Mining and Visualizing Software Repositories to Understand Software Evolution

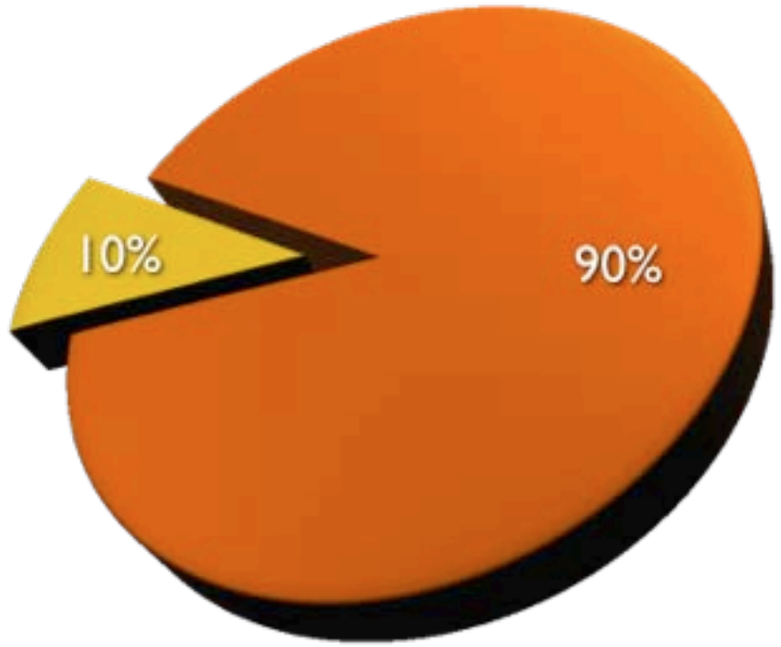
Marco D'Ambros
Faculty of Informatics, University of Lugano
Switzerland

Software Evolution is complex...

6 years of evolution of Mozilla: 3'000'000 lines of C and C++ code, over 1'000'000 of changes, hundreds of developers

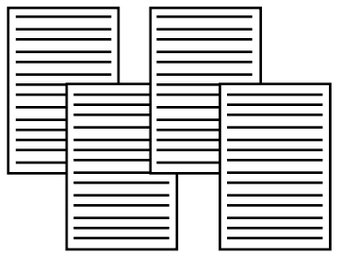
Software Evolution is complex...

... and expensive [1]

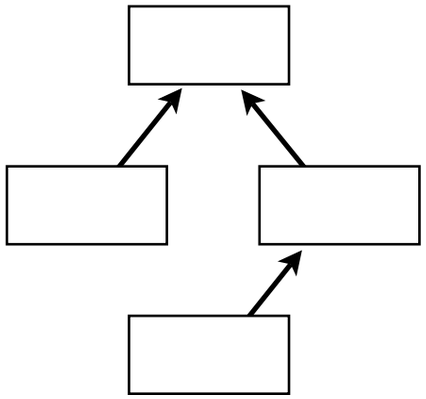


[1] L. Erlikh. *Leveraging legacy system dollars for e-business.* IT Professional 2, 3 (May. 2000)

