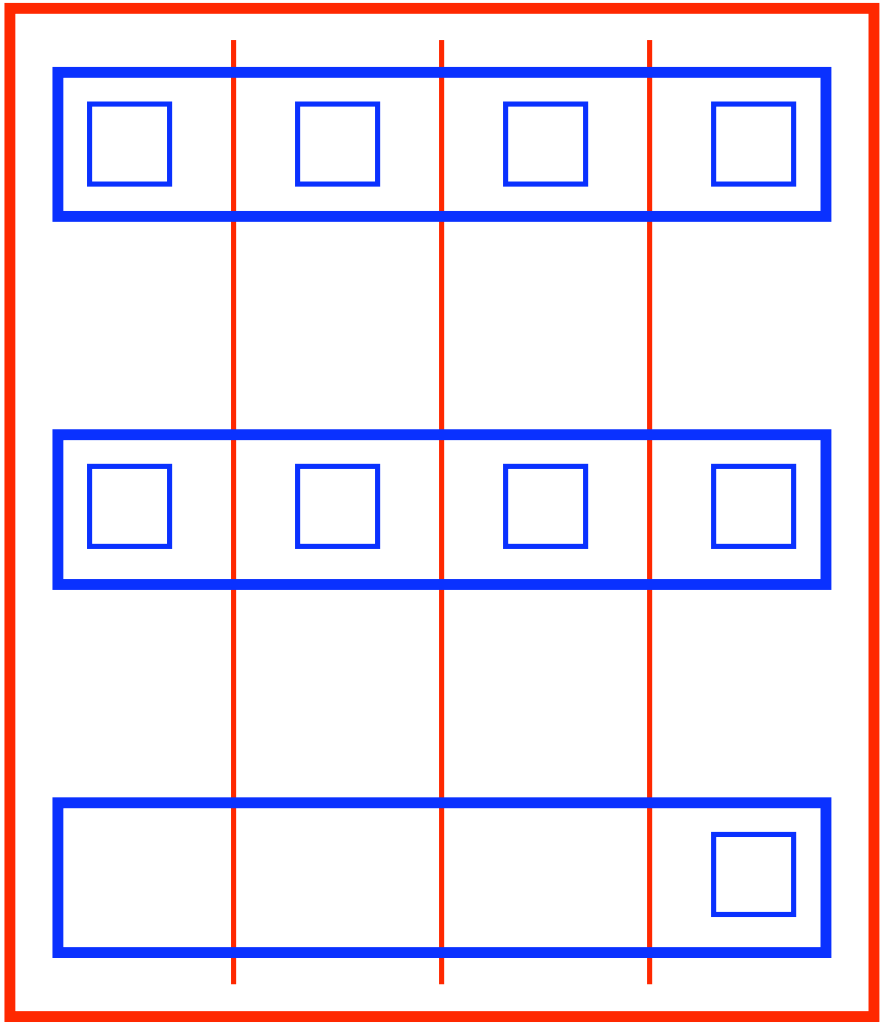
Thesis:

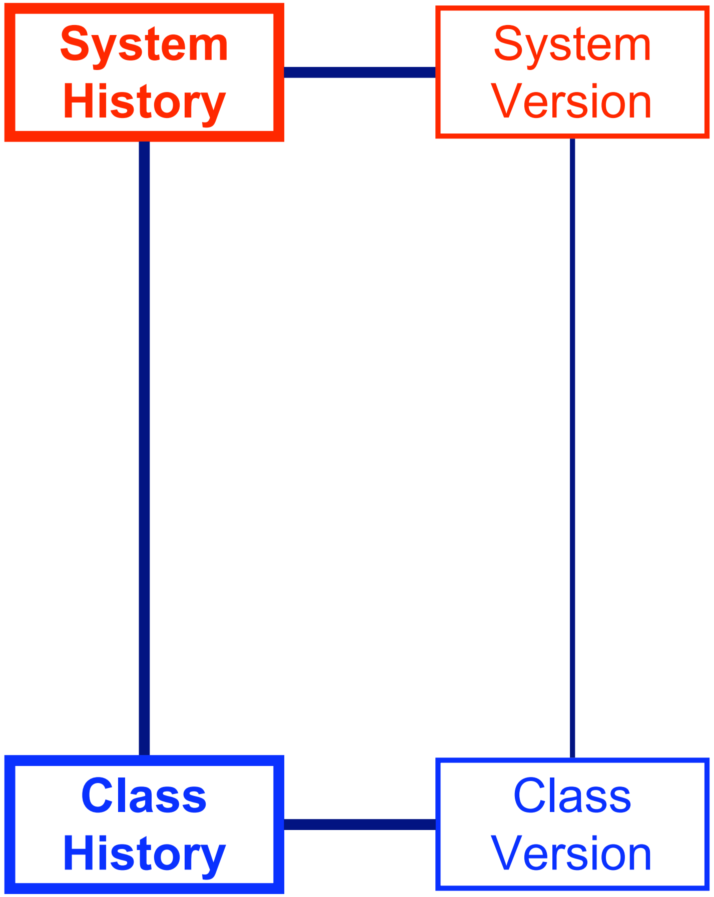
Evolution needs to be modeled as a first class entity

Solution: History encapsulates and characterizes the evolution



Hismo: The history meta-model





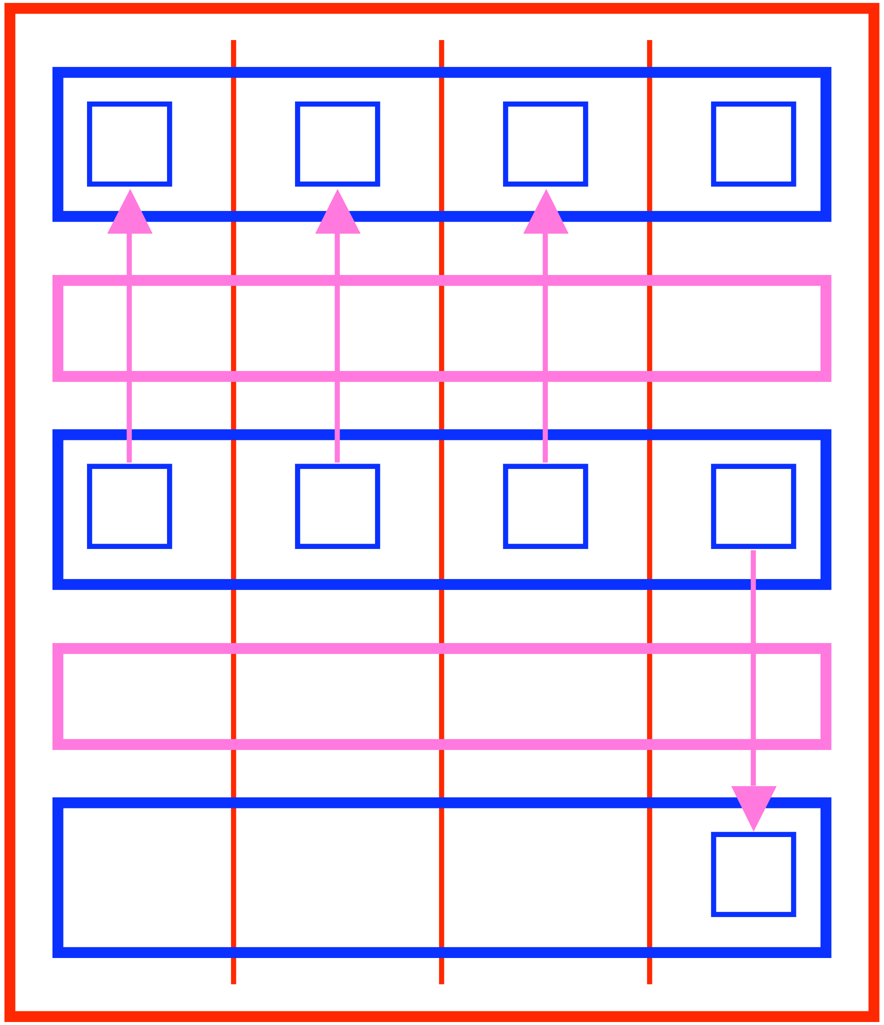
System History

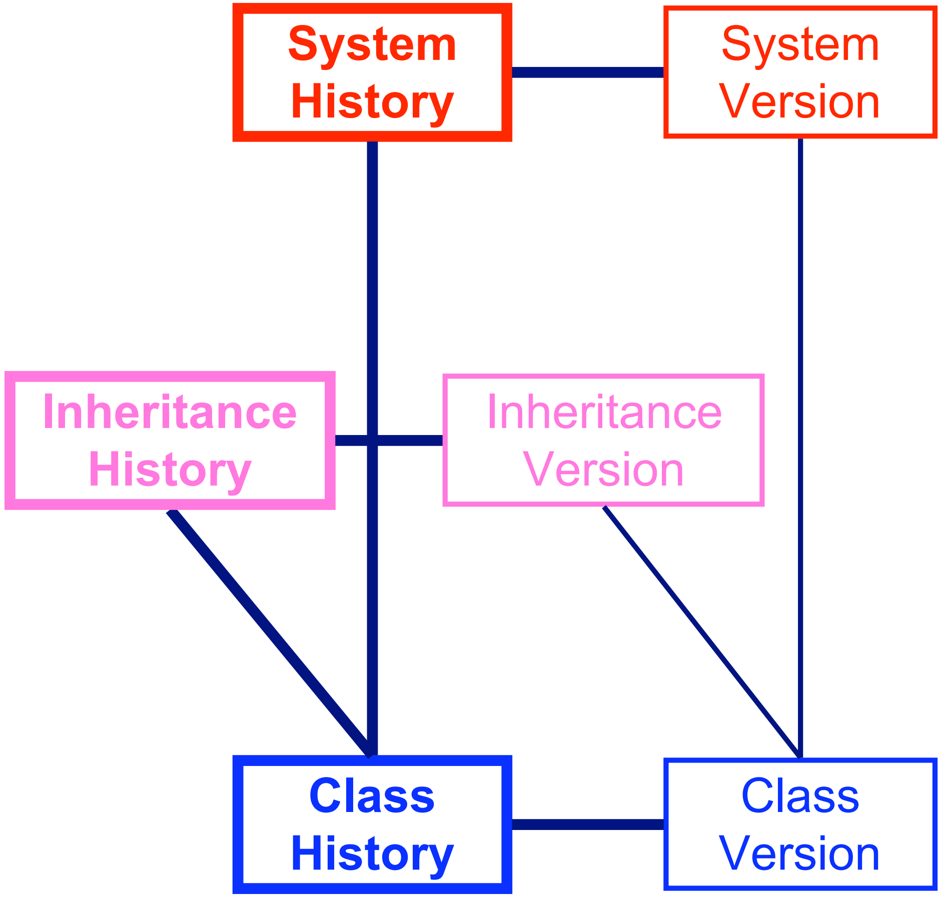
System Version

Class History

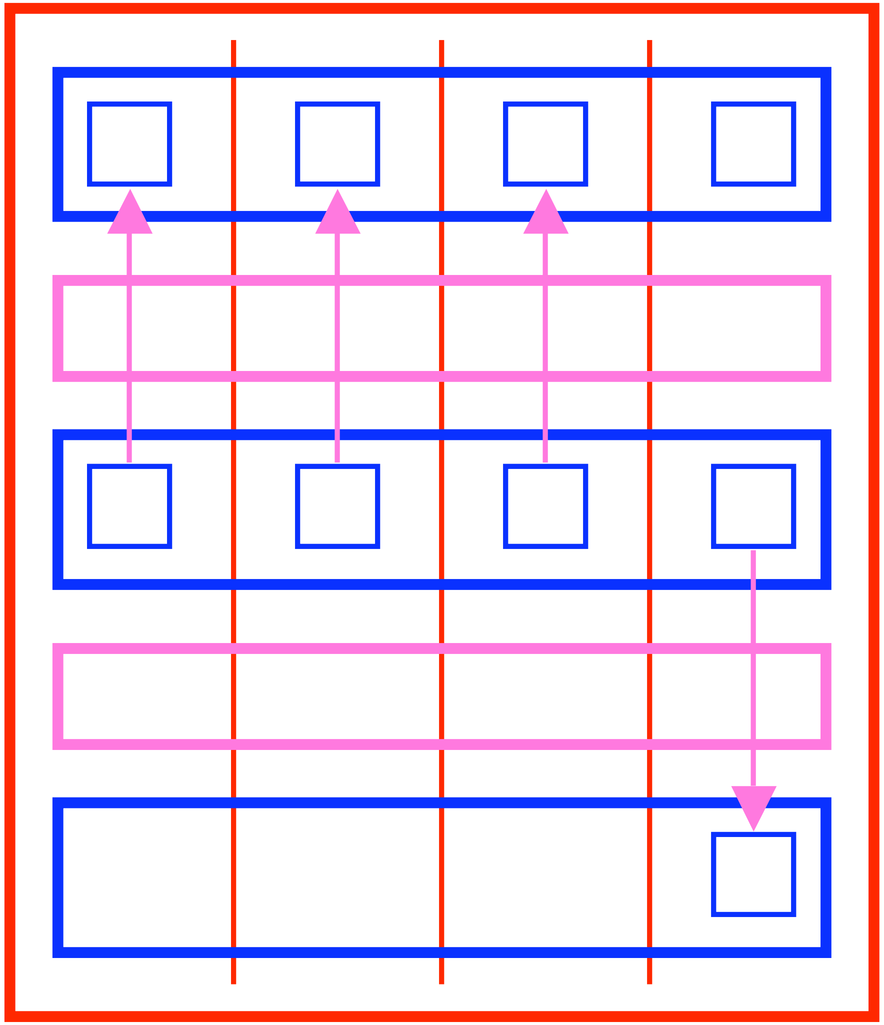
Class Version

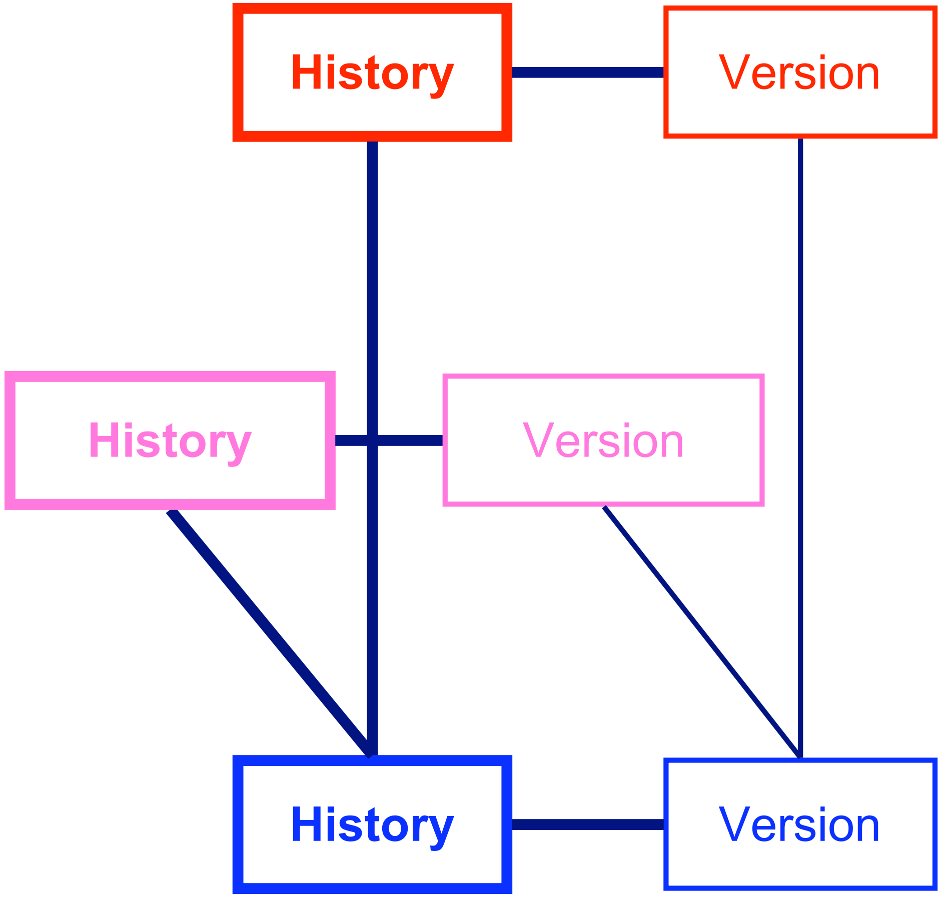
… but, what about relationships?