Software Evolution Analysis

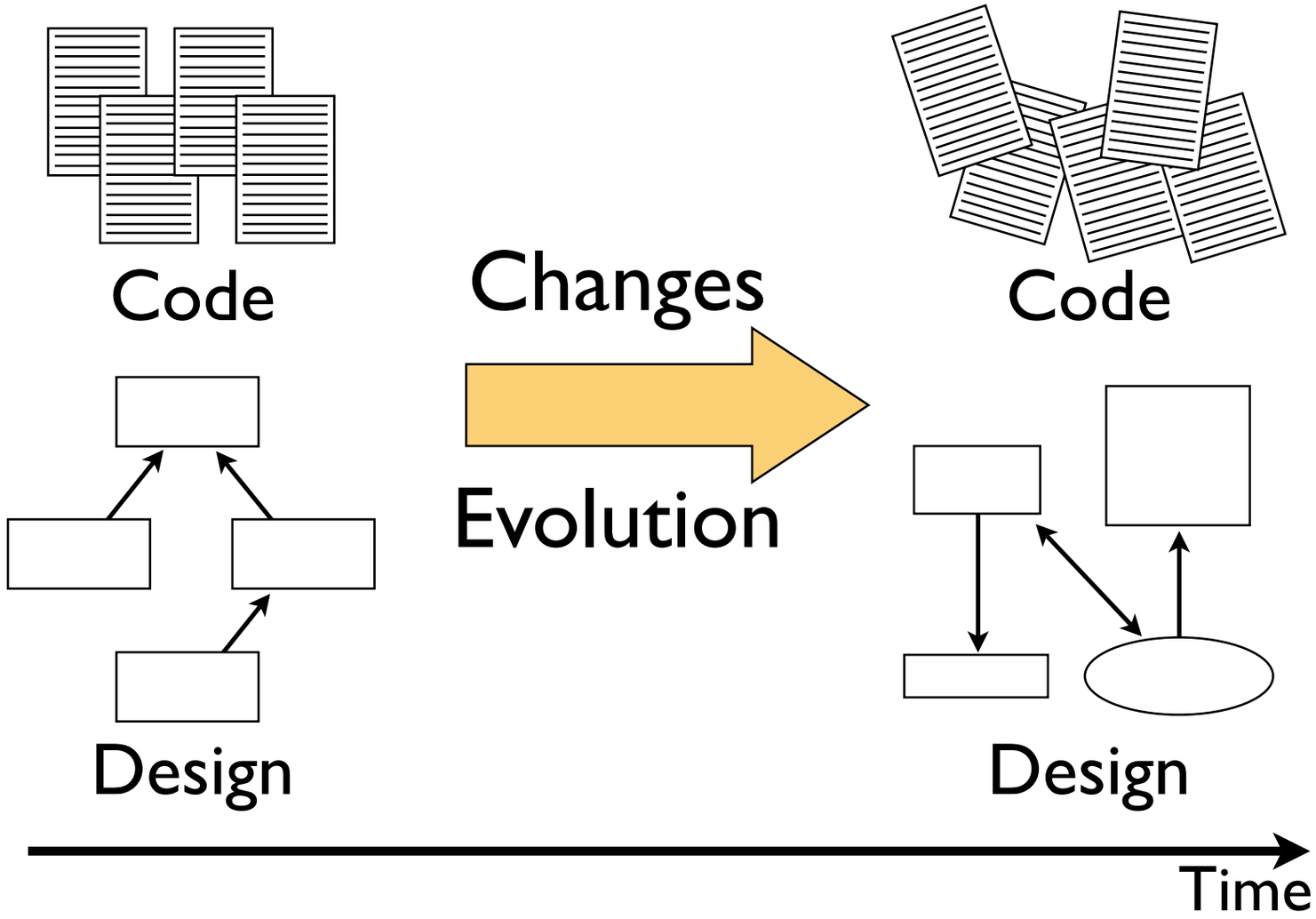


Code

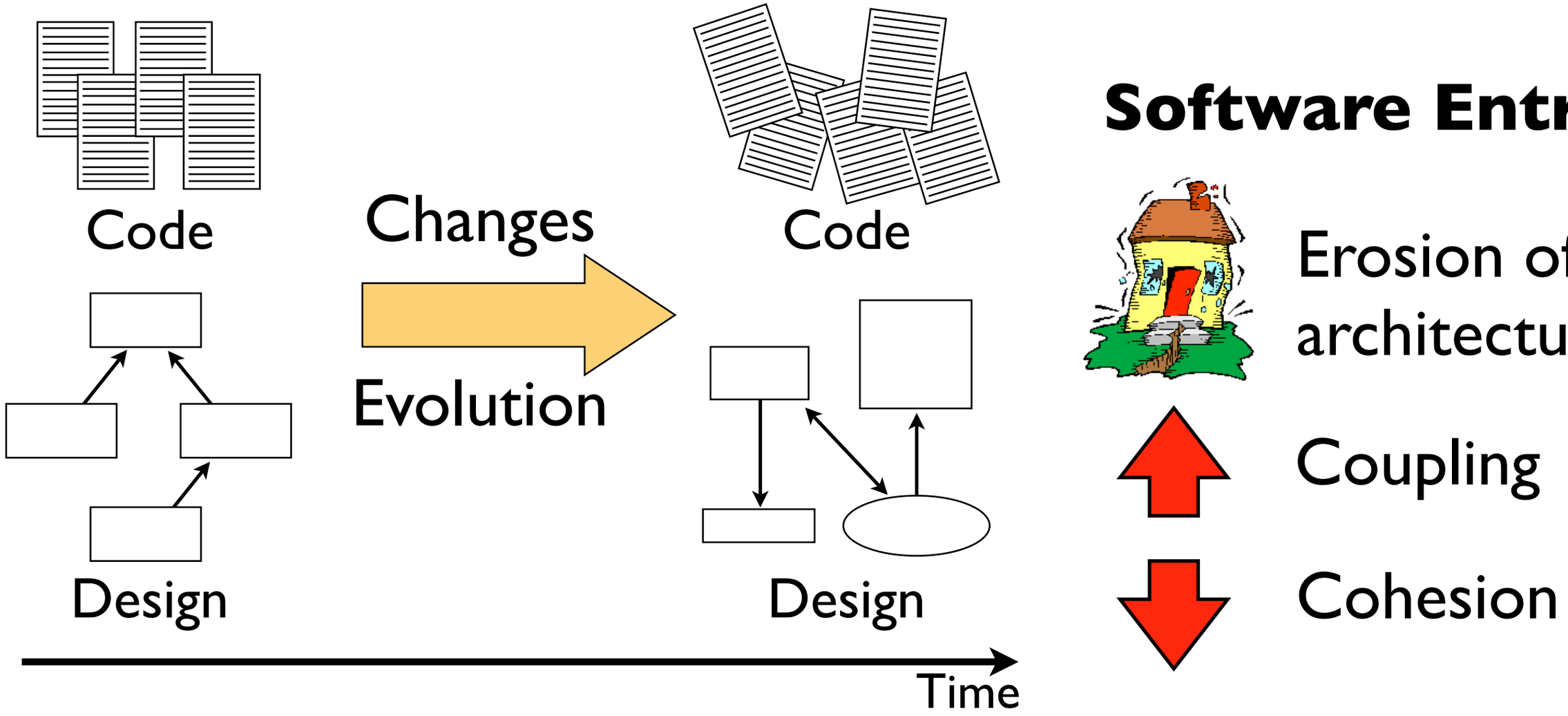


Design

Software Evolution Analysis



Software Evolution Analysis



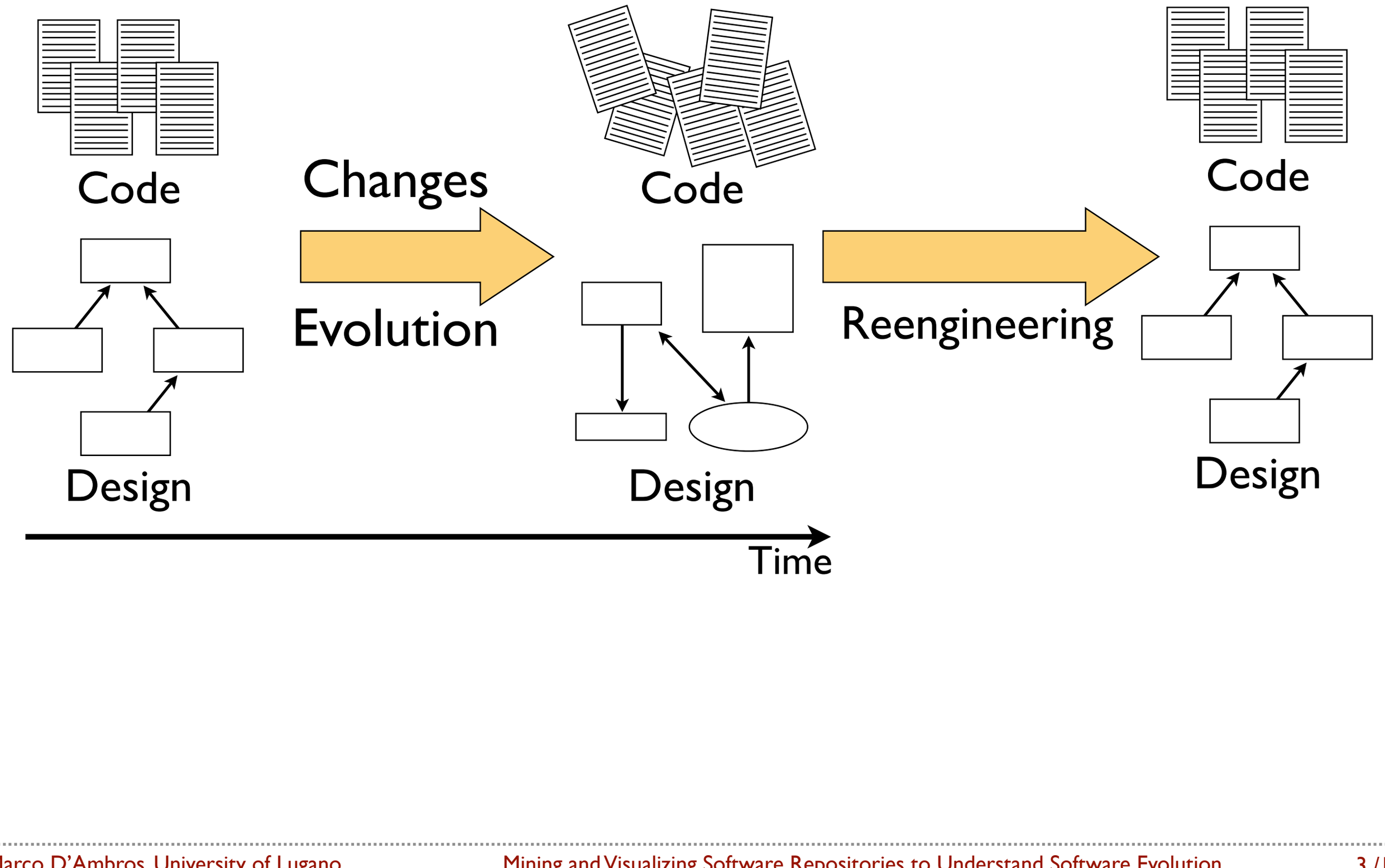
Software Entropy

Erosion of architecture

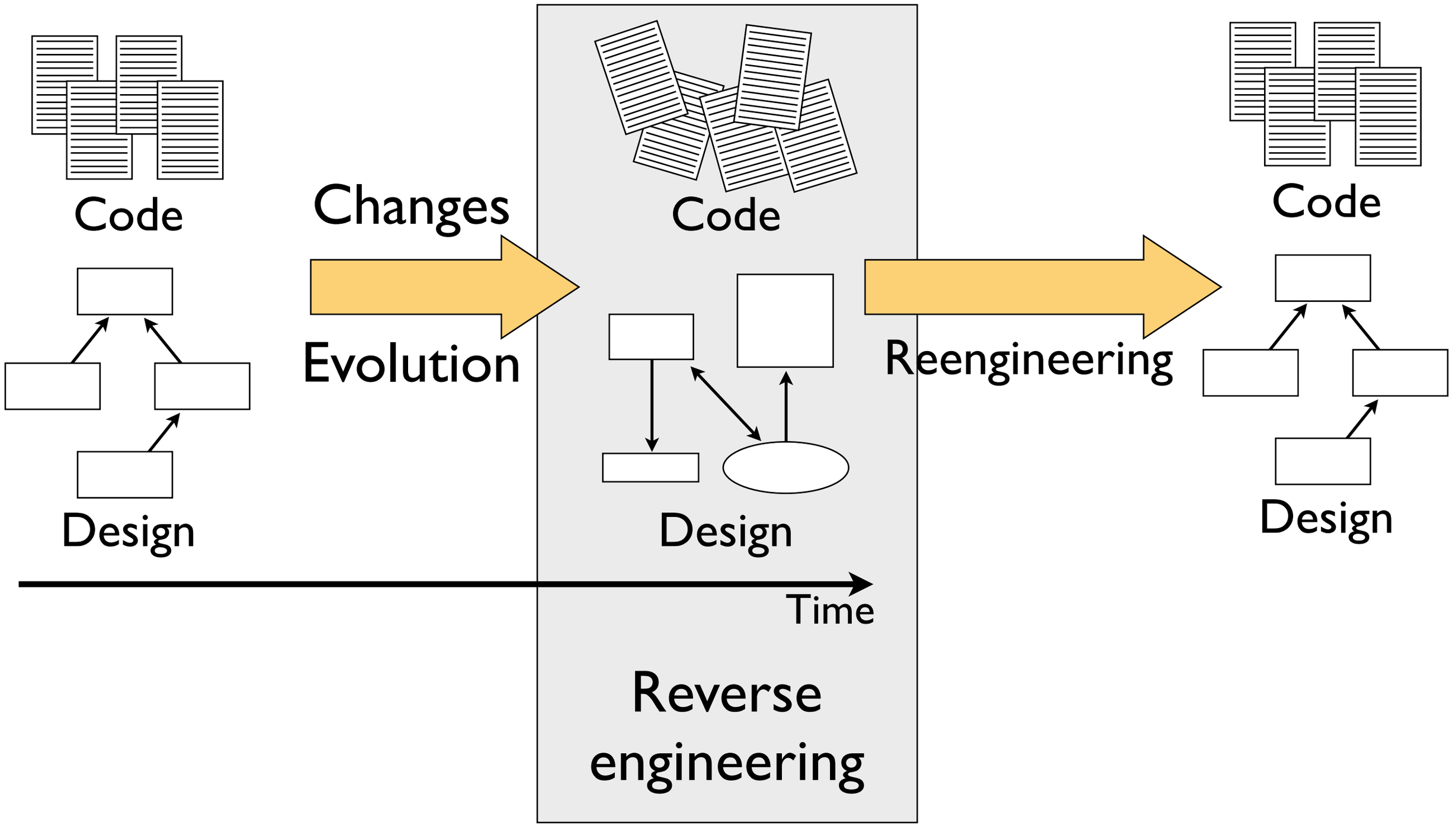
Coupling

Cohesion

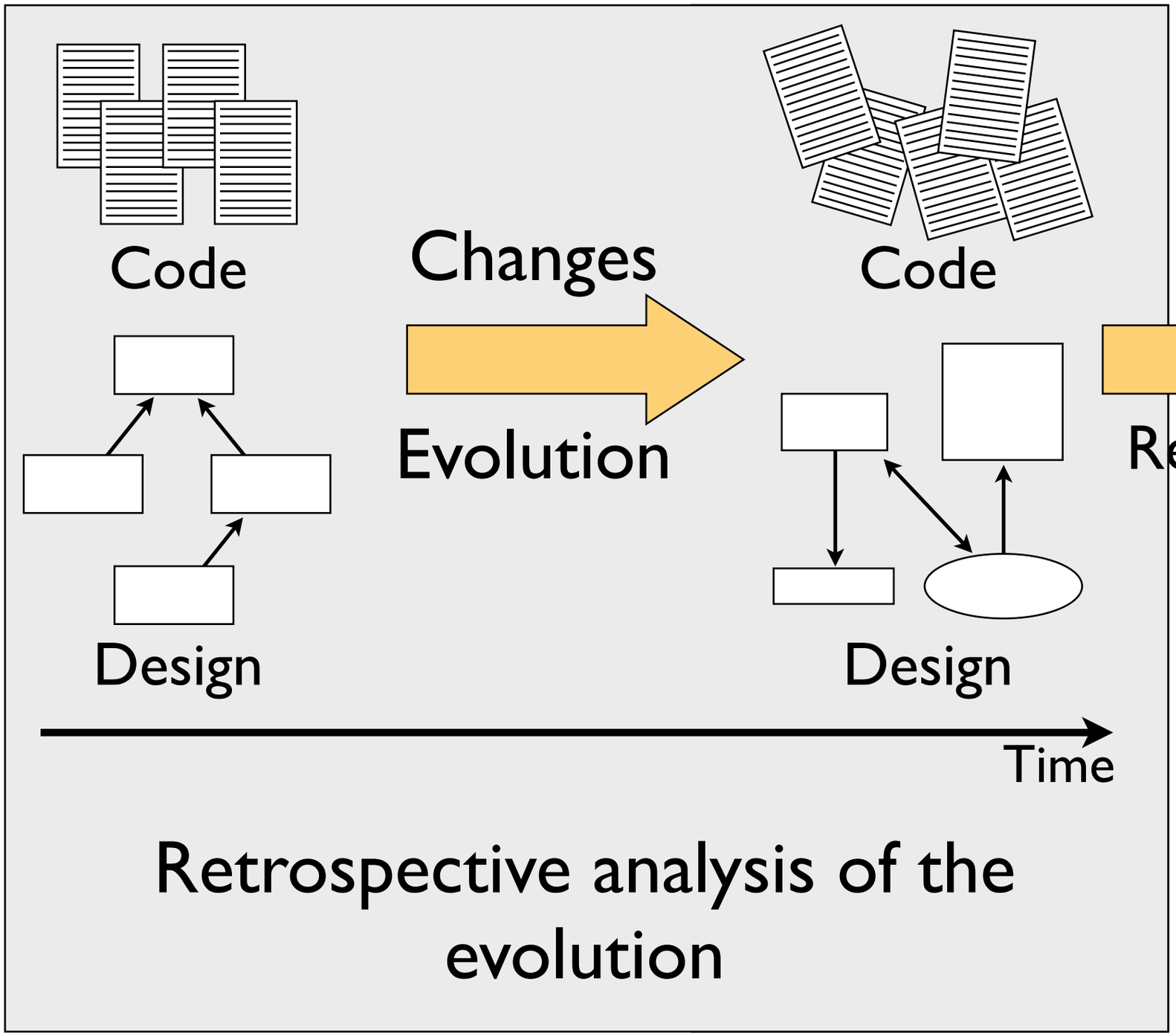
Software Evolution Analysis



Software Evolution Analysis