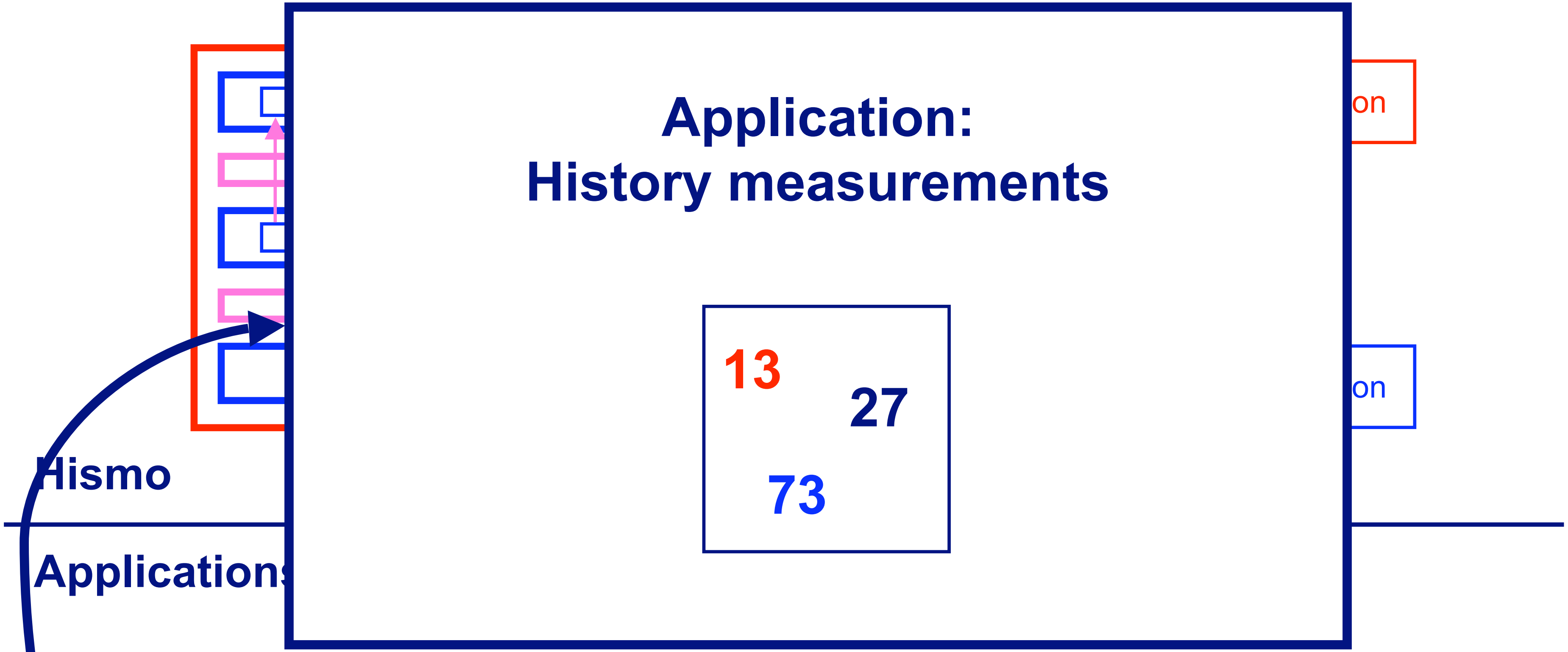


Hismo is obtained by transforming the structural meta-model





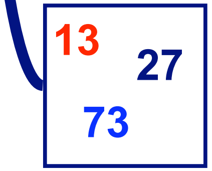
Overview



Application:
History measurements

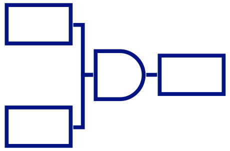
Hismo

Applications

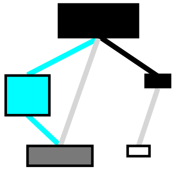


Historical measurements

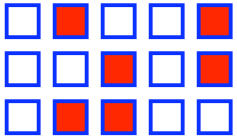
Yesterday’s Weather



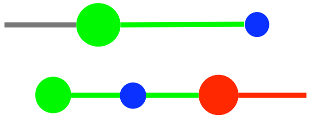
History-based Detection Strategies



Hierarchy evolution

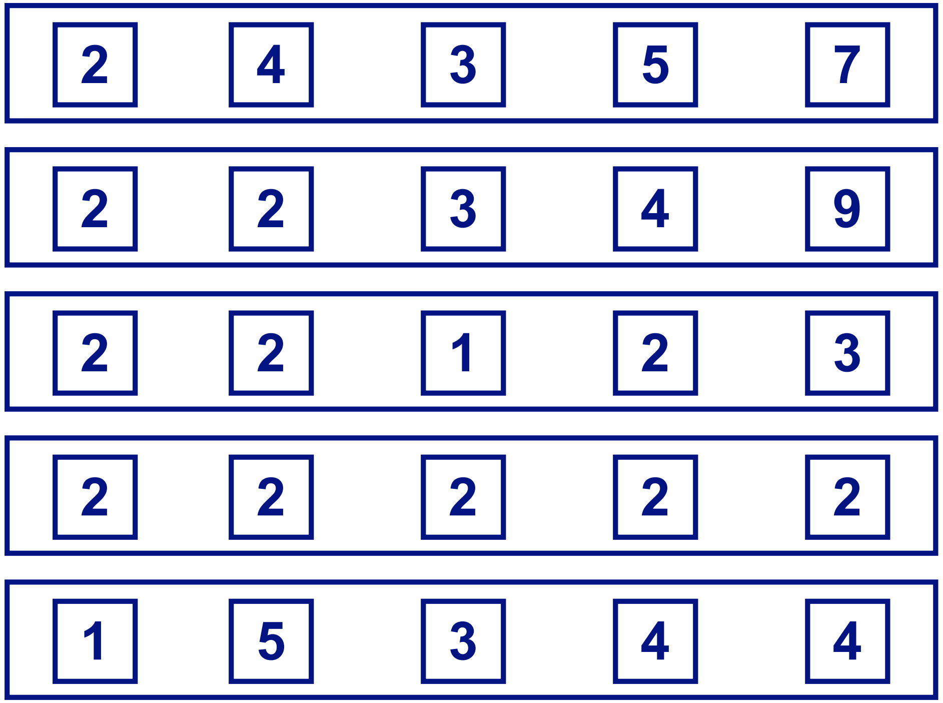


Co-change patterns



Ownership Map

Problem: History holds useful information hidden among large amounts of data



2	4	3	5	7
2	2	3	4	9
2	2	1	2	3
2	2	2	2	2
1	5	3	4	4

How much was a class changed?
When was a class changed?
…

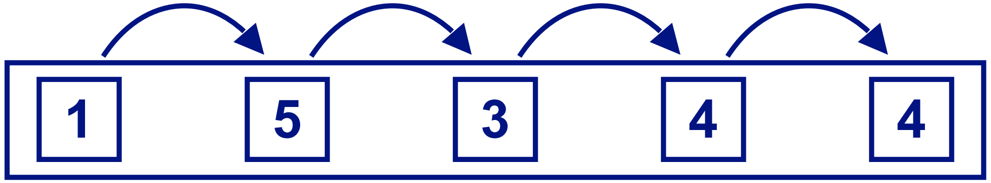
History can be measured:
How much was a class changed?

13 27
73

Evolution of Number of Methods

$$\mathrm{ENOM}(C)=\sum_{i=2}^{n}|\mathrm{NOM}_i(C)-\mathrm{NOM}_{i-1}(C)|$$

$$\mathrm{ENOM}(C)= 4 + 2 + 1 + 0 = 7$$



History can be measured:
When was a class changed?

13 27
73

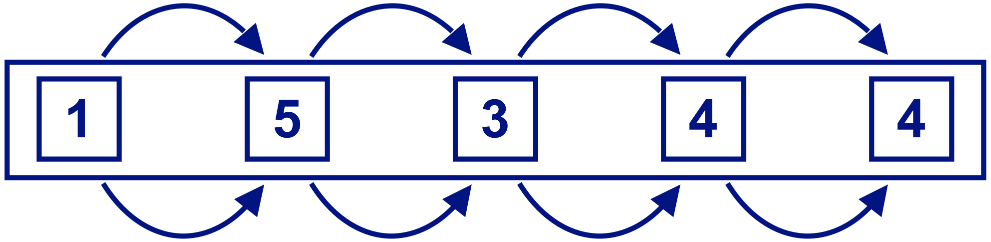
Latest Evolution of Number of Methods

$$LENOM(C) = \sum_{i=2}^{n} |NOM_i(C) - NOM_{i-1}(C)|\ 2^{i-n}$$

Earliest Evolution of Number of Methods

$$EENOM(C) = \sum_{i=2}^{n} |NOM_i(C) - NOM_{i-1}(C)|\ 2^{2-i}$$

$$LENOM(C) = 4\ 2^{-3} + 2\ 2^{-2} + 1\ 2^{-1} + 0\ 2^{0} = 1$$



$$EENOM(C) = 4\ 2^{0} + 2\ 2^{-1} + 1\ 2^{-2} + 0\ 2^{-3} = 5.125$$

History measurements compress aspects of the evolution into numbers

						ENOM	LENOM	EENOM
A	2	4	3	5	7	7	3.37	3.25
B	2	2	3	4	9	7	5.75	1.37
C	2	2	1	2	3	3	1	2
D	2	2	2	2	2	0	0	0
E	1	5	3	4	4	7	1	5.12

History measurements compress aspects of the evolution into numbers

13 27
73

		ENOM	LENOM	EENOM
A	Balanced changer	7	3.37	3.25
B	Late changer	7	5.75	1.37
C		3	1	2
D	Dead stable	0	0	0
E	Early changer	7	1	5.12

Many measurements can be defined at different levels of abstraction …

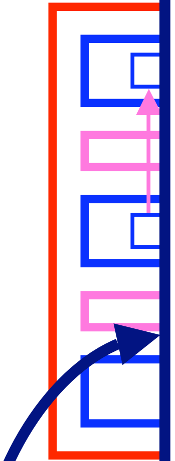
Evolution
Latest/Earliest Evolution
Stability
Historical Max/Min
Historical Average
Growth Trend
…

of

Number of Methods
Number of Statements
Cyclomatic Complexity
Lines of Code
Number of Classes
Number of modules
…

… But measurements are a means not a goal

Overview



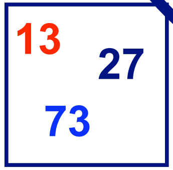
on

on

Application: Yesterday's Weather

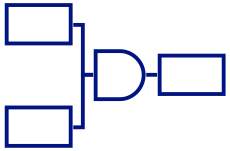
Hismo

Applications

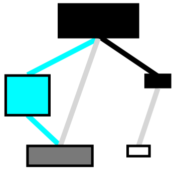


Historical measurements

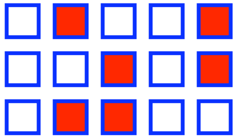
Yesterday's Weather



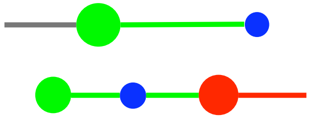
History-based Detection Strategies



Hierarchy evolution



Co-change patterns



Ownership Map

© Tudor Gîrba

Common Wisdom: The recently changed parts are likely to change in the near future

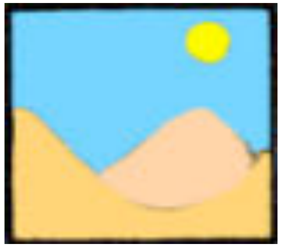
[Mens,Demeyer '01]

Is the common wisdom relevant?

Yesterday's Weather metaphor:

- It expresses the chances of having the same weather today as we had yesterday
- It is location specific

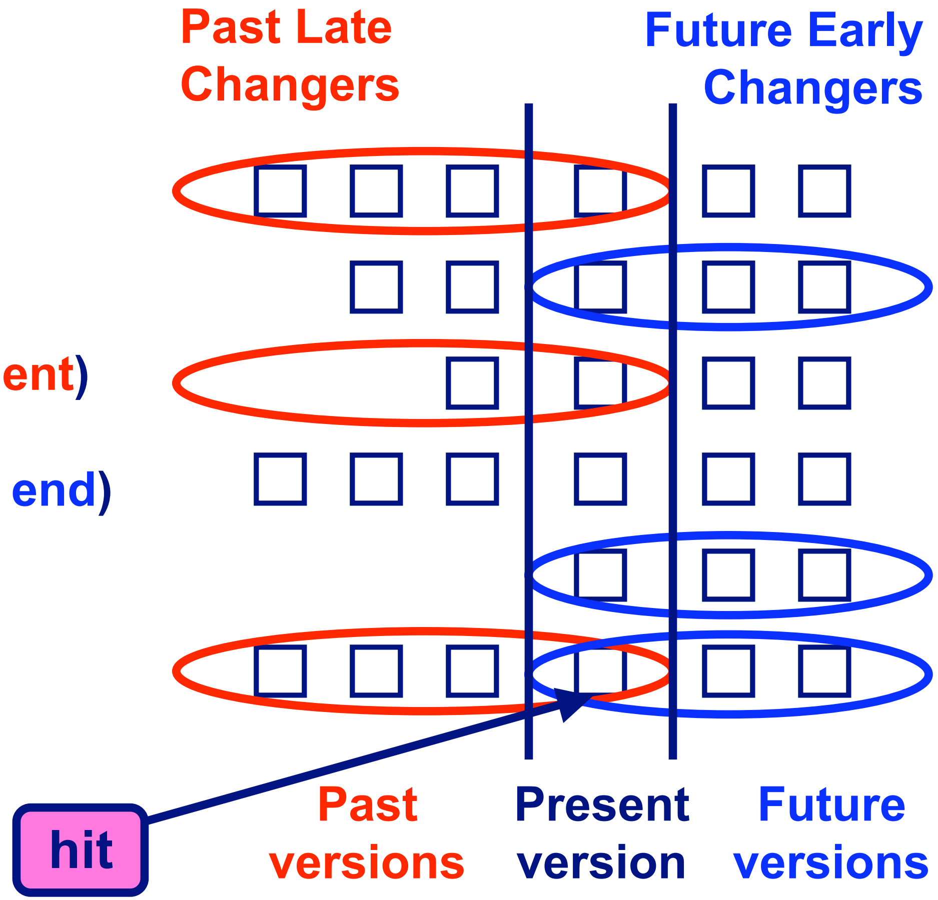
Switzerland - 30%



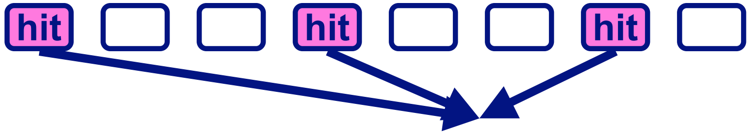
Sahara - 90%

Yesterday’s Weather: For each given version we check the common wisdom

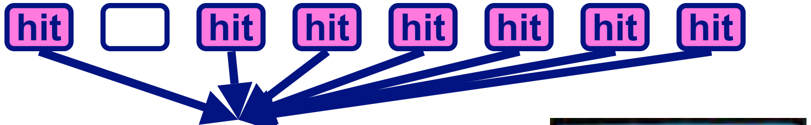
```
YesterdayWeatherHit(present):
  past:=histories.topLENOM(start, present)
  future:=histories.topEENOM(present, end)
  past.intersectWith(future).notEmpty()
```



Overall Yesterday’s Weather shows the localization of changes in time



$$YW = \frac{3\ \text{hits}}{8\ \text{possible hits}} = 37\%$$



$$YW = \frac{7\ \text{hits}}{8\ \text{possible hits}} = 87\%$$

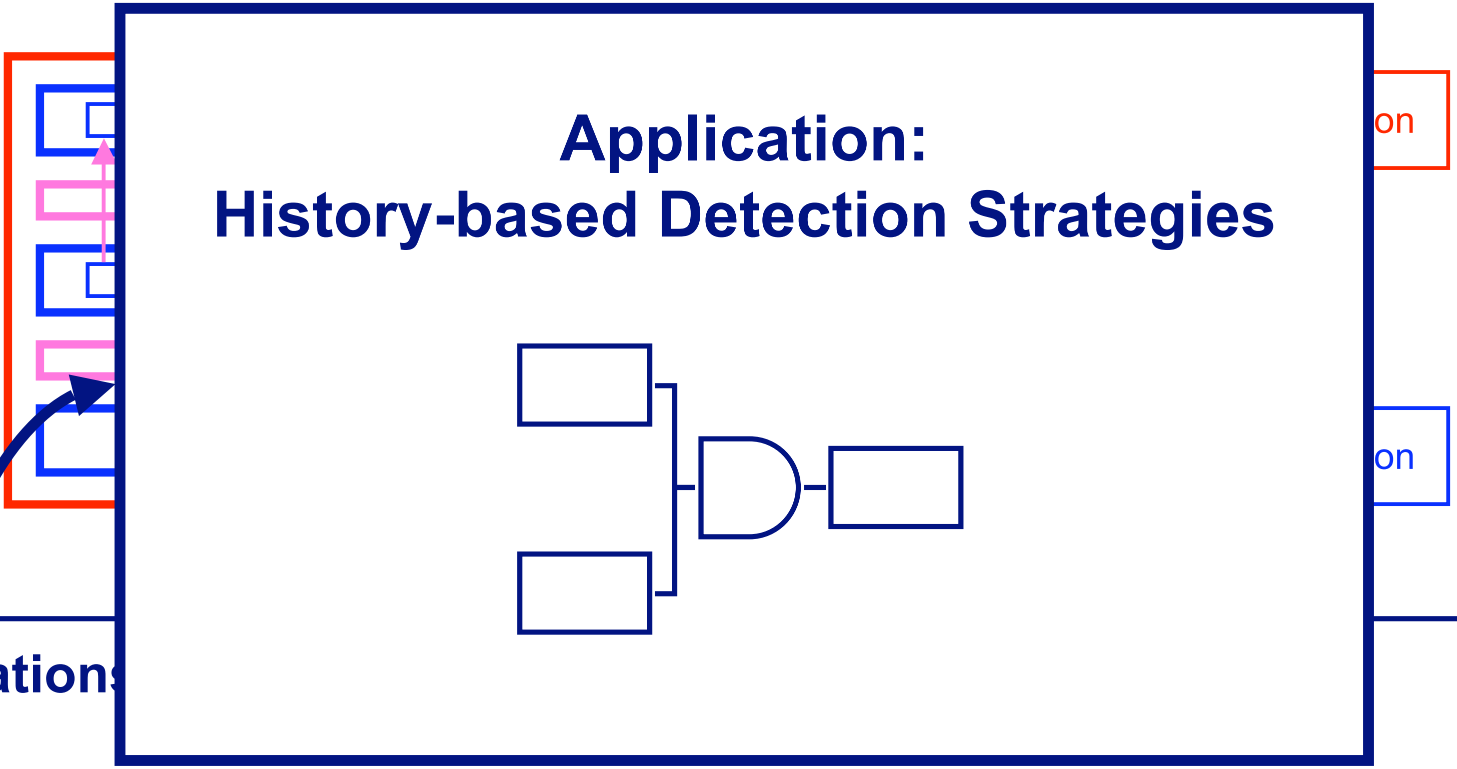
Case studies:

40 versions of CodeCrawler (180 classes): 100%

40 versions of Jun (700 classes): 79%

40 versions of Jboss (4000 classes): 53%

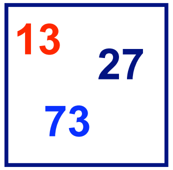
Overview



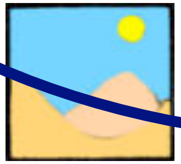
Application:
History-based Detection Strategies

Hismo

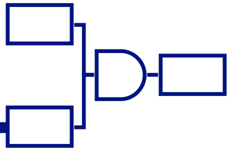
Applications



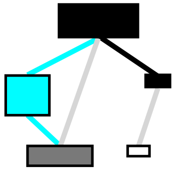
Historical measurements



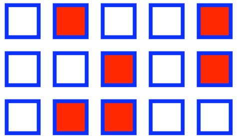
Yesterday’s Weather



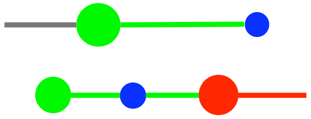
History-based Detection Strategies



Hierarchy evolution



Co-change patterns

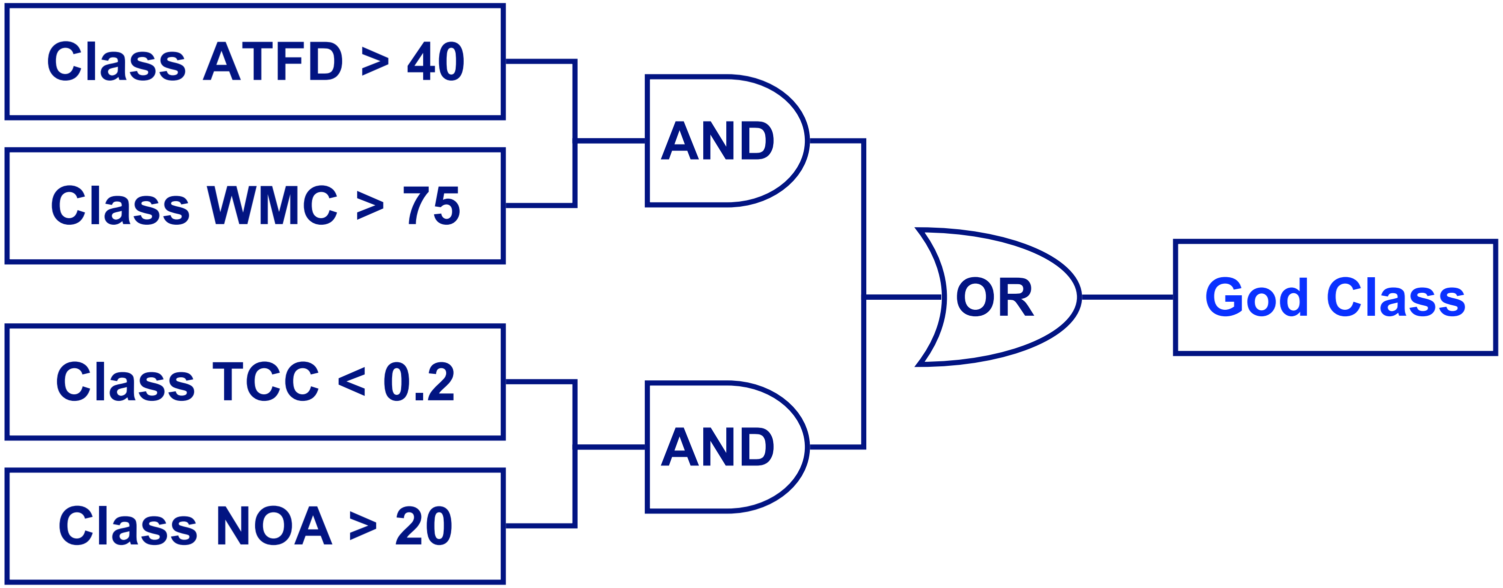


Ownership Map

Context: Detection Strategies detect design flaws based on measurements [Marinescu '04]

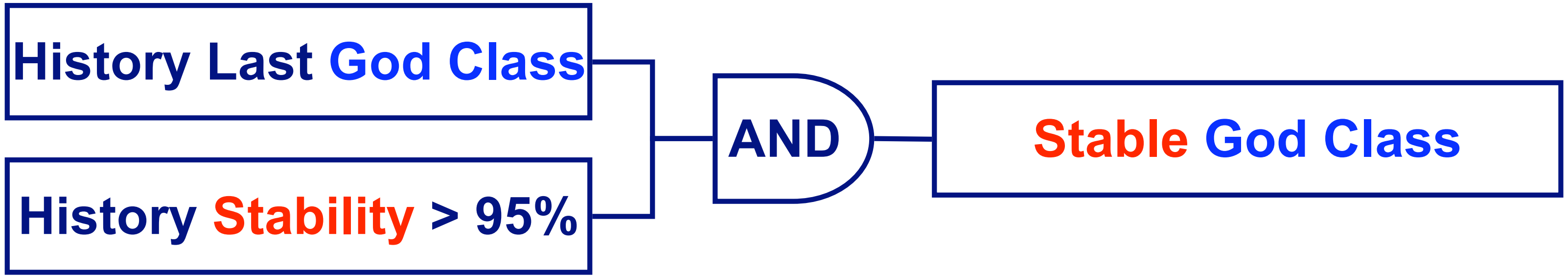
Example: God Class

Maintainability problem because it encapsulates a lot of knowledge



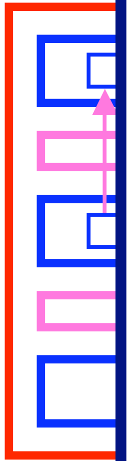
History-based Detection Strategies take evolution into account

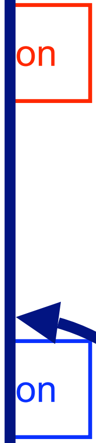
Example: a Stable God Class is not necessarily a bad one



Case study: 5 out of 24 God Classes in Jun were stable and harmless

Overview

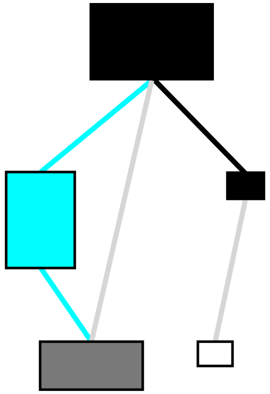


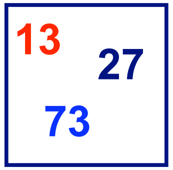


Hismo

Applications

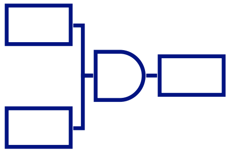
Application: Characterizing the evolution of class hierarchies