Software Evolution Analysis

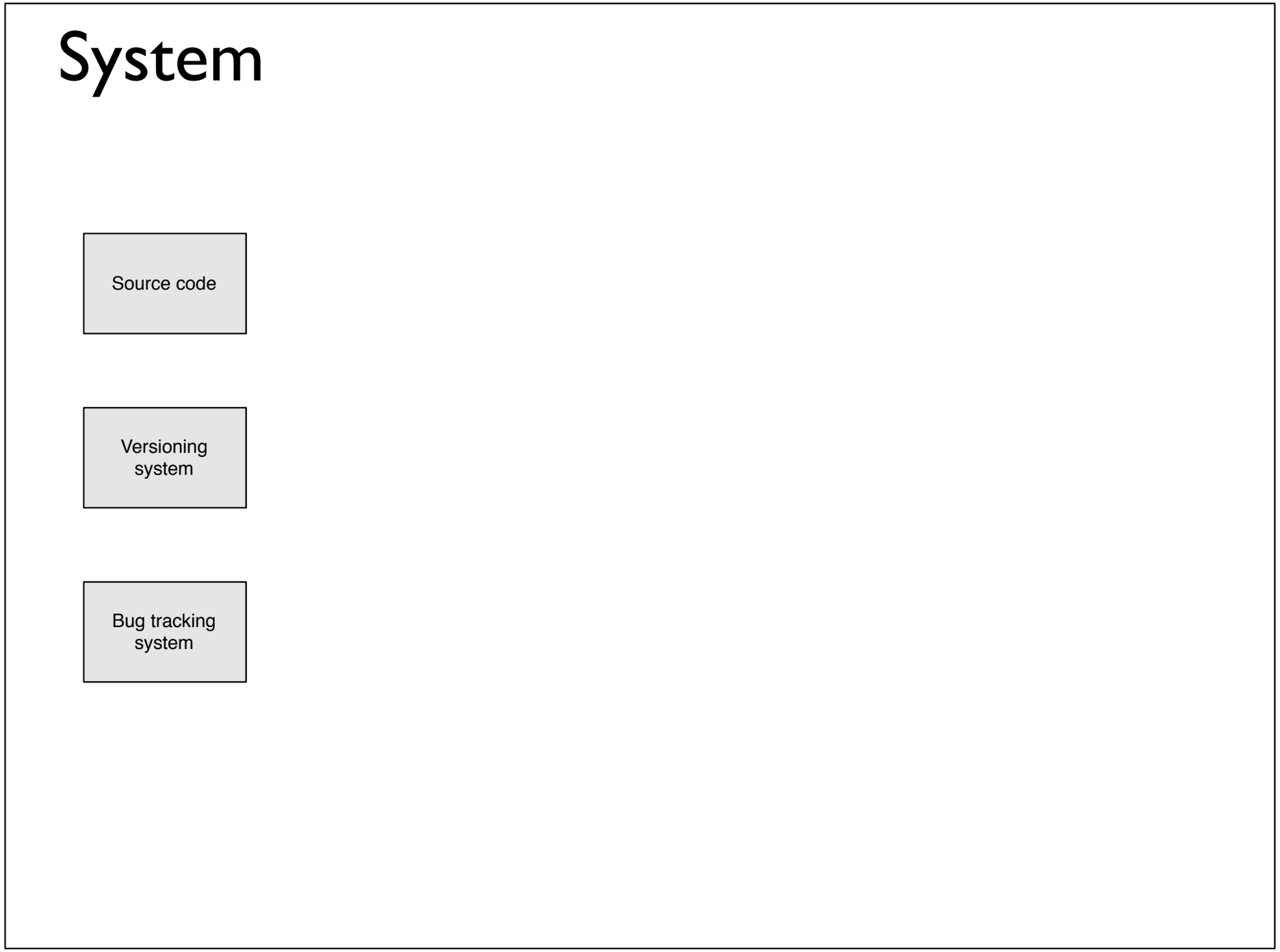


Software Evolution Analysis

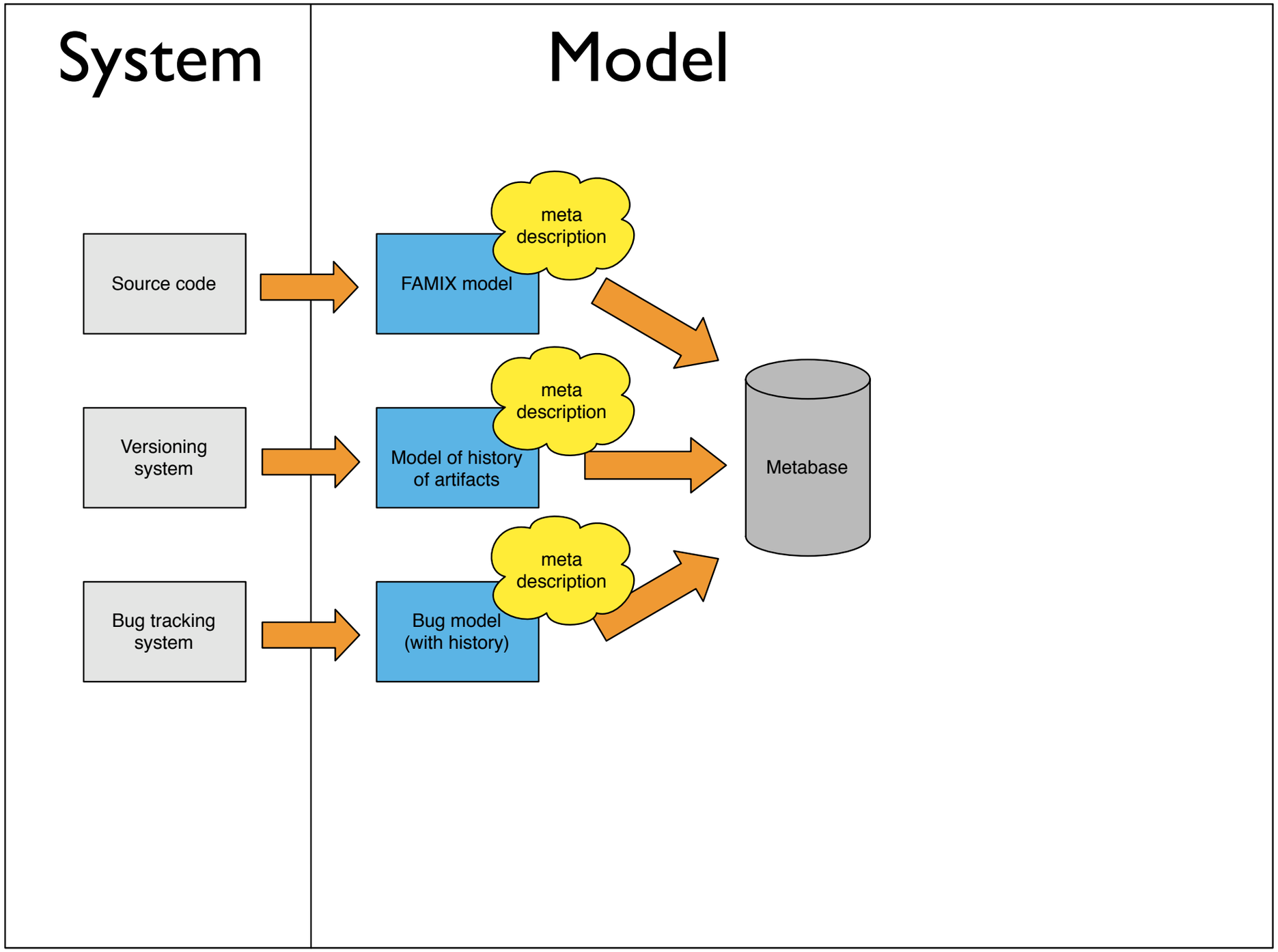
Goal

Detecting potential shortcoming in the architecture, design and logical structure of the system

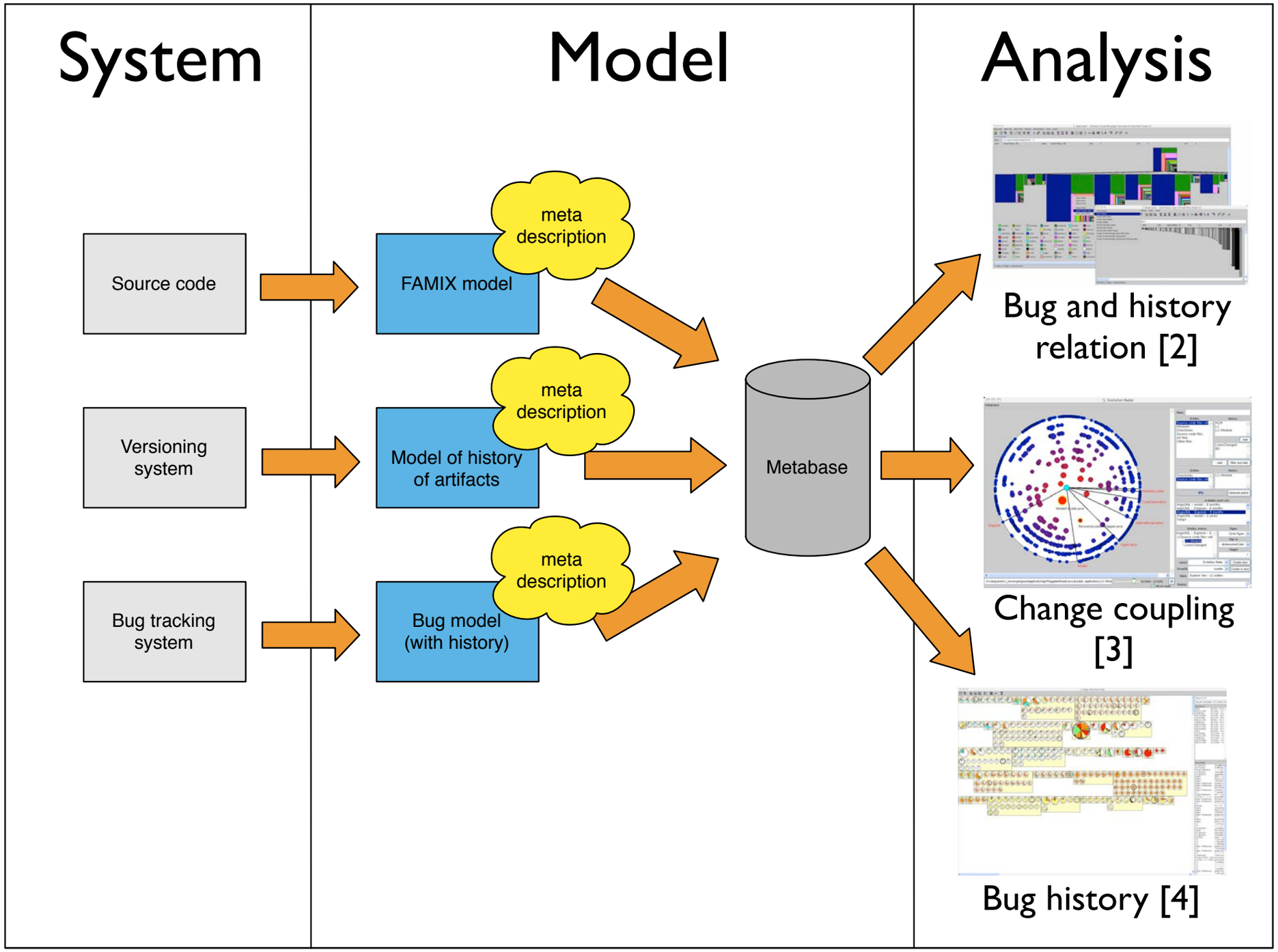
Our approach



Our approach



Our approach



[2] M. D'Ambros, Michele Lanza. *Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships*. CSMR 2006

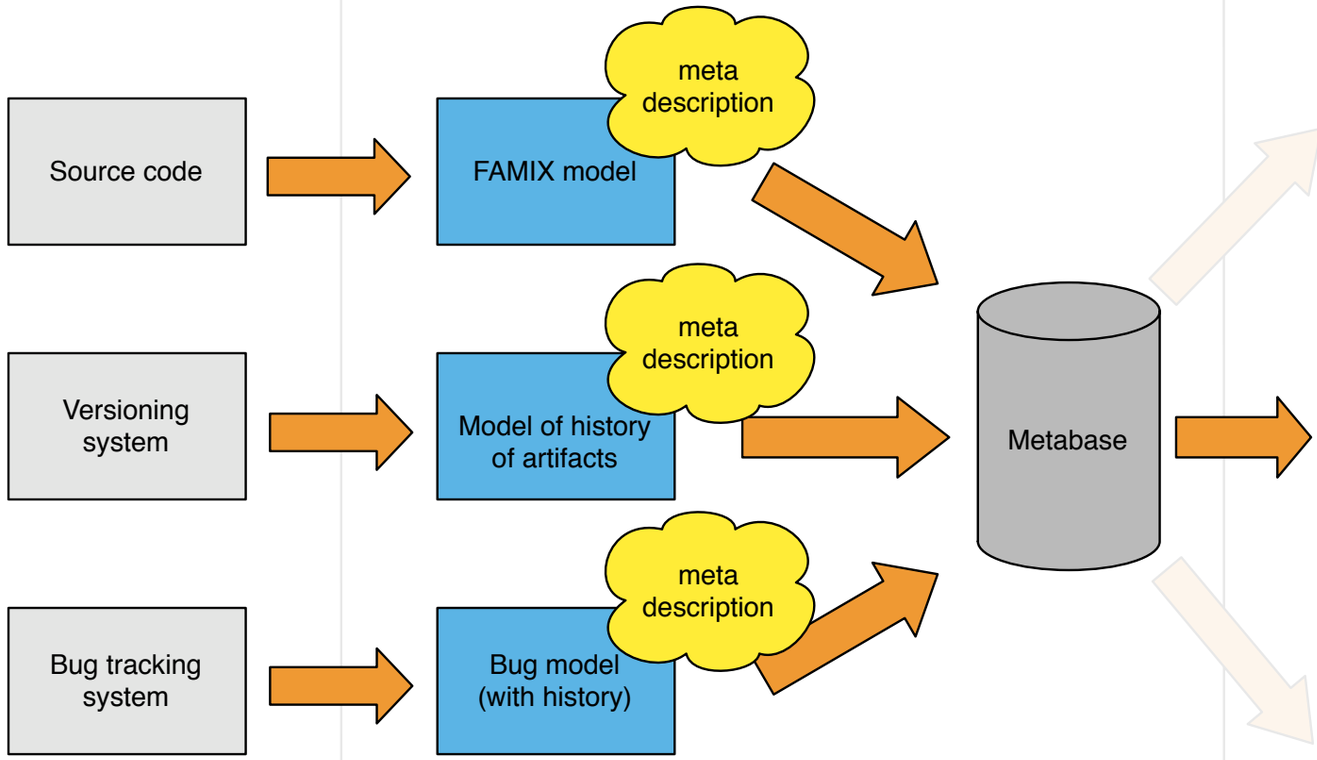
[3] M. D'Ambros, Michele Lanza. *Reverse Engineering with Logical Coupling*. WCRE 2006

[4] M. D'Ambros, Michele Lanza, Martin Pinzger. "*A Bug's Life*" - *Visualizing a Bug Database*. VISSOFT 2007

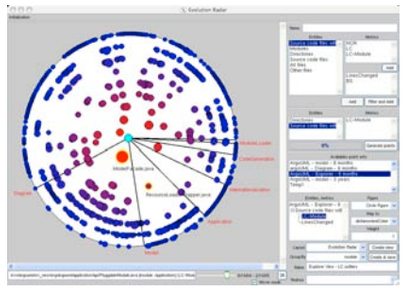
Our approach

System Model Analysis

Available as web service



Bug and history relation [2]



Change coupling [3]

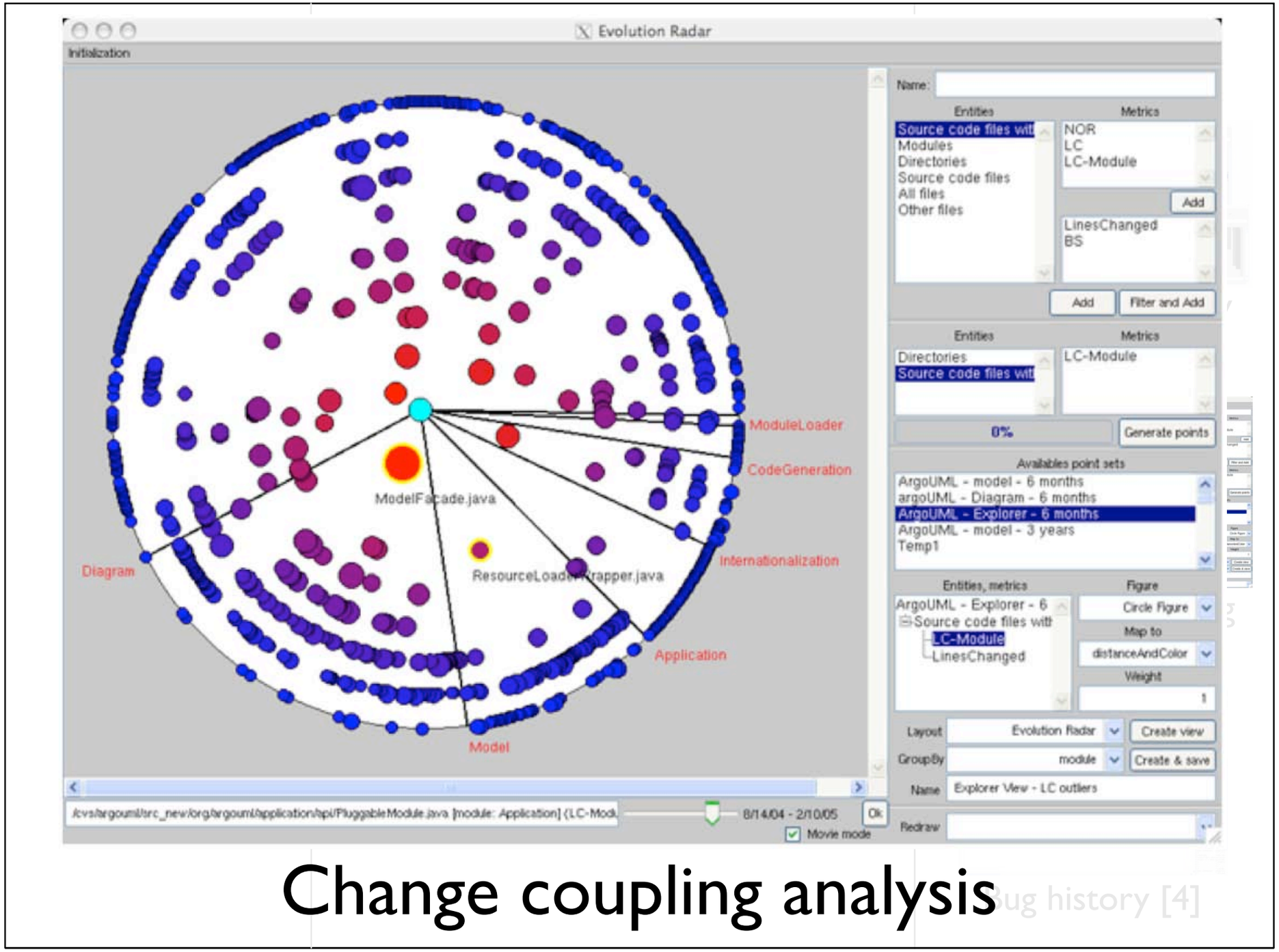
Bug history [4]

[2] M. D'Ambros, Michele Lanza. *Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships*. CSMR 2006

[3] M. D'Ambros, Michele Lanza. *Reverse Engineering with Logical Coupling*. WCRE 2006

[4] M. D'Ambros, Michele Lanza, Martin Pinzger. "*A Bug's Life*" - *Visualizing a Bug Database*. VISSOFT 2007

Our approach



Change coupling analysis

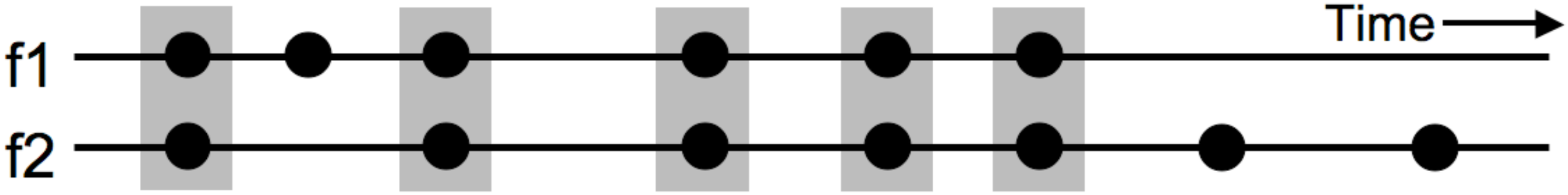
Bug history [4]

[2] M. D'Ambros, Michele Lanza. *Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships*. CSMR 2006

[3] M. D'Ambros, Michele Lanza. *Reverse Engineering with Logical Coupling*. WCRE 2006

[4] M. D'Ambros, Michele Lanza, Martin Pinzger. "*A Bug's Life*" - *Visualizing a Bug Database*. VISSOFT 2007

Change Coupling (CC)