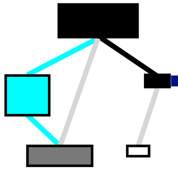


Historical measurements

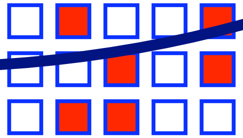
Yesterday’s Weather



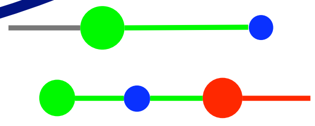
History-based Detection Strategies



Hierarchy evolution

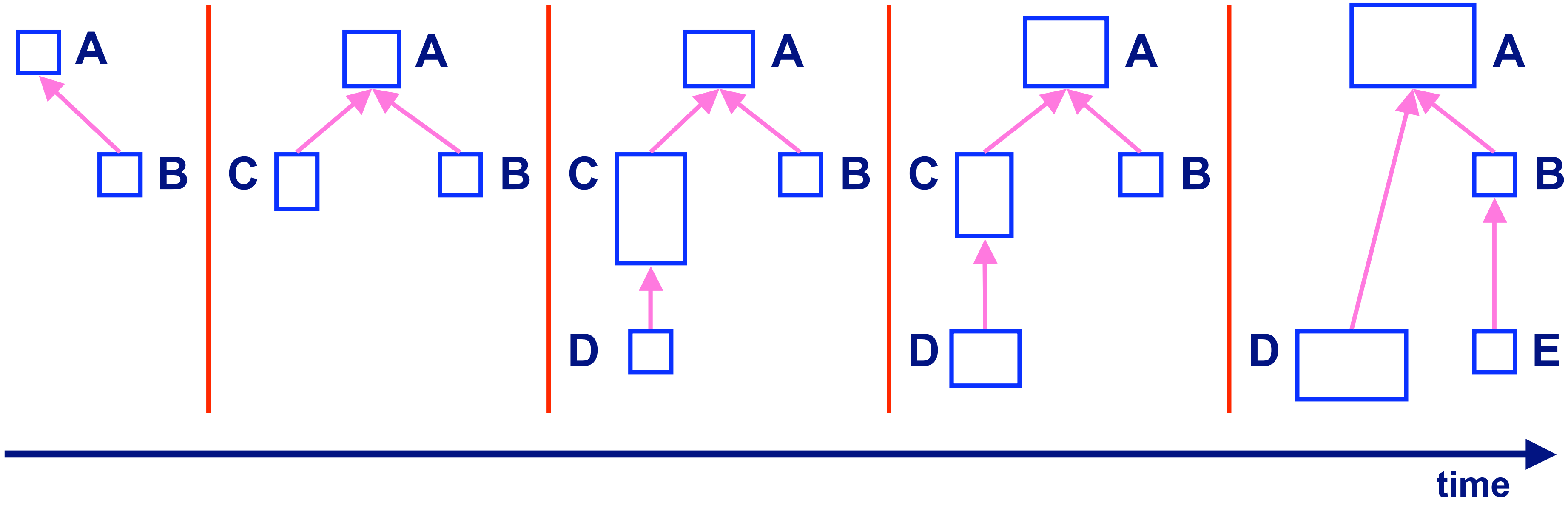


Co-change patterns



Ownership Map

Context: Given the evolution of a hierarchy …



A is persistent

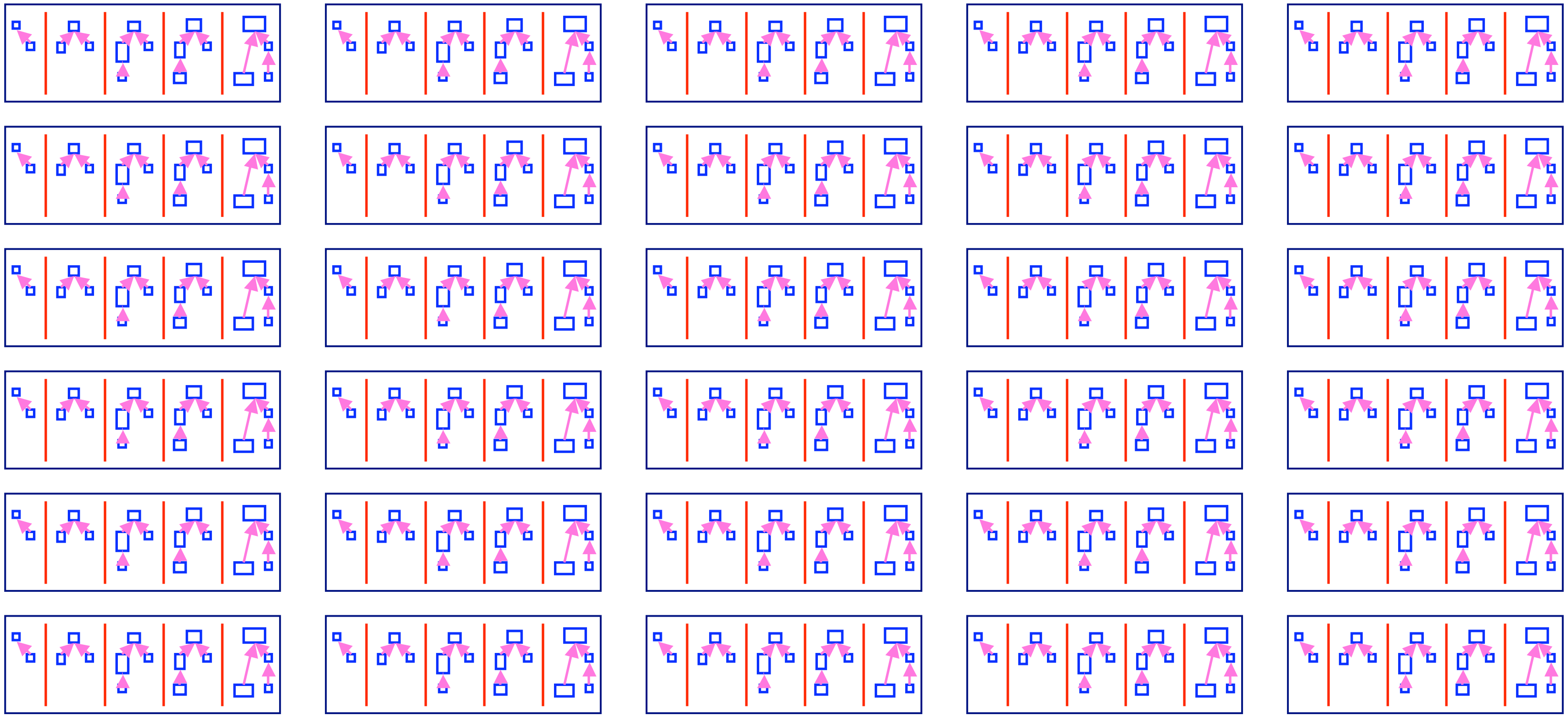
C was removed

B is stable

E is newborn

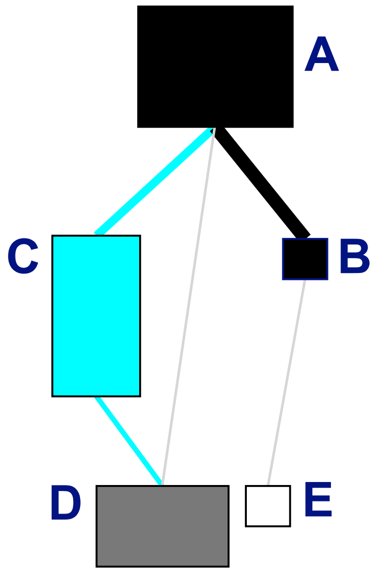
D inherited from C and then from A …

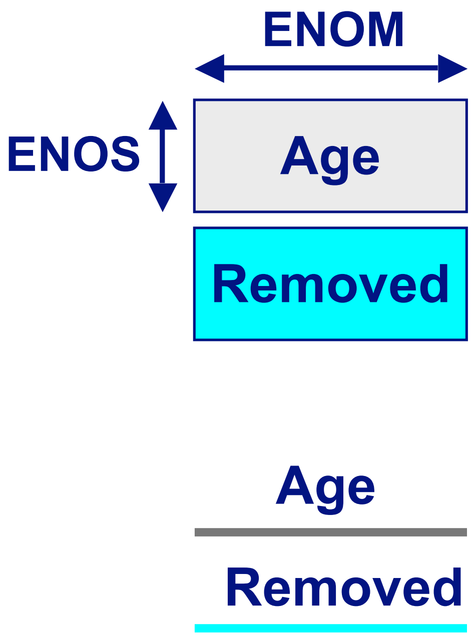
… but useful information is hidden among large amounts of data



How were the hierarchies evolved?

Hierarchy Evolution Complexity View characterizes class hierarchy histories





Class History

Inheritance History

A is persistent

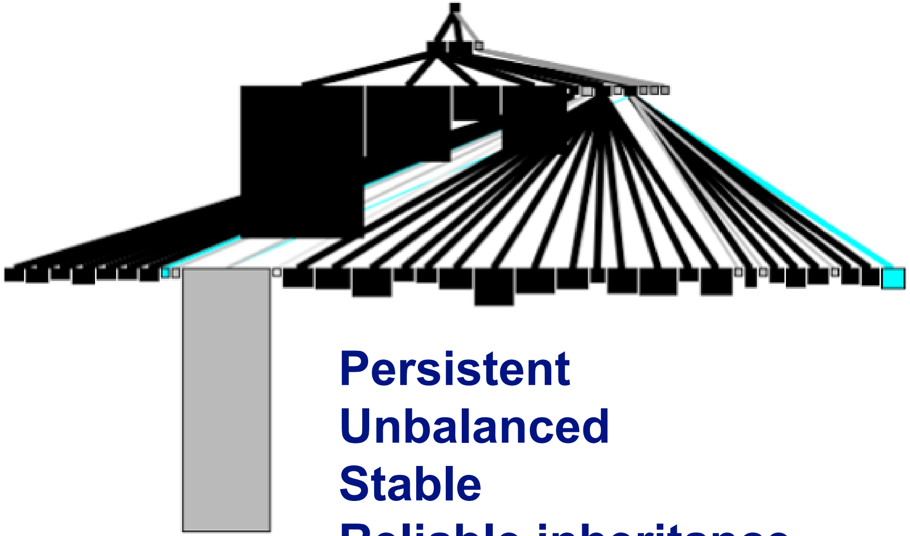
C was removed

B is stable

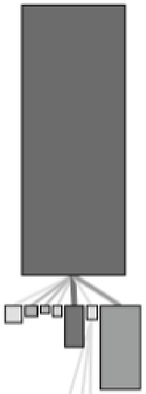
E is newborn

D inherited from C and then from A …

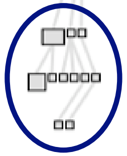
Case study: Class hierarchies in Jun reveal evolution patterns



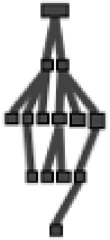
Persistent
Unbalanced
Stable
Reliable inheritance



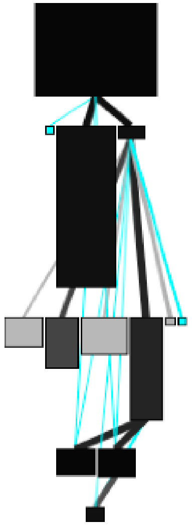
Young
Unstable root
Reliable inheritance



Newborn

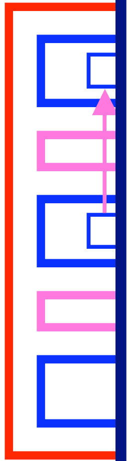


Old
Stable
Balanced
Reliable inheritance



Old
Unstable
Unbalanced
Unreliable inheritance

Overview



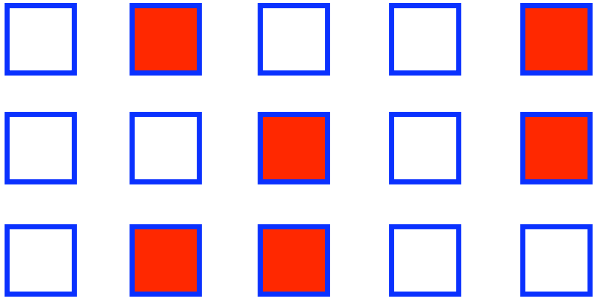
on

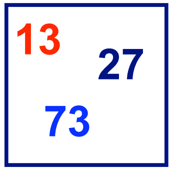
on

Hismo

Applications

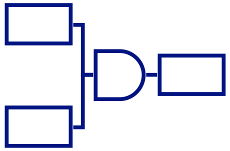
Application: Detecting co-change patterns



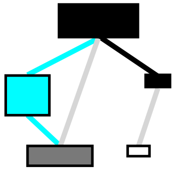


Historical measurements

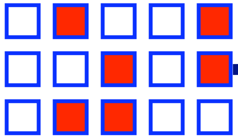
Yesterday’s Weather



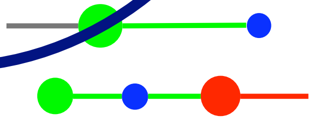
History-based Detection Strategies



Hierarchy evolution



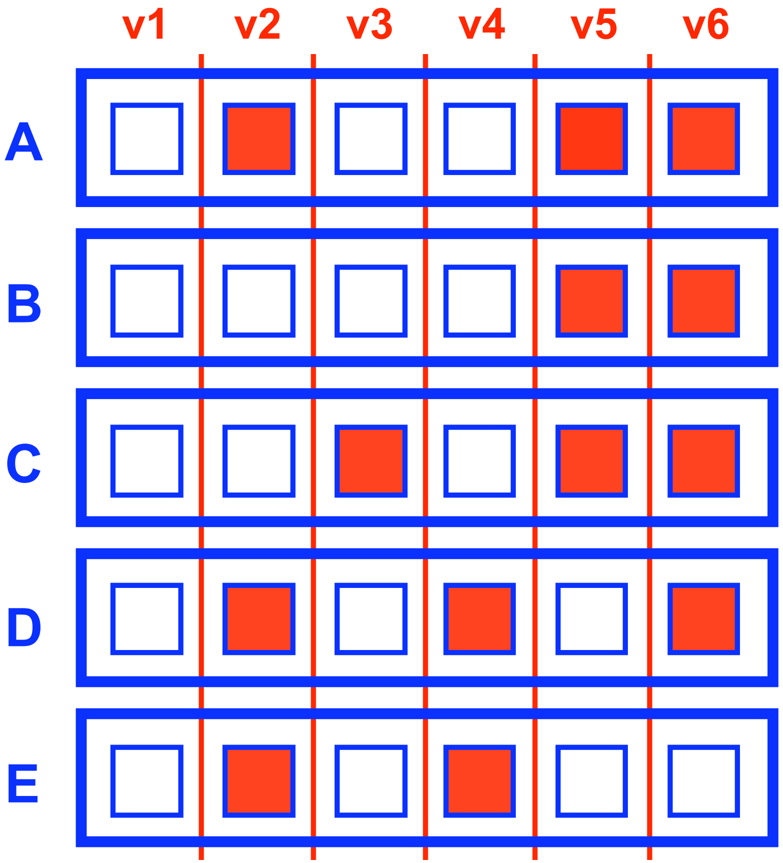
Co-change patterns



Ownership Map

Context: Repeated co-changes reveal hidden dependencies

[Gall etal. '98]



	v1	v2	v3	v4	v5	v6
A	□	🟥	□	□	🟥	🟥
B	□	□	□	□	🟥	🟥
C	□	□	🟥	□	🟥	🟥
D	□	🟥	□	🟥	□	🟥
E	□	🟥	□	🟥	□	□

Can we identify co-change patterns like:

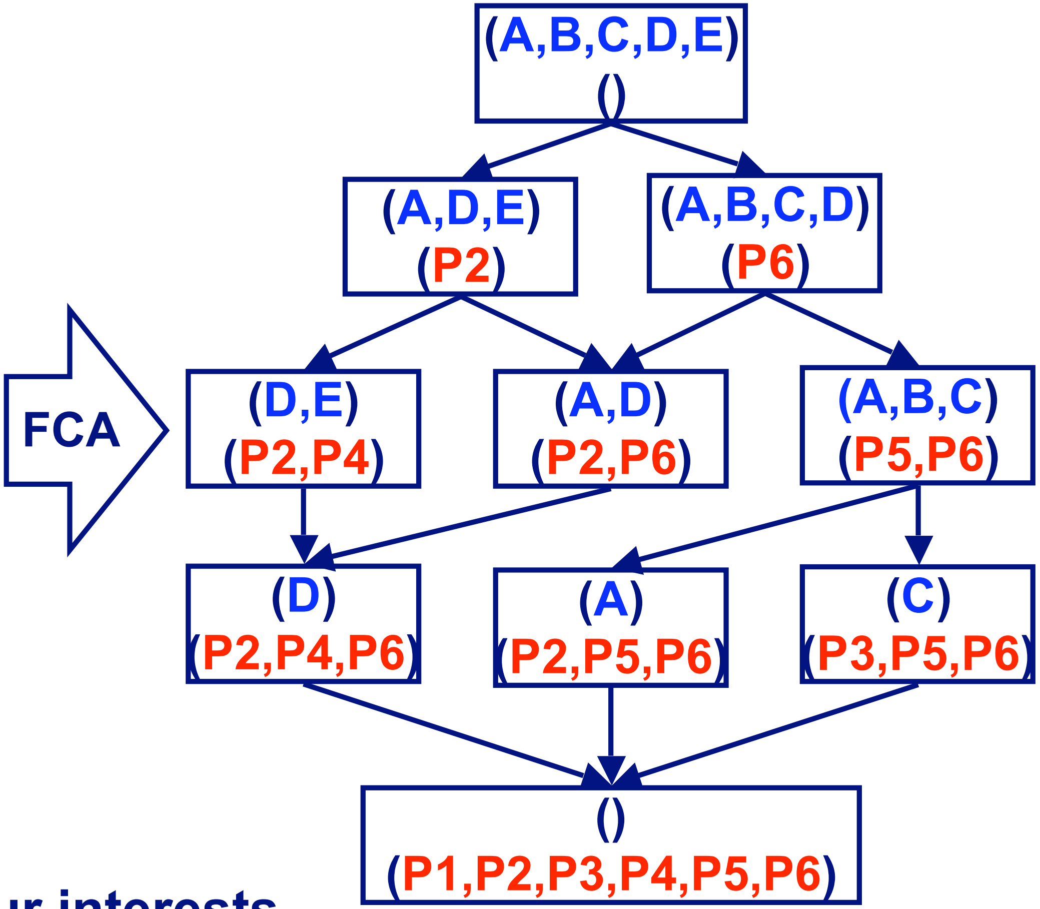
- Parallel Inheritance
- Shotgun Surgery
- ...

?

Formal Concept Analysis (FCA) finds elements that have properties in common

[Ganter, Wille '99]

	P1	P2	P3	P4	P5	P6
A		■			■	■
B					■	■
C			■		■	■
D		■		■		■
E		■		■		

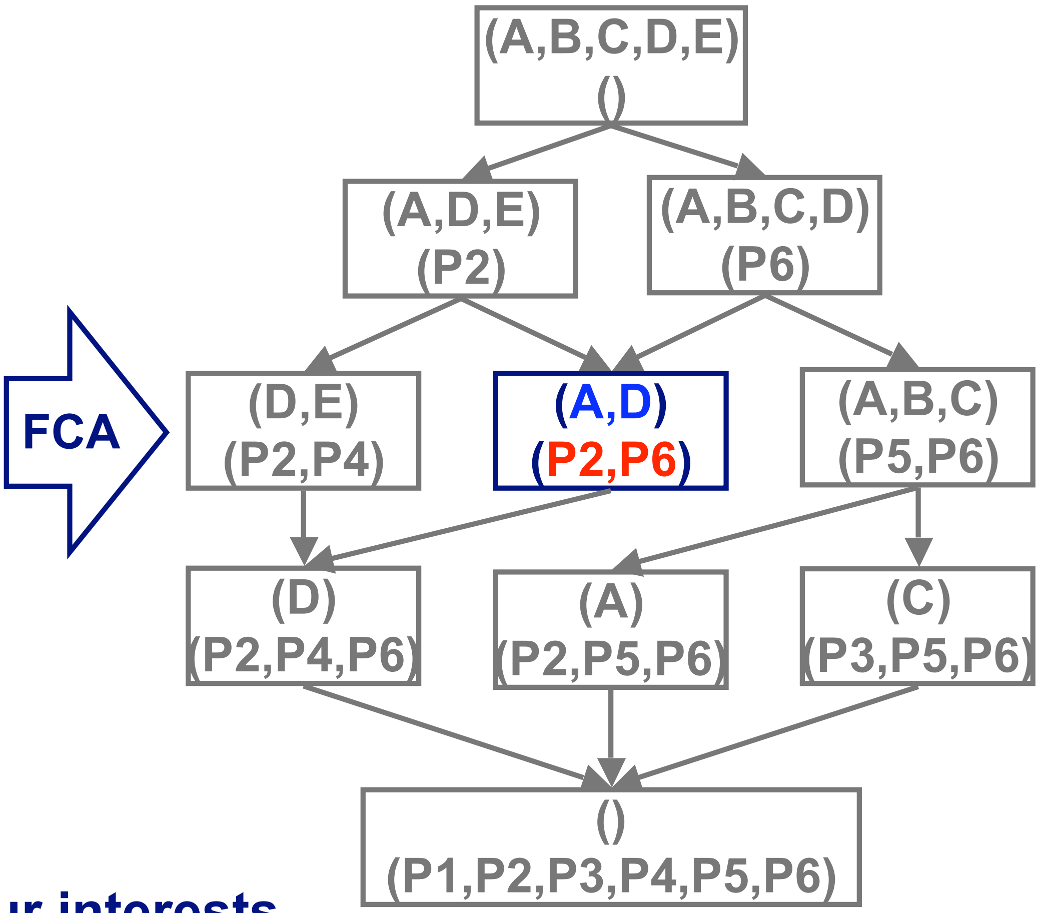


To use FCA, we need to map our interests on elements and properties

Formal Concept Analysis (FCA) finds elements that have properties in common

[Ganter, Wille ‘99]

	P1	P2	P3	P4	P5	P6
A		■			■	■
B					■	■
C			■		■	■
D		■		■		■
E		■		■		



FCA

(A,B,C,D,E)
()

(A,D,E)
(P2)

(A,B,C,D)
(P6)

(D,E)
(P2,P4)

(A,D)
(P2,P6)

(A,B,C)
(P5,P6)

(D)
(P2,P4,P6)

(A)
(P2,P5,P6)

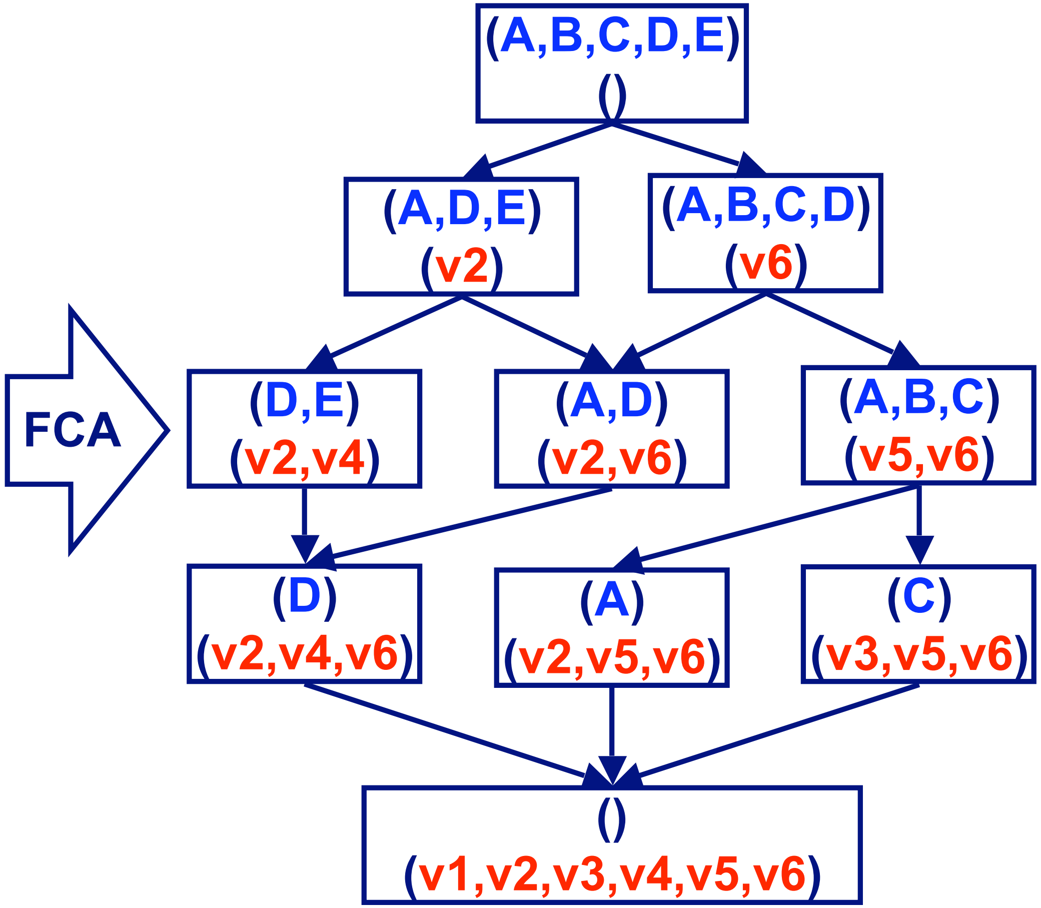
(C)
(P3,P5,P6)

()
(P1,P2,P3,P4,P5,P6)

To use FCA, we need to map our interests on elements and properties

We use FCA to identify entities that co-changed repeatedly

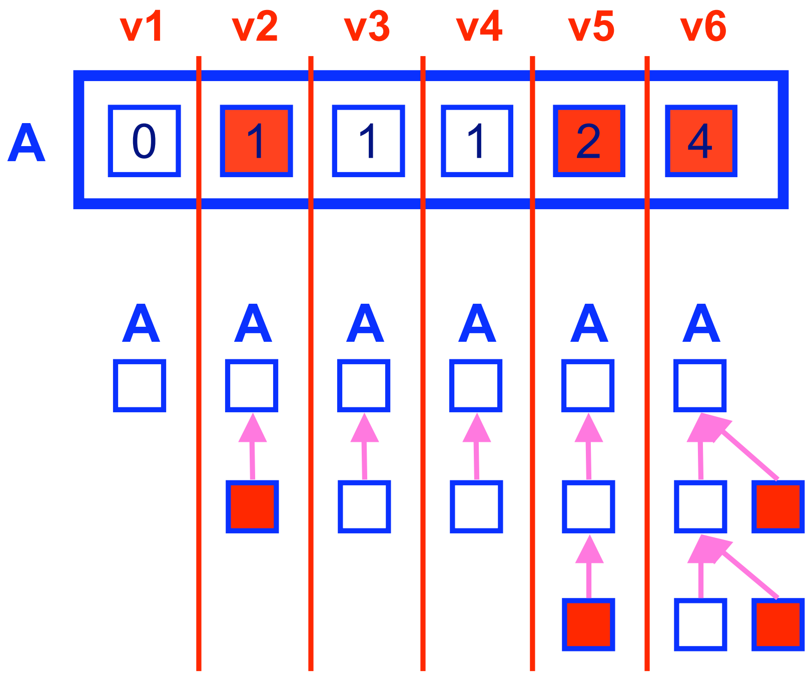
	v1	v2	v3	v4	v5	v6
A	□	🟥	□	□	🟥	🟥
B	□	□	□	□	🟥	🟥
C	□	□	🟥	□	🟥	🟥
D	□	🟥	□	🟥	□	🟥
E	□	🟥	□	🟥	□	□



Elements = Histories

Properties = “changed in version X”

Example: Parallel inheritance denotes children added in several hierarchies



Elements = ClassHistories

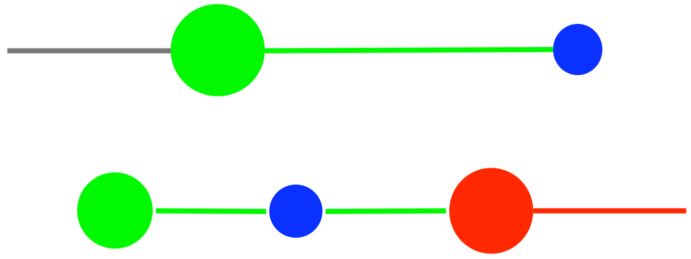
Properties = “changed number of children in version X”

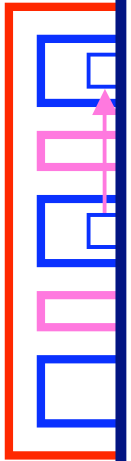
Case study: JBoss

ServiceMBeanSupport JBossTestCase	14 versions
EJBLocalHome EJBLocalObject	9 versions