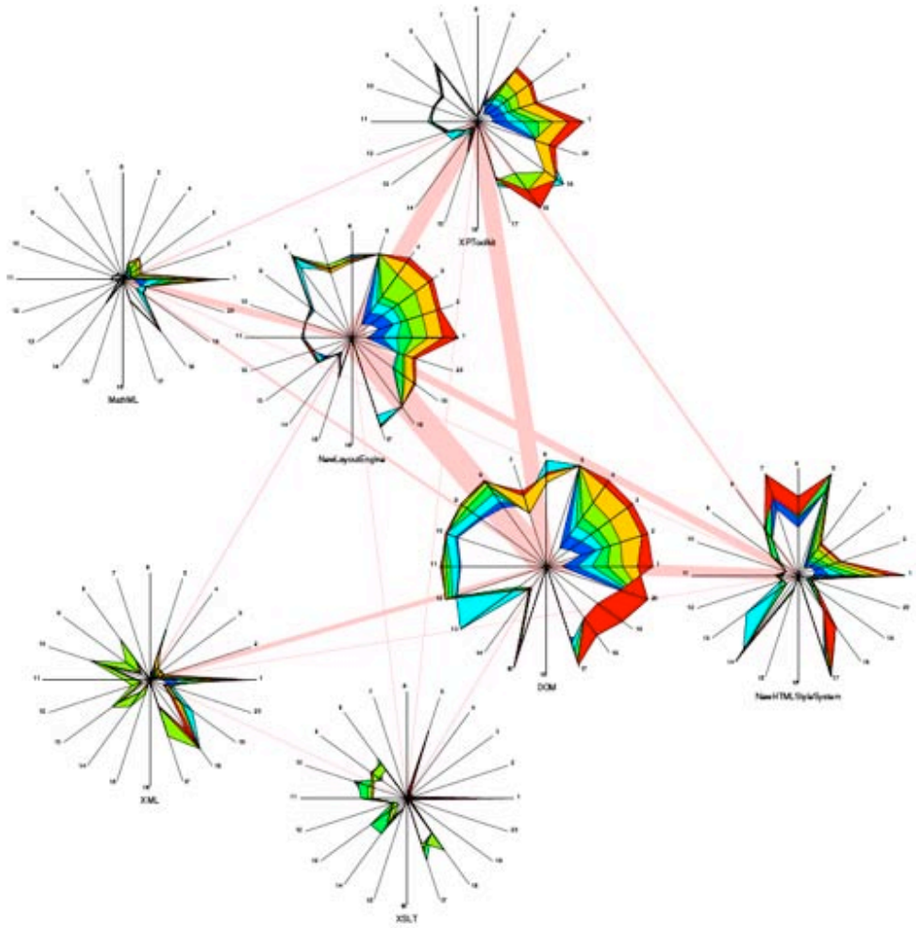
- Implicit dependencies between artifacts observed to change together
- Introduced by Gall *et. al.* in [5]
- Benefits
  - Lightweight
  - Visible only in the evolution, not in the code or documentation
  - Orthogonal to structural analysis

[5] Gall et. al. *Detection of Logical Coupling Based on Product Release History*. ICSM 1998

Current approaches to CC

Architecture level (*e.g.* [6])

- Dependencies among modules or subsystems
- Problem: Loss of detailed information



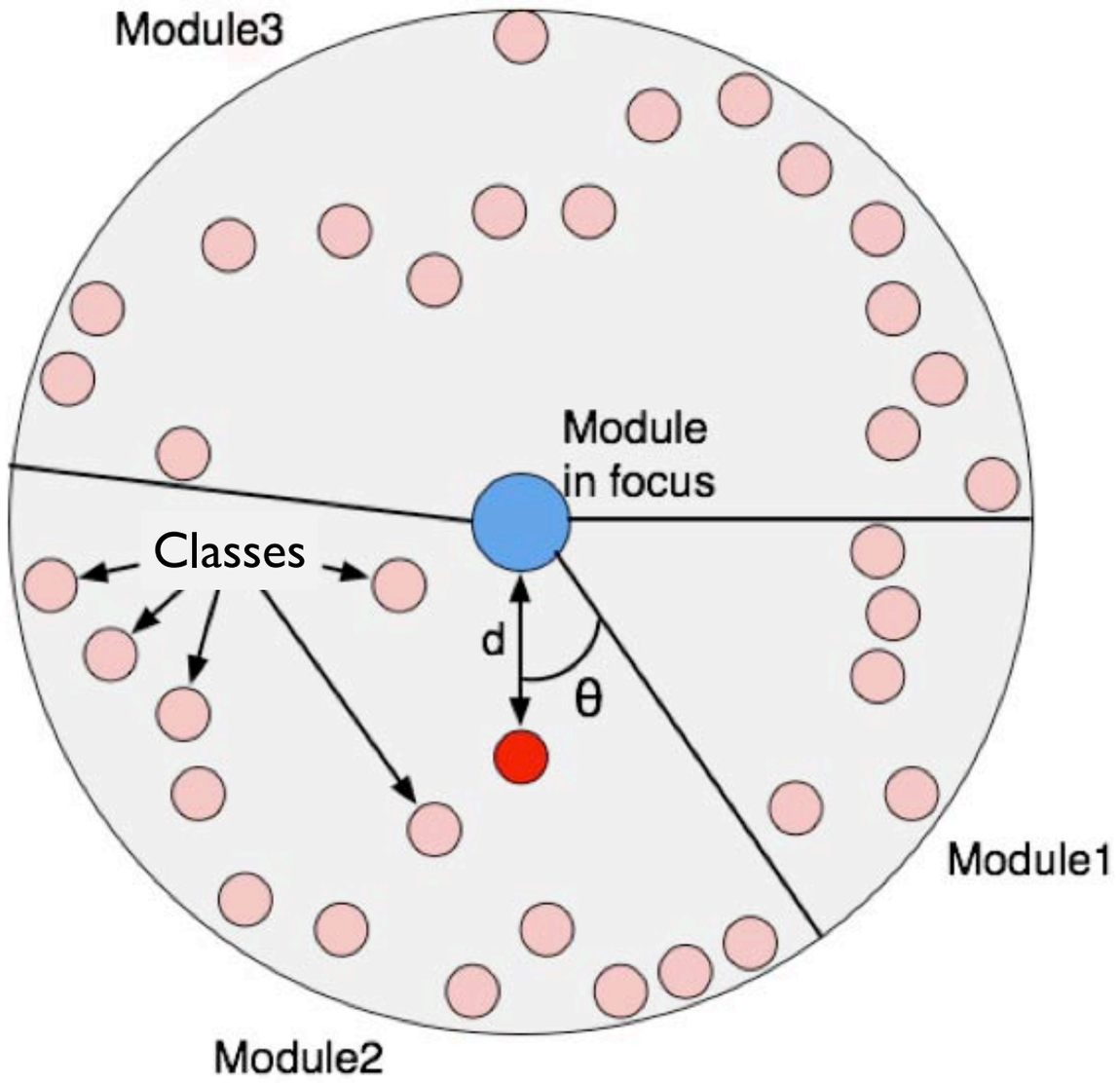
File (or finer) level (*e.g.* [7])

- Predict entities which are likely to be modified
- Problem: No global view of the system

[6] *Pinzger et. al.* Visualizing Multiple Evolution Metrics. SoftVis 2005
[7] *Zimmermann et. al.* Mining version histories to guide software changes. ICSE 2004

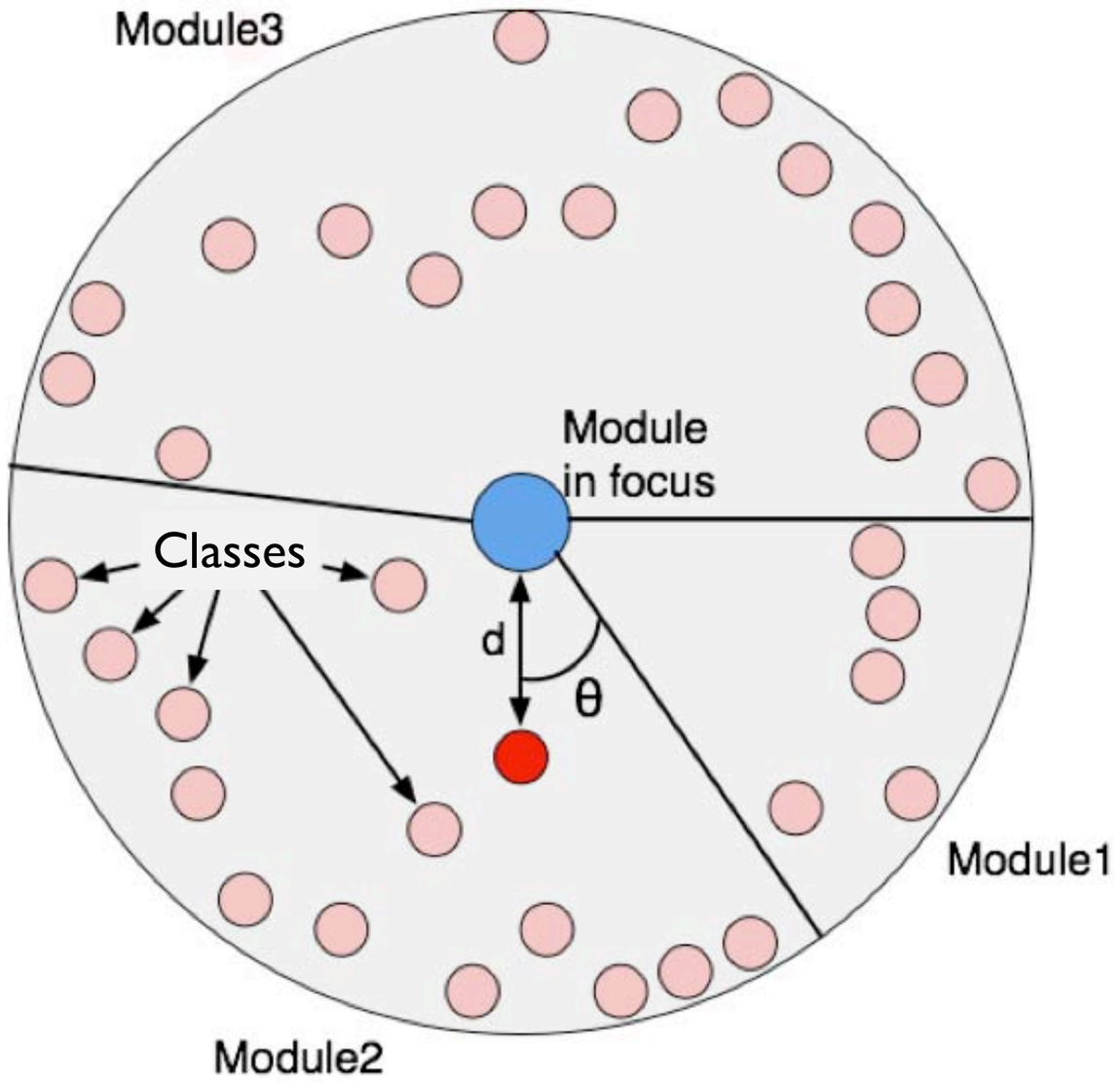
The Evolution Radar

- The module in focus (or reference module) is placed in the center



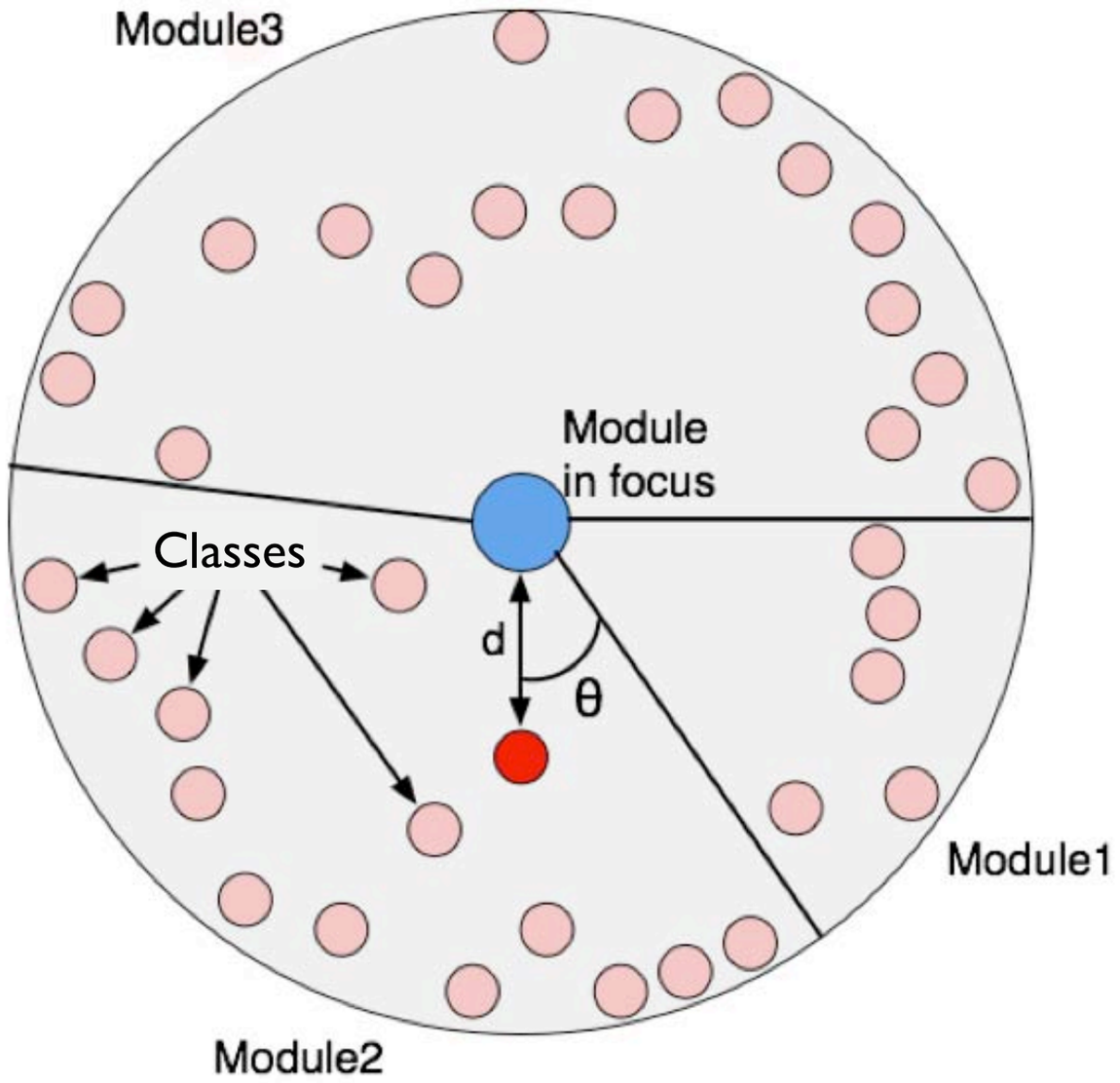
The Evolution Radar

- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors



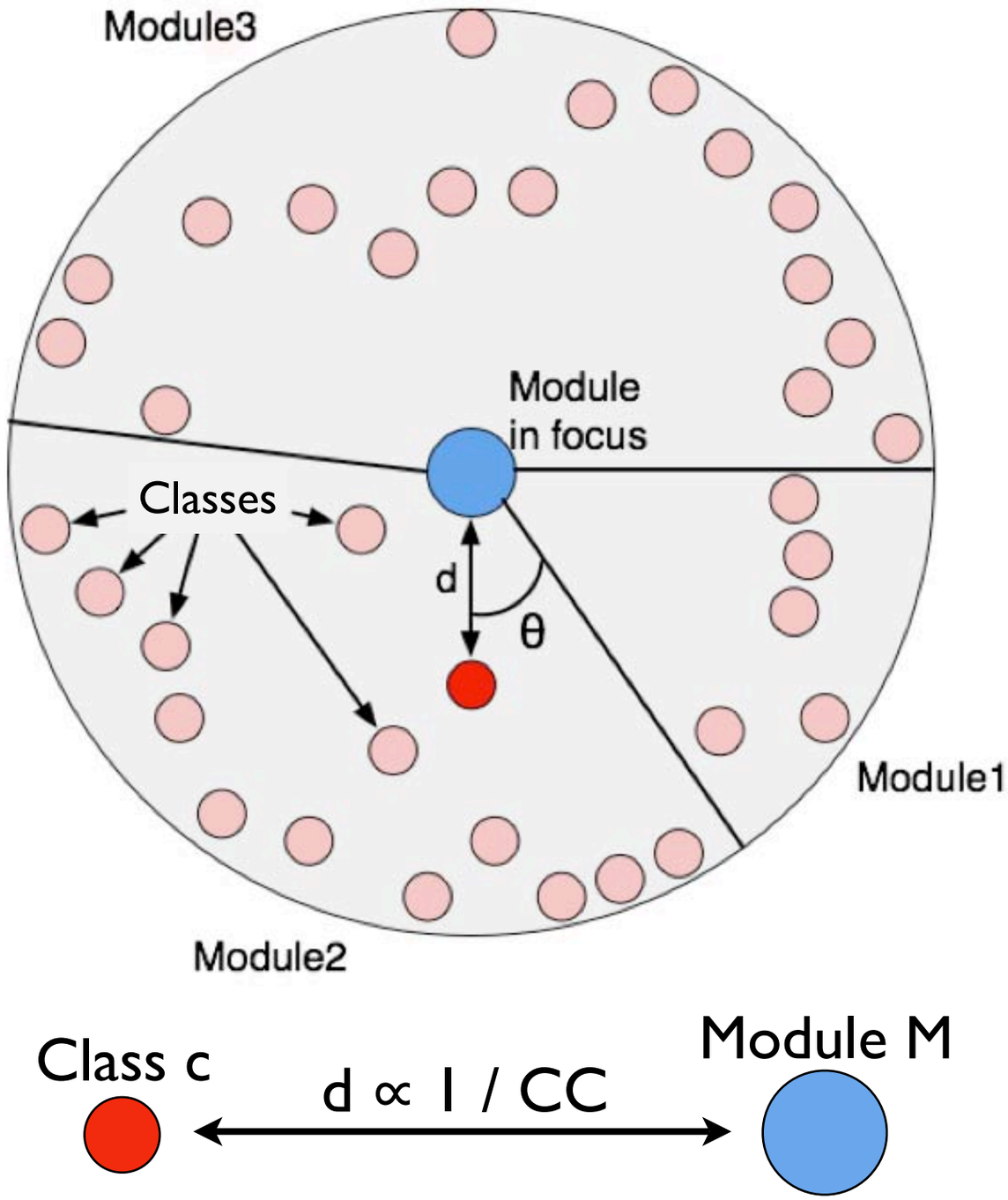
The Evolution Radar

- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors
- For each module all its classes are rendered as colored circles and positioned using polar coordinates:



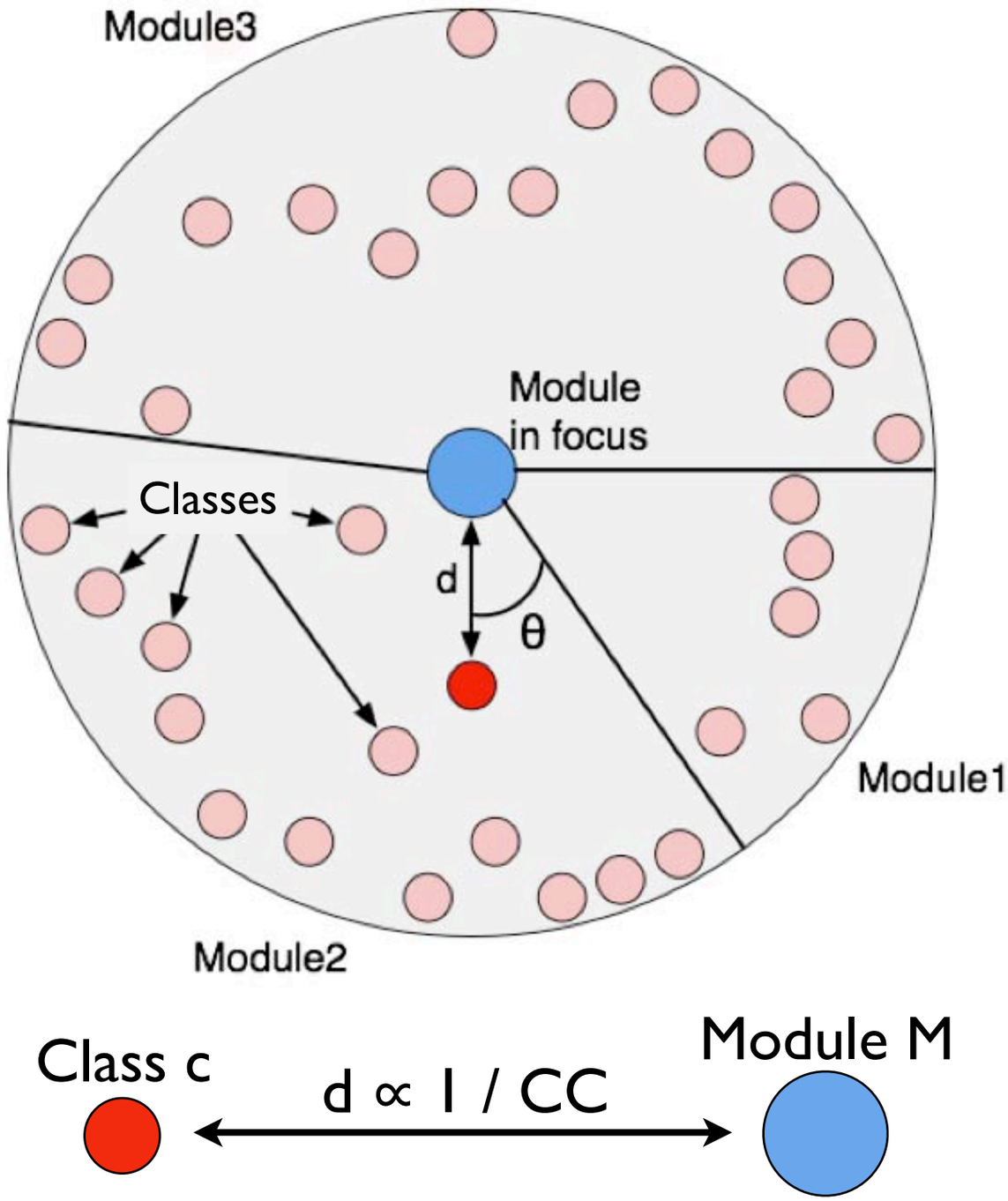
The Evolution Radar

- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors
- For each module all its classes are rendered as colored circles and positioned using polar coordinates:
  - **d**: inverse proportional to CC