Overview

Application: Ownership map



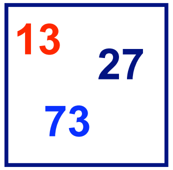


on

on

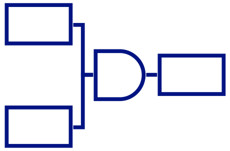
Hismo

Applications

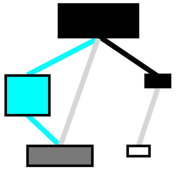


Historical measurements

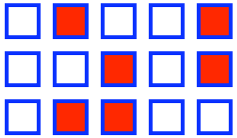
Yesterday's Weather



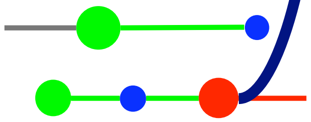
History-based Detection Strategies



Hierarchy evolution



Co-change patterns

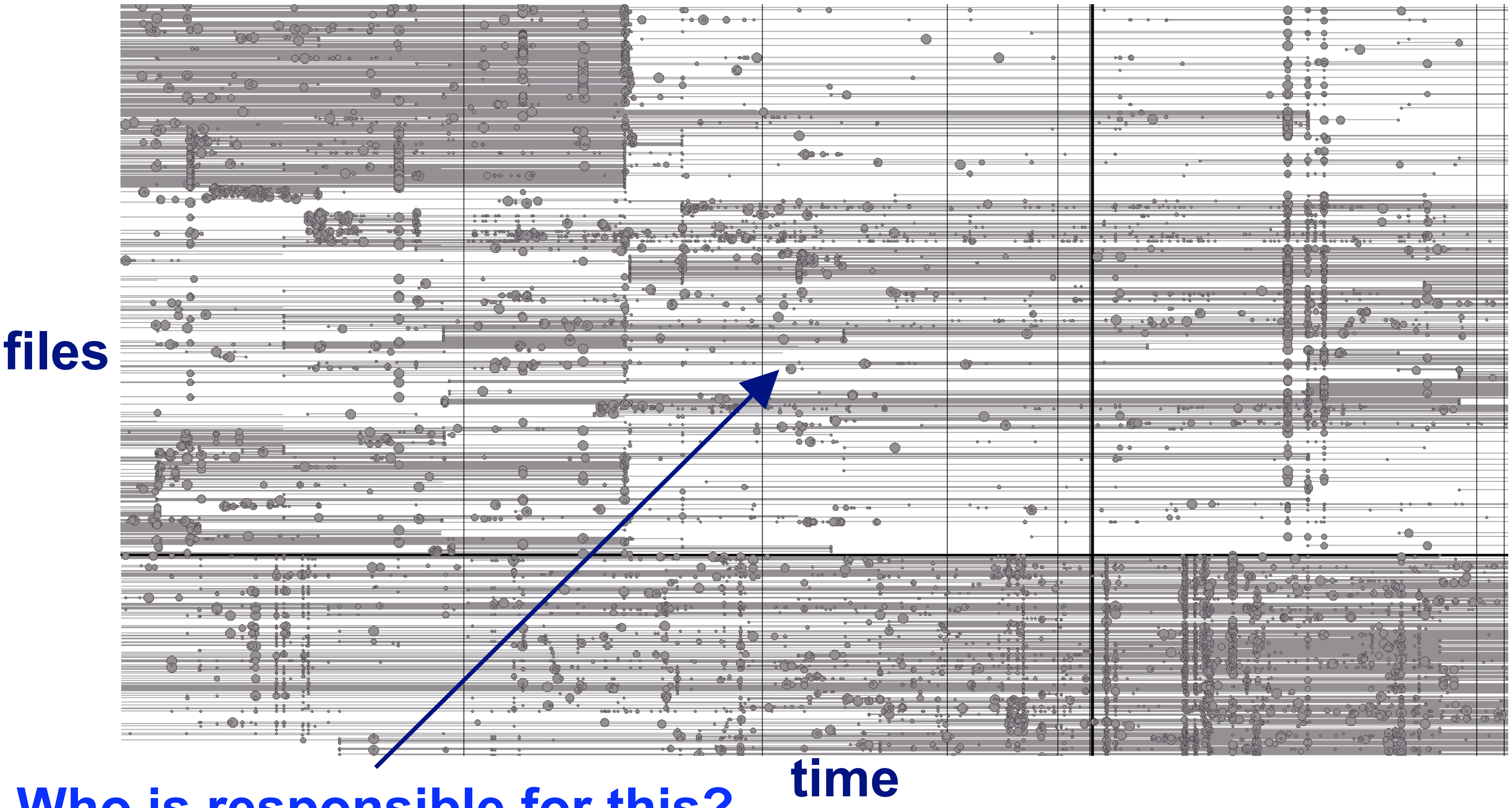


Ownership Map

Context: The code history might tell you what happened, but not why it happened

Case study: Outsight

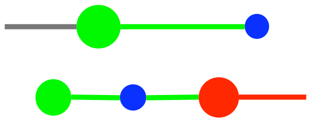
[Rysselberghe, Demeyer ‘04]

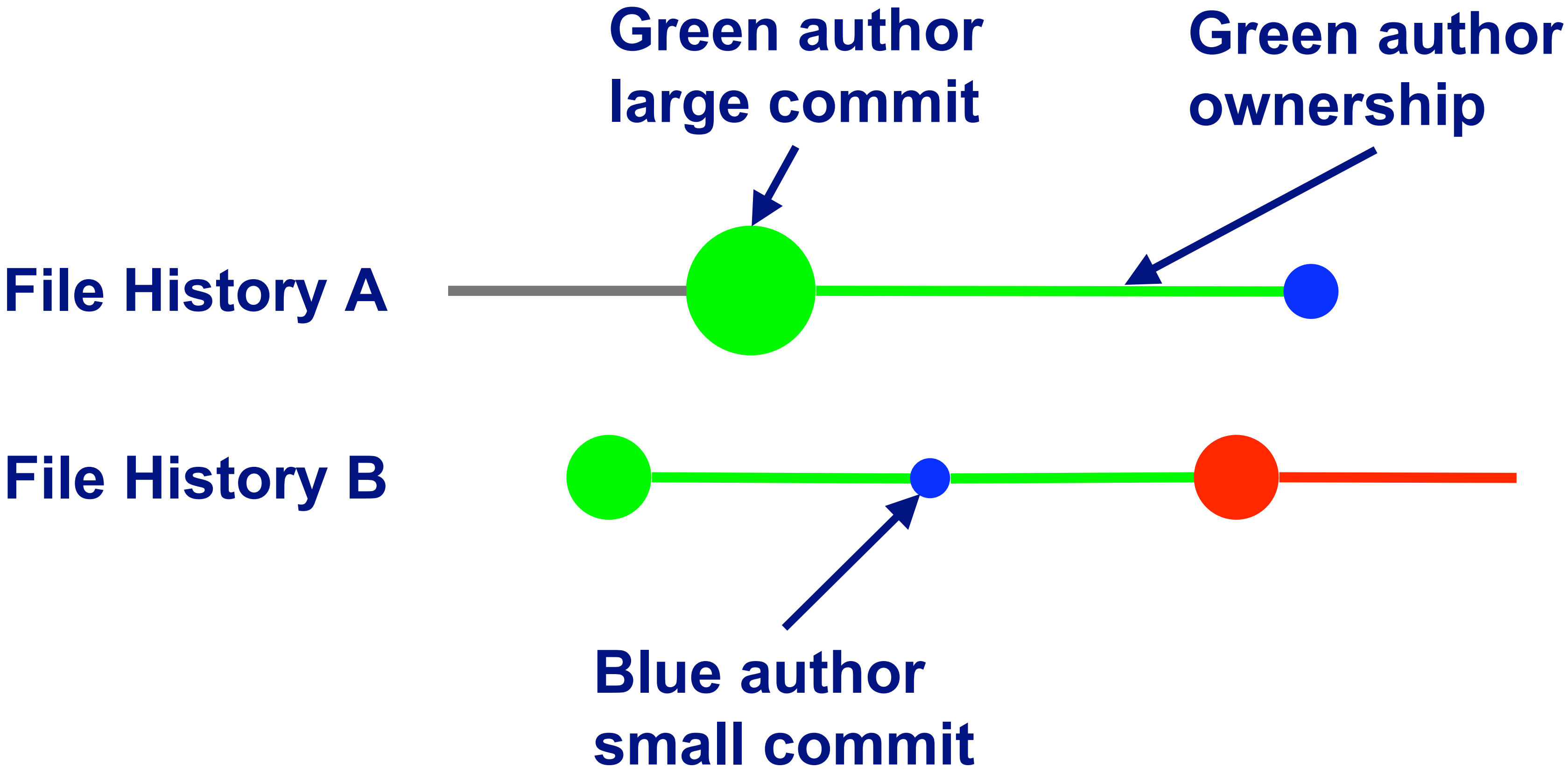


Who is responsible for this?