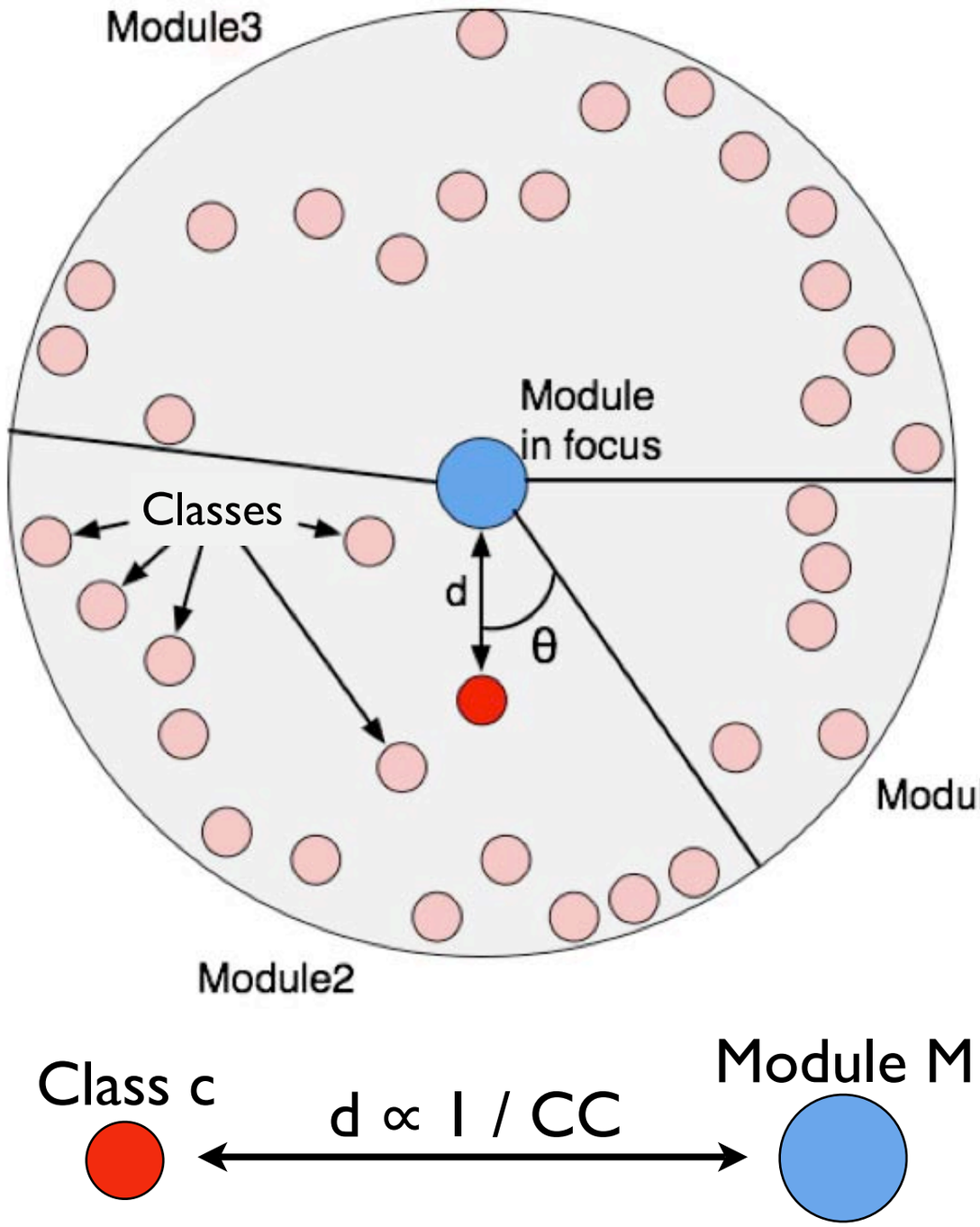
The Evolution Radar

- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors
- For each module all its classes are rendered as colored circles and positioned using polar coordinates:
  - **d**: inverse proportional to CC
  - **θ**: alphabetical sorting and uniform distribution



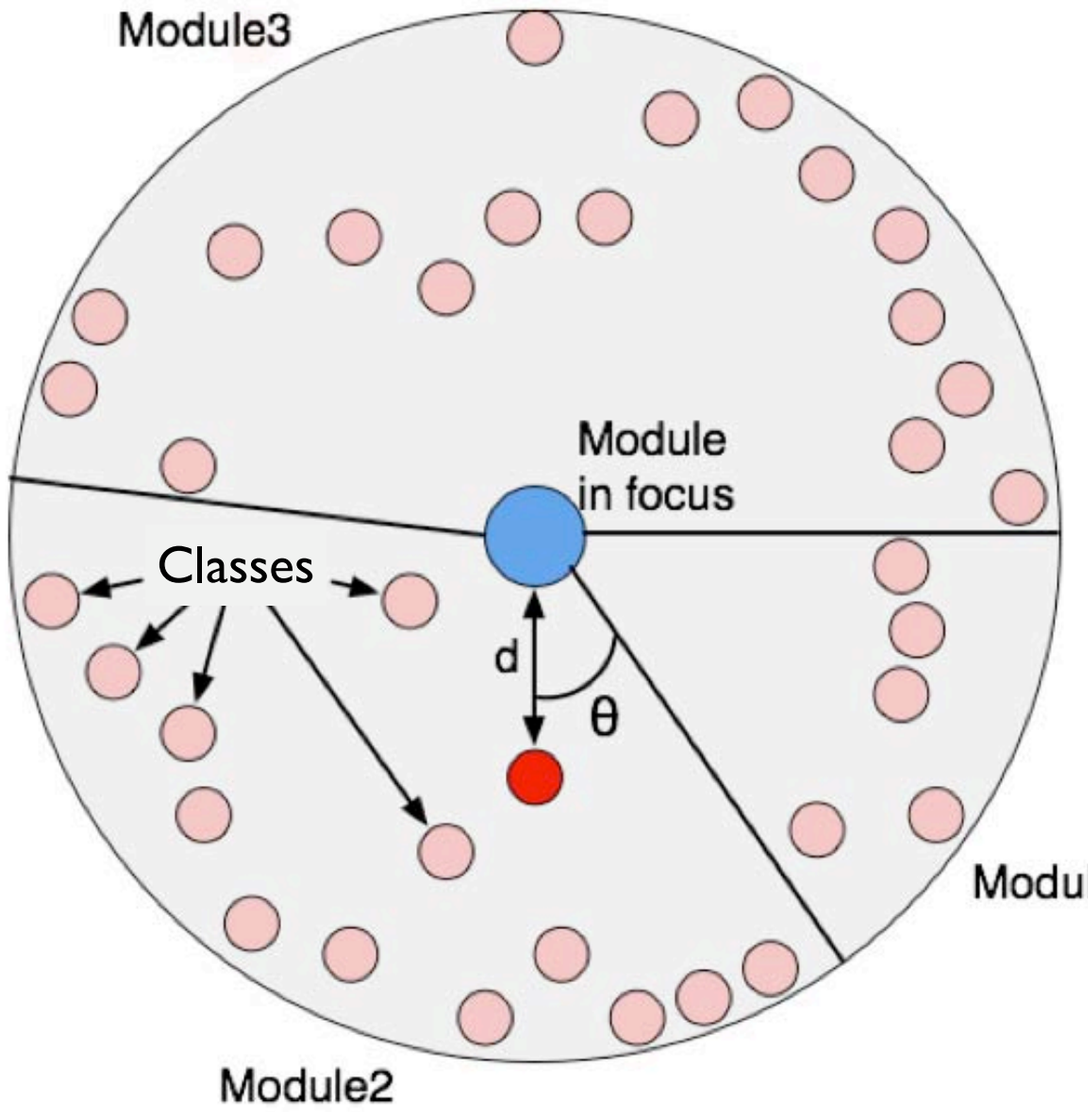
The Evolution Radar

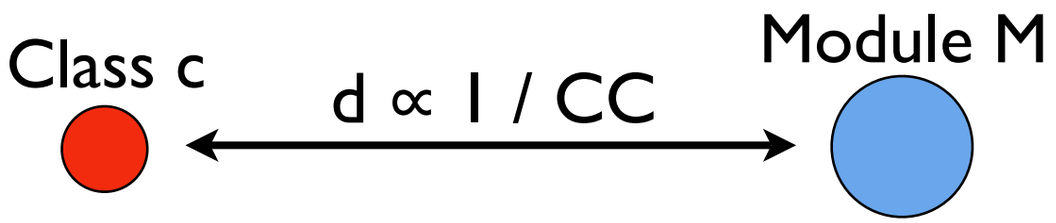
- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors
- For each module all its classes are rendered as colored circles and positioned using polar coordinates:
  - **d**: inverse proportional to CC
  - **θ**: alphabetical sorting and uniform distribution
- Metrics can be mapped on the size and color of figures