© Tudor Gîrba

We color the lines to show which author owned which files in which period





Green author large commit

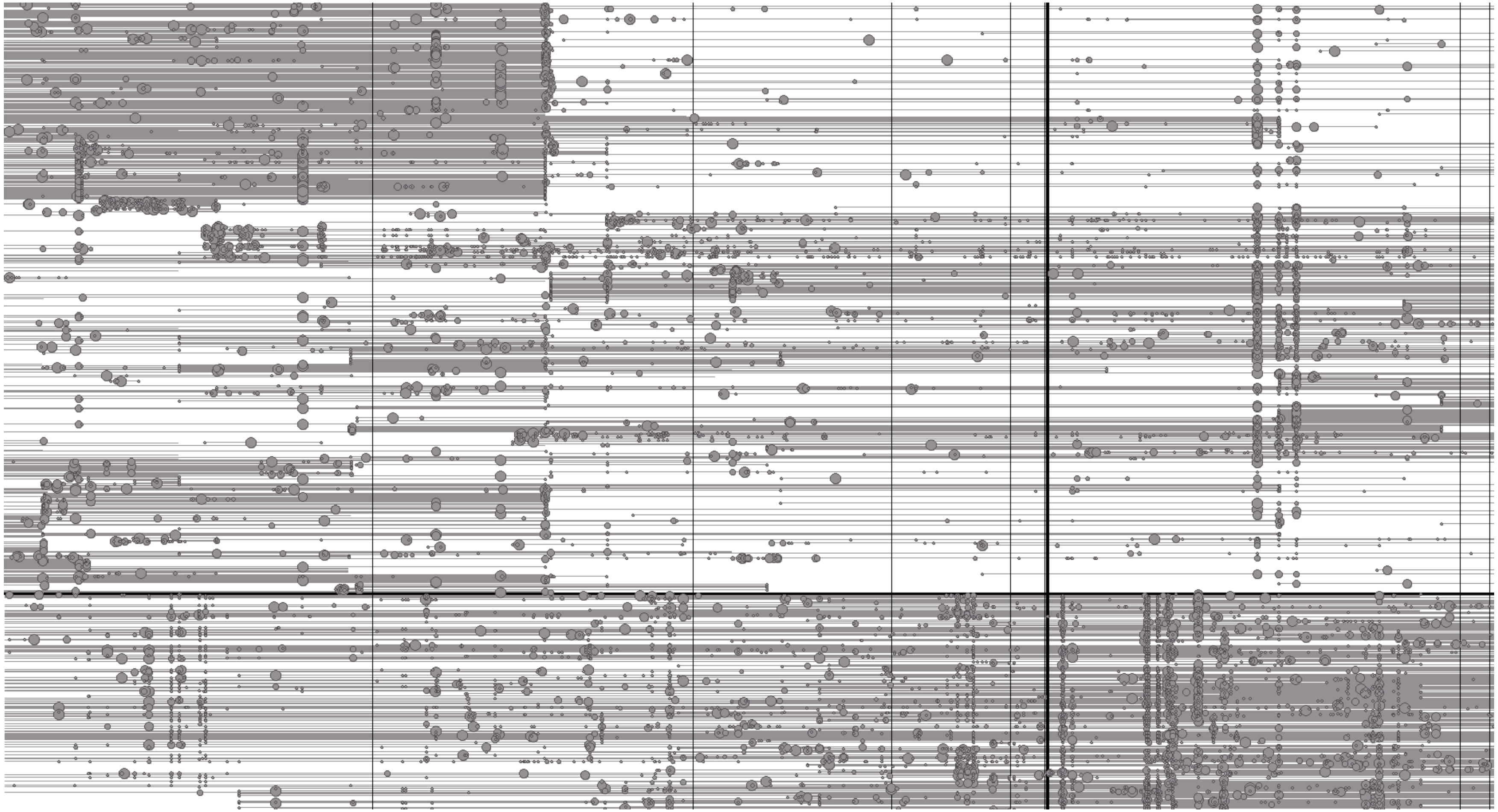
Green author ownership

File History A

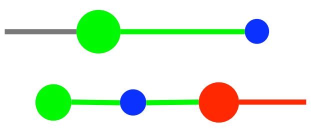
File History B

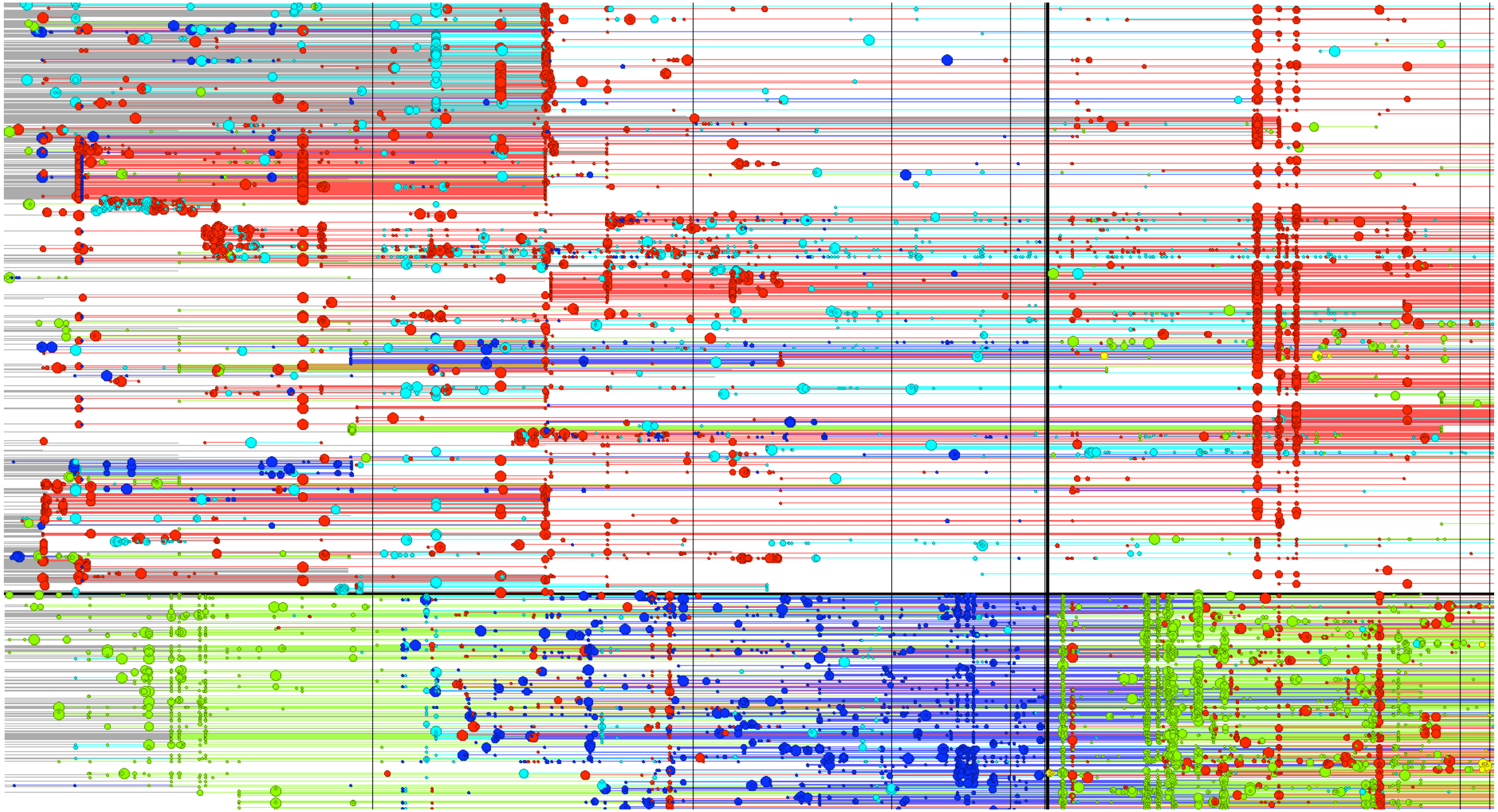
Blue author small commit

The commit history shows what happened

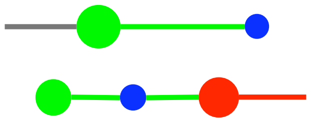


Ownership Map shows which author owned which files in which period



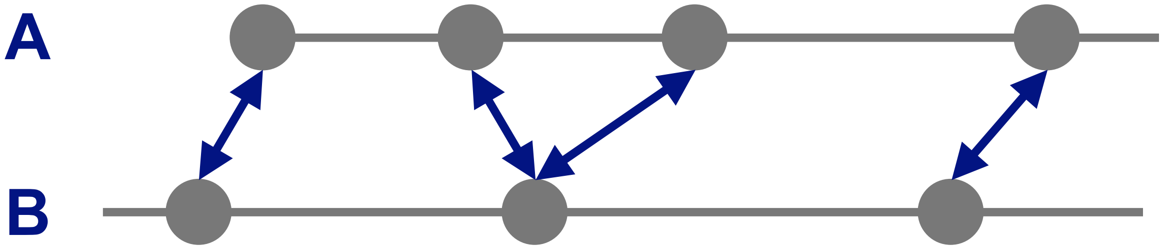


We cluster the file histories to favor colored blocks inside each module

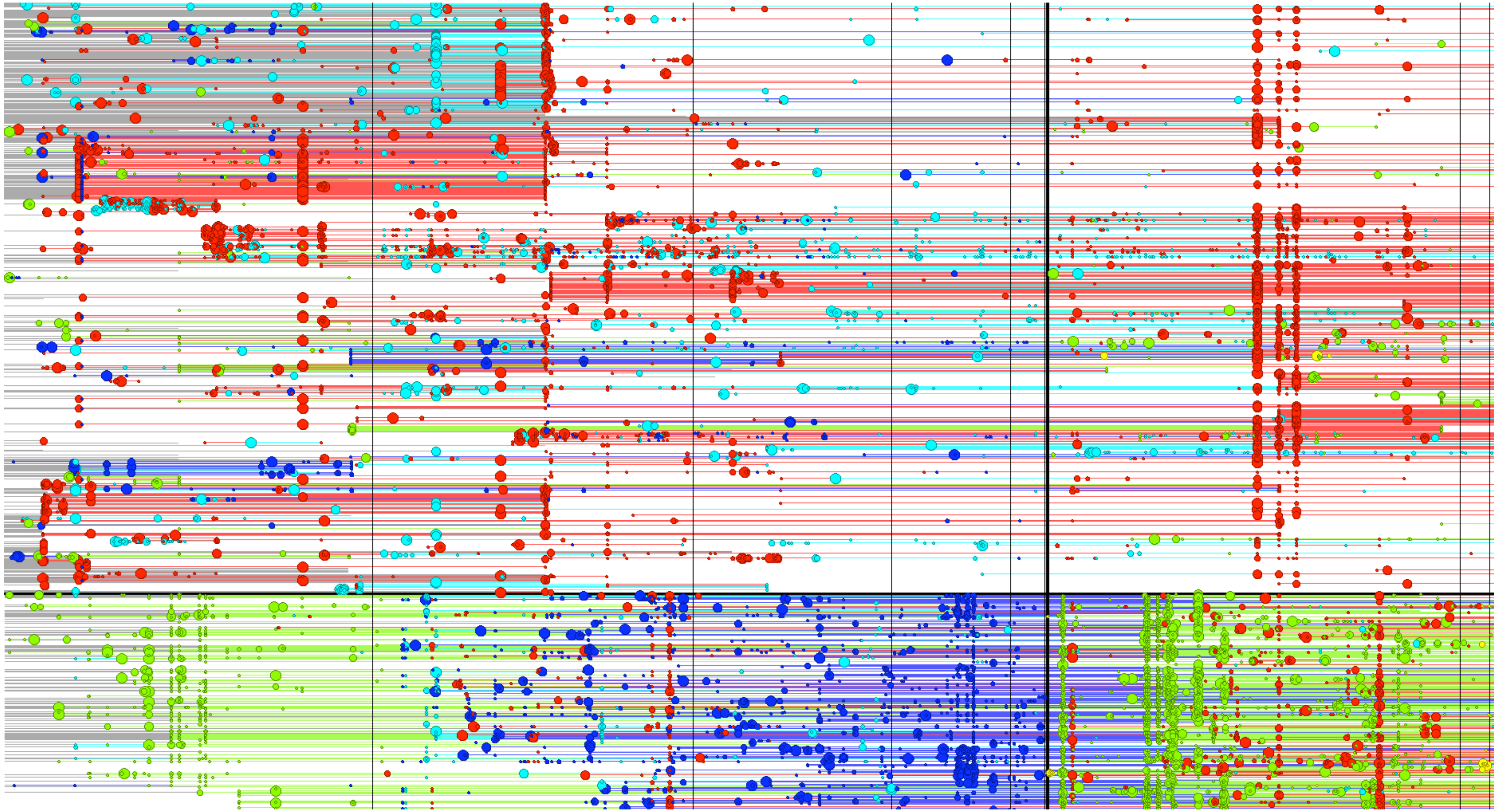


We use the Hausdorf distance between the commit timestamps

$$d(A, B) = \sum_{a \in A} \min^2\{\, |a - b| \mid b \in B \,\}$$

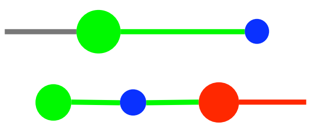


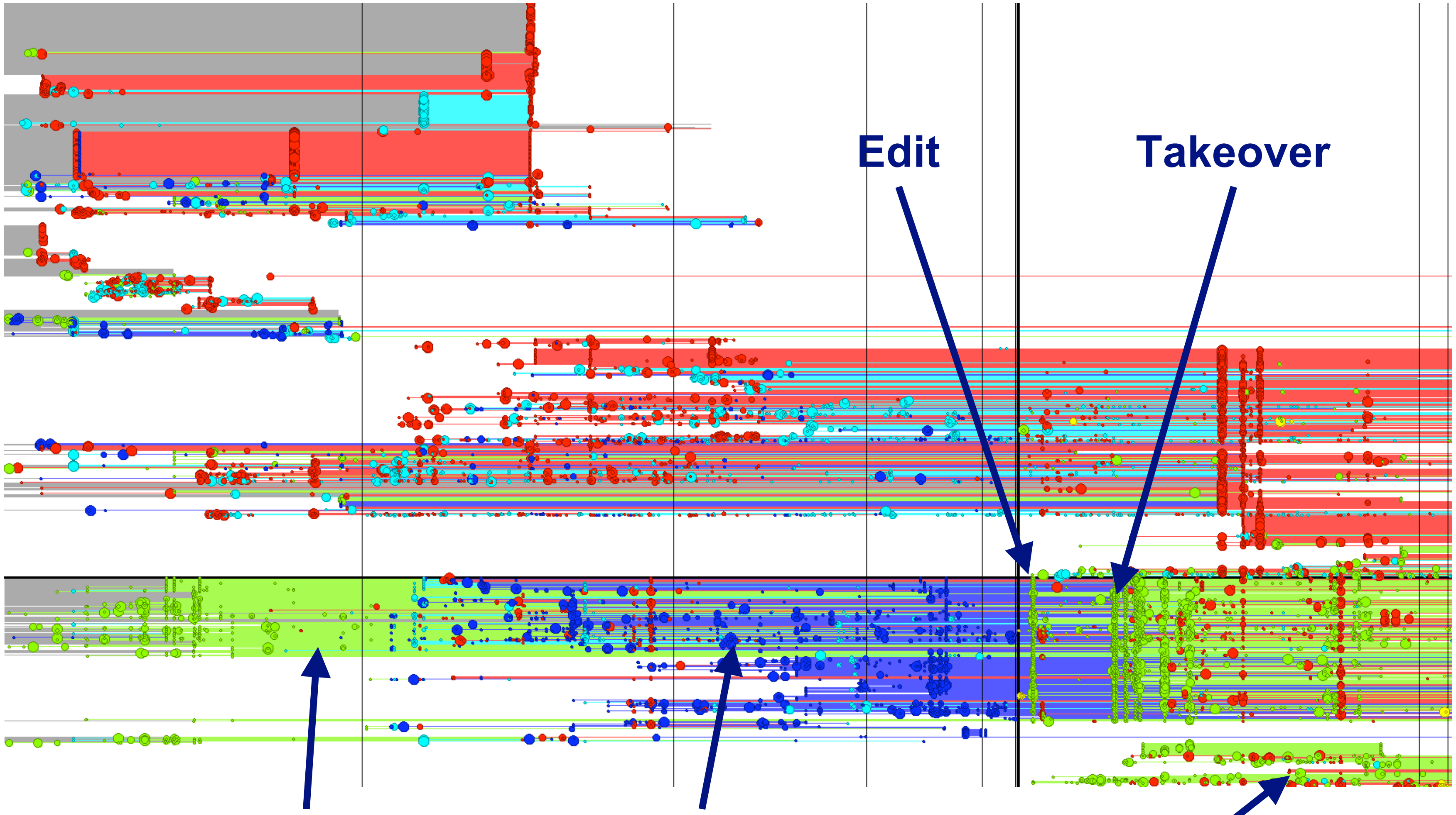
Ownership Map on alphabetically ordered files is not very useful, but …



© Tudor Gîrba

The ordered Ownership Map reveals developer patterns





Edit

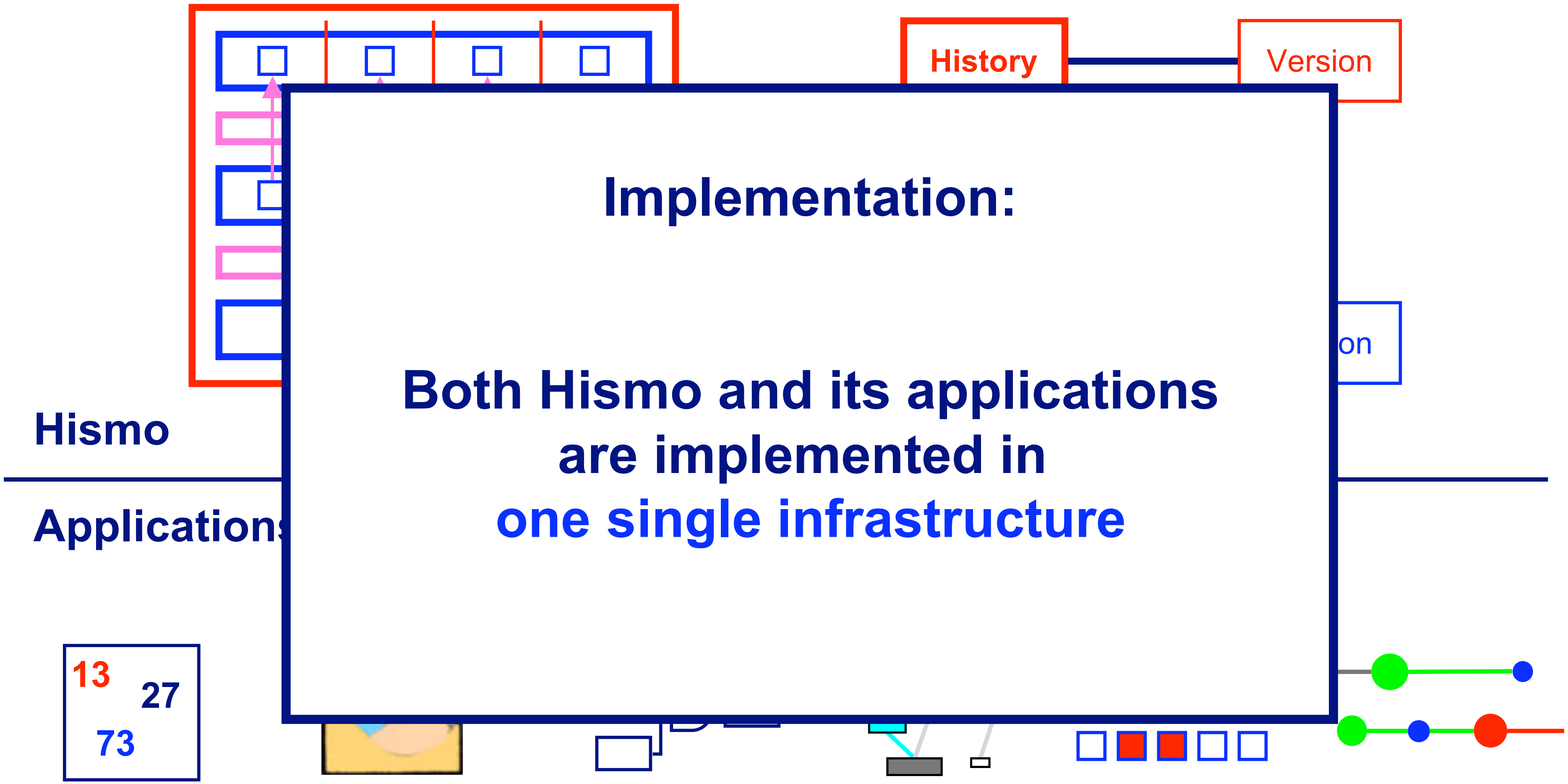
Takeover

Monologue

Familiarization

Dialogue

Overview



Implementation:

Both Hismo and its applications are implemented in **one single infrastructure**

Historical measurements

Yesterday’s Weather

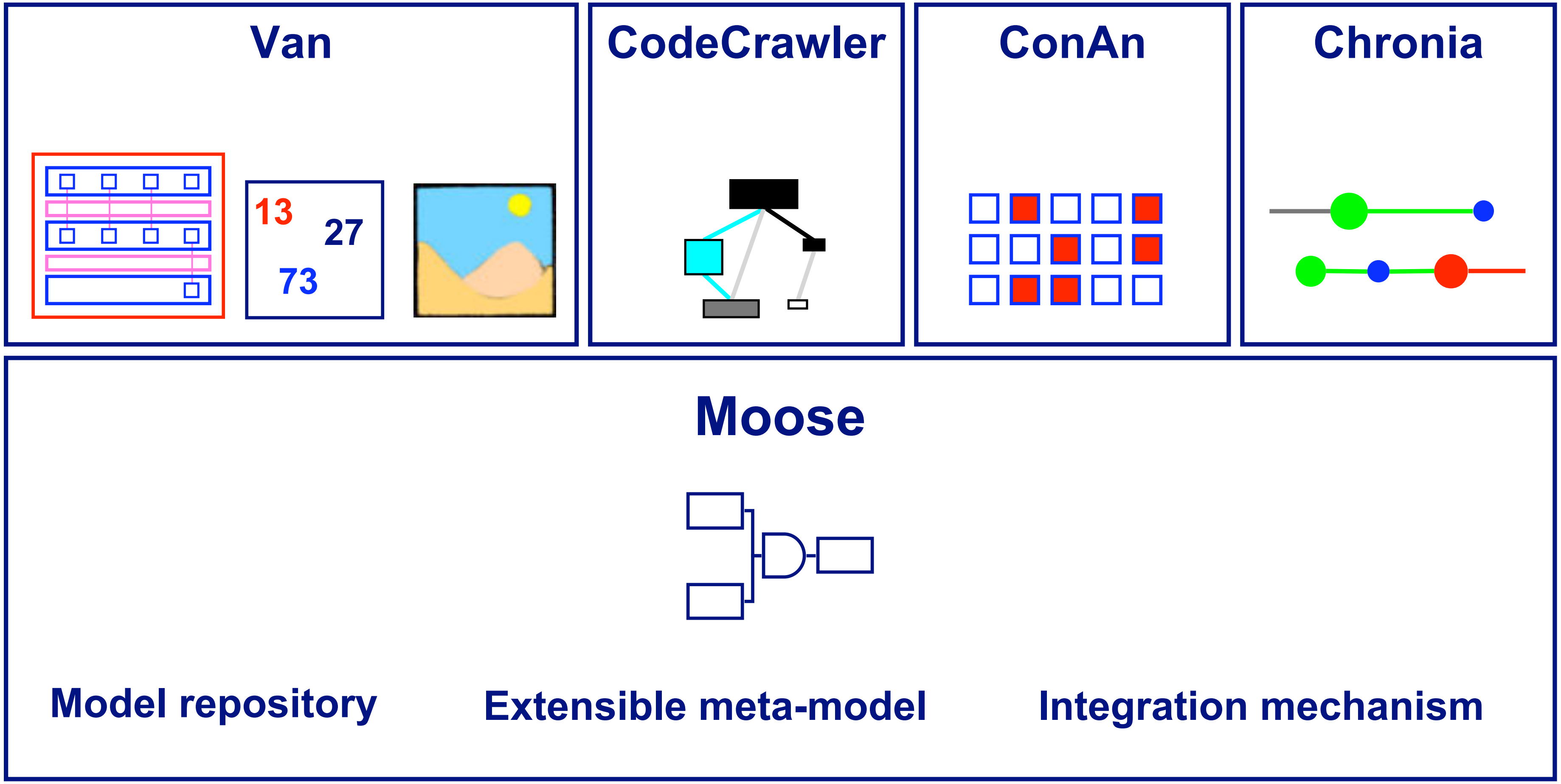
History-based Detection Strategies

Hierarchy evolution

Co-change patterns

Ownership Map

Implementation: All tools are integrated into Moose



Van

CodeCrawler

ConAn

Chronia

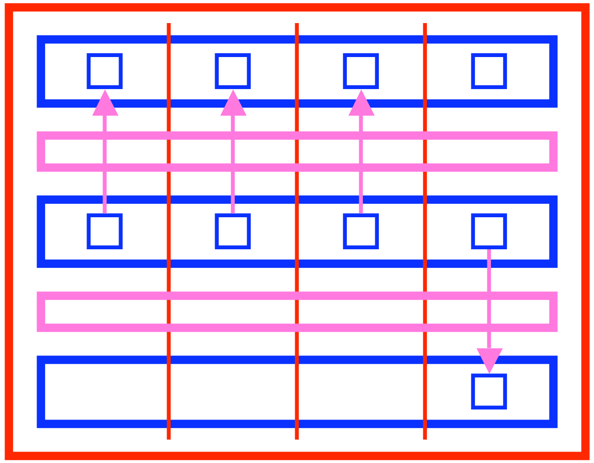
Moose

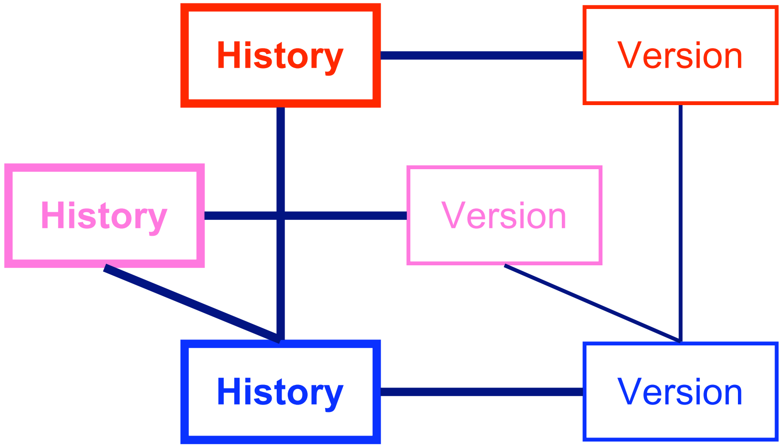
Model repository

Extensible meta-model

Integration mechanism

Conclusion: Hismo offers a uniform way of expressing evolution analyses

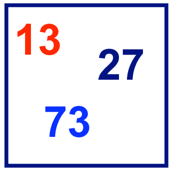




Hismo

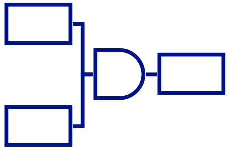
Questions?

Applications

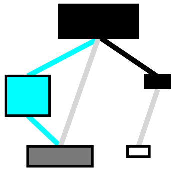


Historical measurements

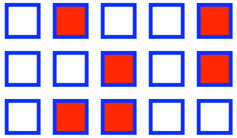
Yesterday's Weather



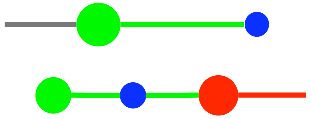
History-based Detection Strategies



Hierarchy evolution

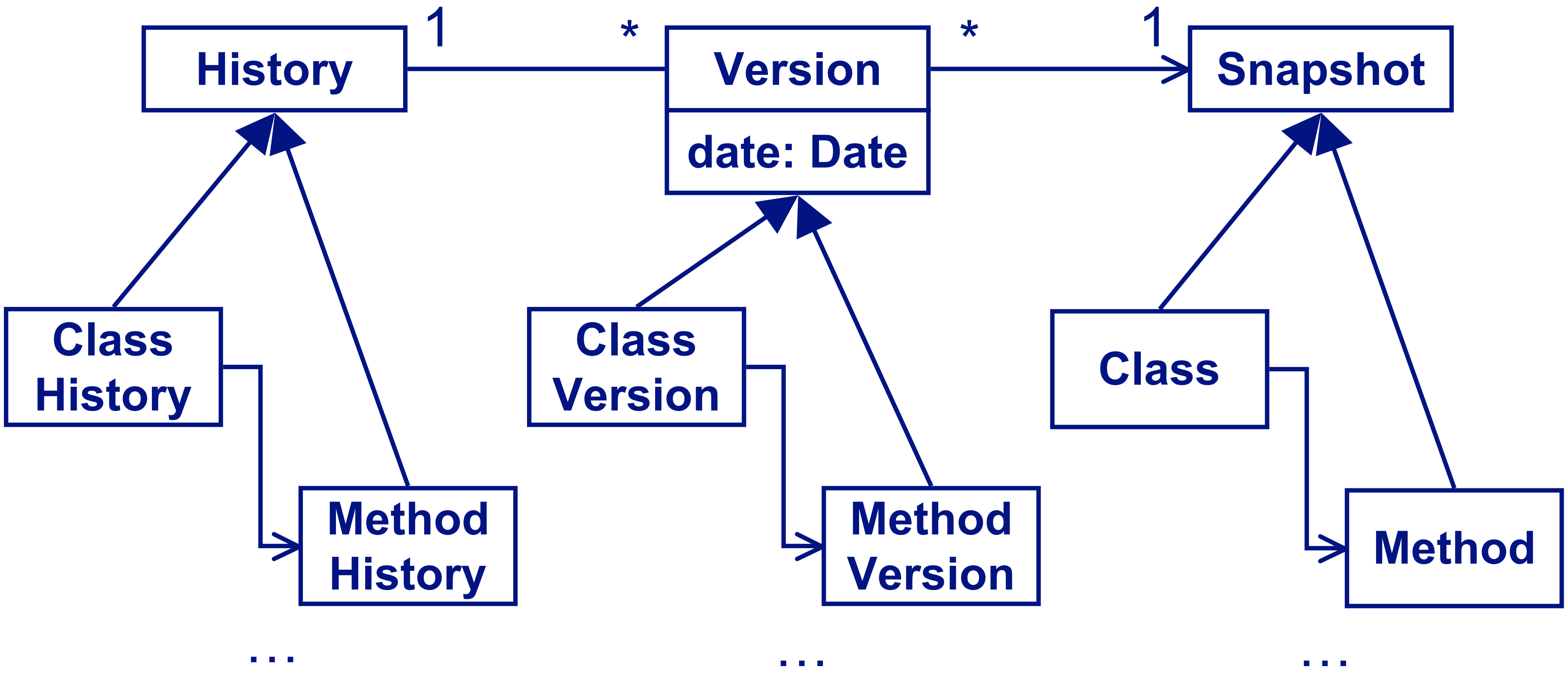


Co-change patterns

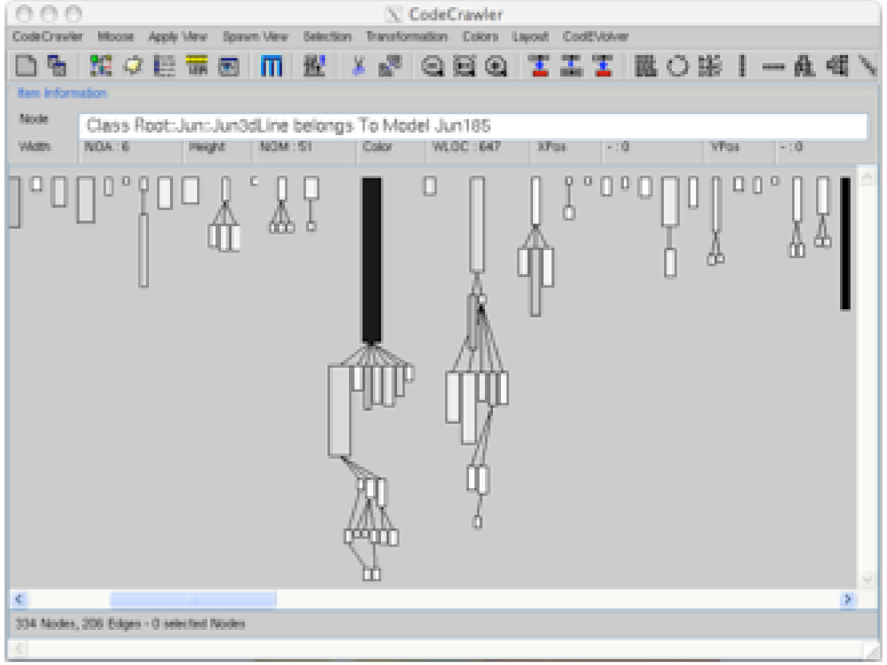


Ownership Map

Hismo: History is a sequence of Versions, where a Version adds the notion of time to Snapshot



The techniques are orthogonal to the type of data

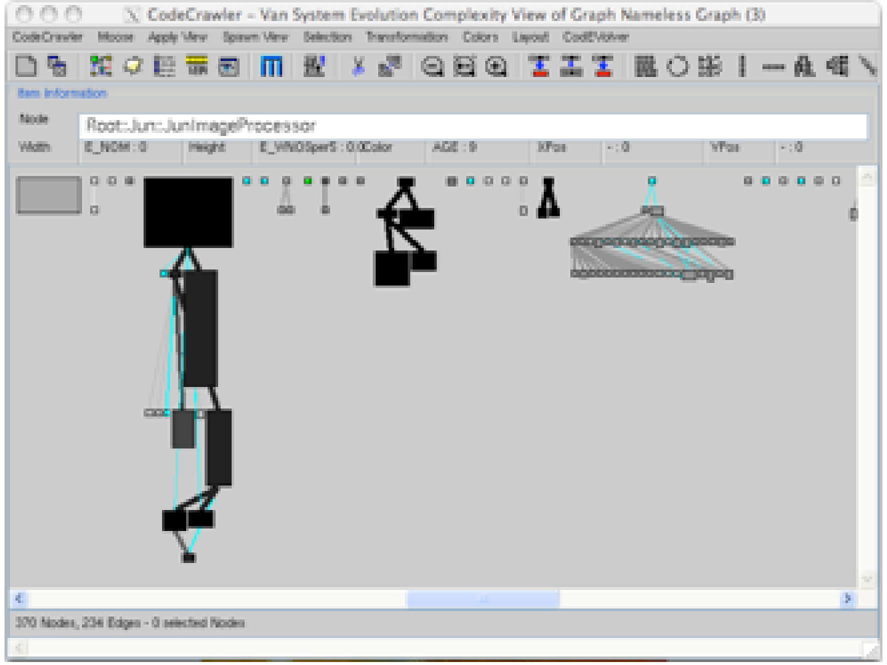


CodeCrawler
CodeCrawler Moose Apply View Spawn View Selection Transformation Colors Layout CodEVolver
Item Information

Node: Class Root::Jun::Jun3dLine belongs To Model Jun185

Width: NOA : 6 | Height: NOM : 51 | Color: WLOC : 647 | XPos: - : 0 | YPos: - : 0

334 Nodes, 206 Edges - 0 selected Nodes



CodeCrawler - Van System Evolution Complexity View of Graph Nameless Graph (3)
CodeCrawler Moose Apply View Spawn View Selection Transformation Colors Layout CodEVolver
Item Information

Node: Root::Jun::JunImageProcessor

Width: E_NOM : 0 | Height: E_WNOSperS : 0.0 | Color: AGE : 9 | XPos: - : 0 | YPos: - : 0

370 Nodes, 234 Edges - 0 selected Nodes

Node = class

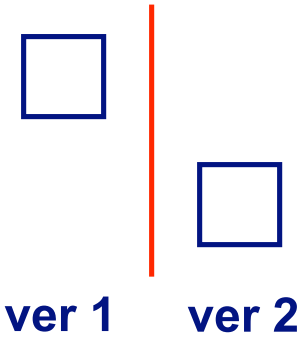
Edge = inheritance

Node = Class History

Edge = Inheritance History

Entity identity: Are two entities at different points in time the versions of the same history?

The current versioning systems record snapshots



What are the names?

What are the properties?

What we would like

Preserve the identity in the environment

Record changes as they happen