The Evolution Radar

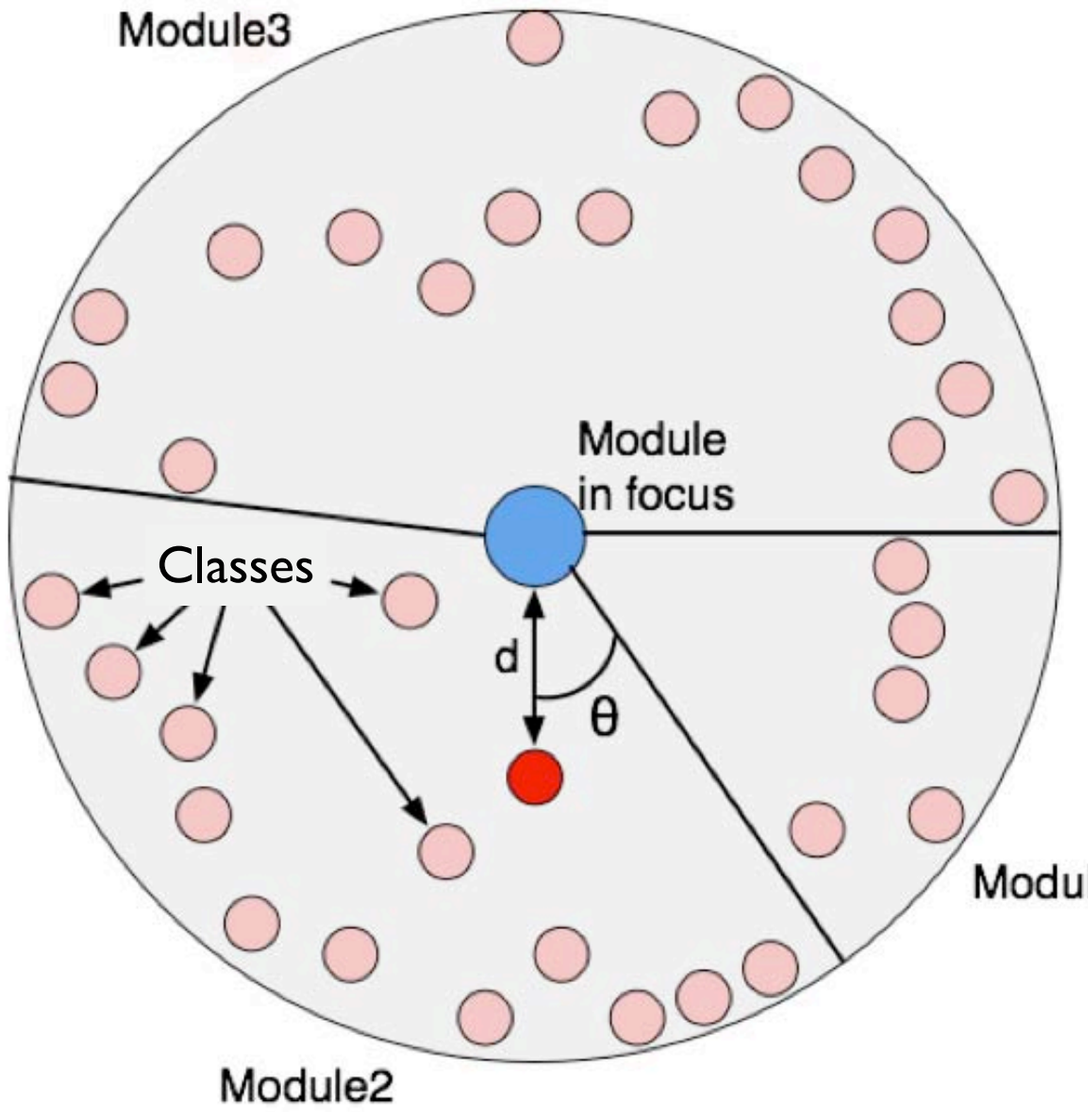
- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors
- For each module all its classes are rendered as colored circles and positioned using polar coordinates:
  - **d**: inverse proportional to CC
  - **θ**: alphabetical sorting and uniform distribution
- Metrics can be mapped on the size and color of figures
- CC between two classes is the number of “shared” commits

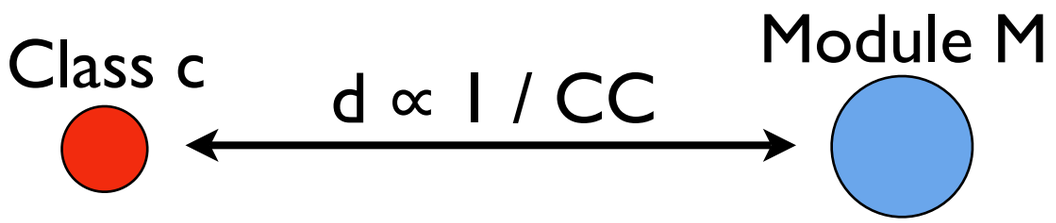




The Evolution Radar

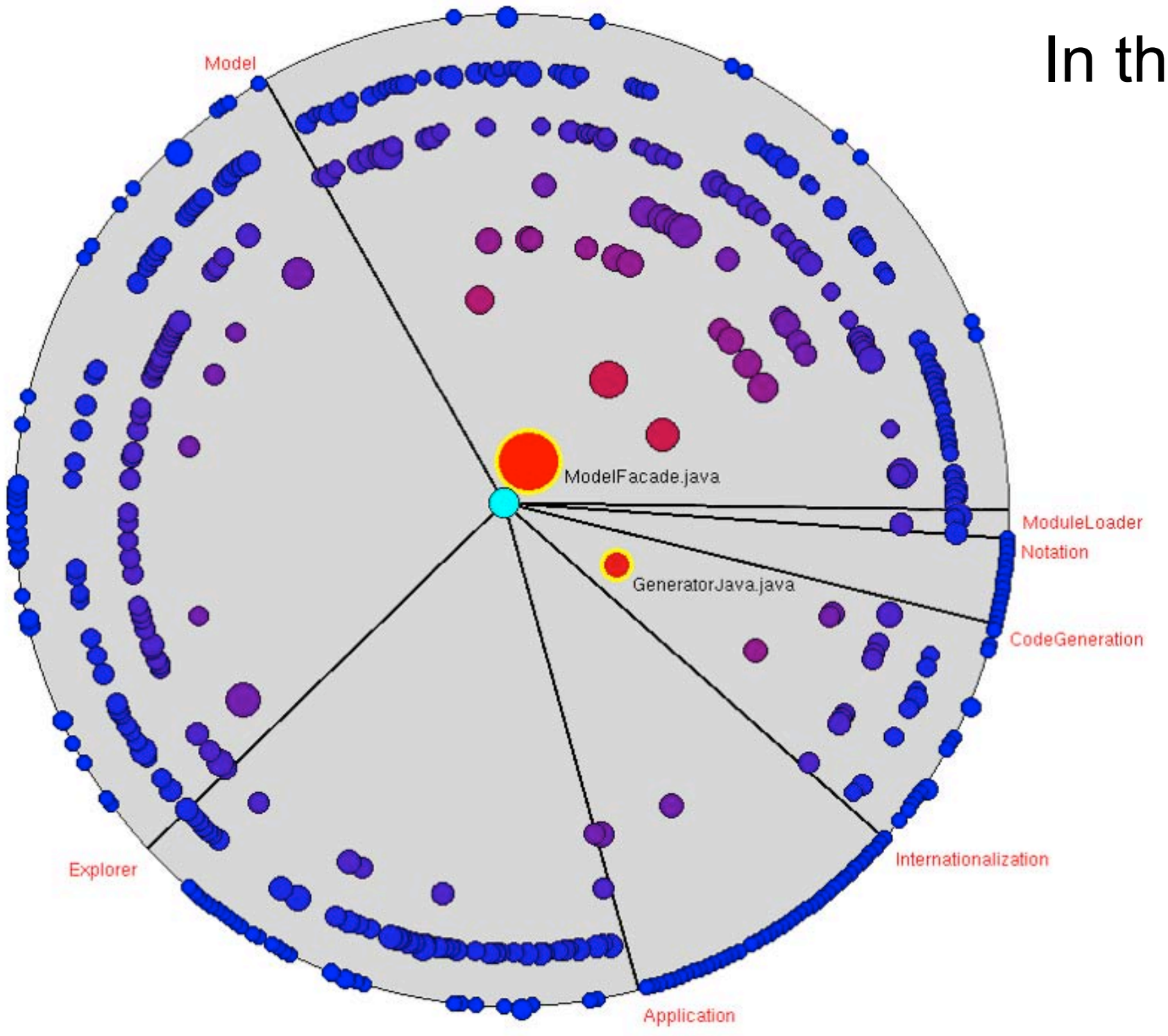
- The module in focus (or reference module) is placed in the center
- All the other modules are shown as sectors
- For each module all its classes are rendered as colored circles and positioned using polar coordinates:
  - **d**: inverse proportional to CC
  - **θ**: alphabetical sorting and uniform distribution
- Metrics can be mapped on the size and color of figures
- CC between two classes is the number of “shared” commits
- CC between a class and a module is defined by means of a group operator





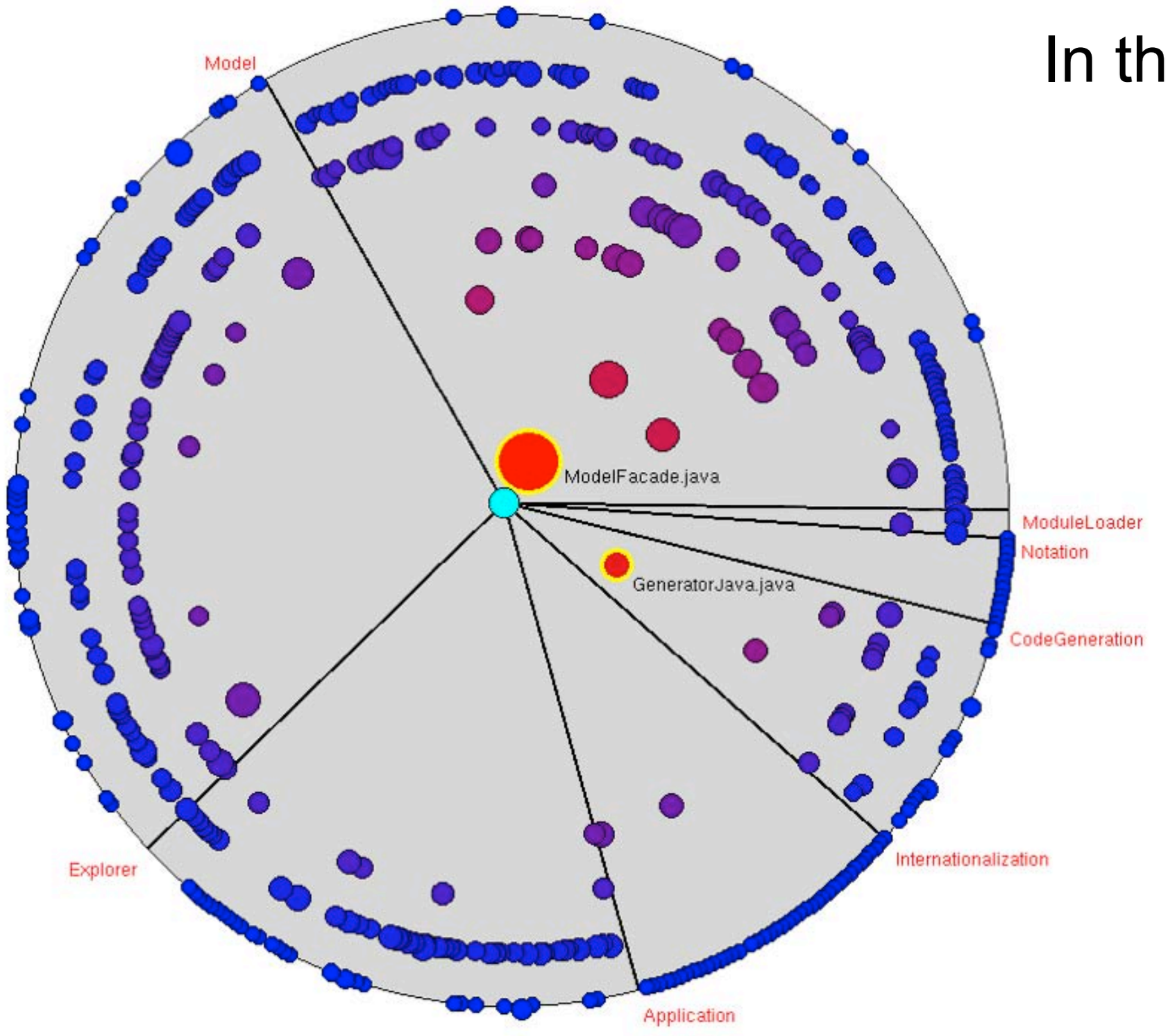
$$CC(M,c) = \max_{c_i \in M} CC(c_i,c)$$

Evolution Radar Exemplified



In the radar we can see:

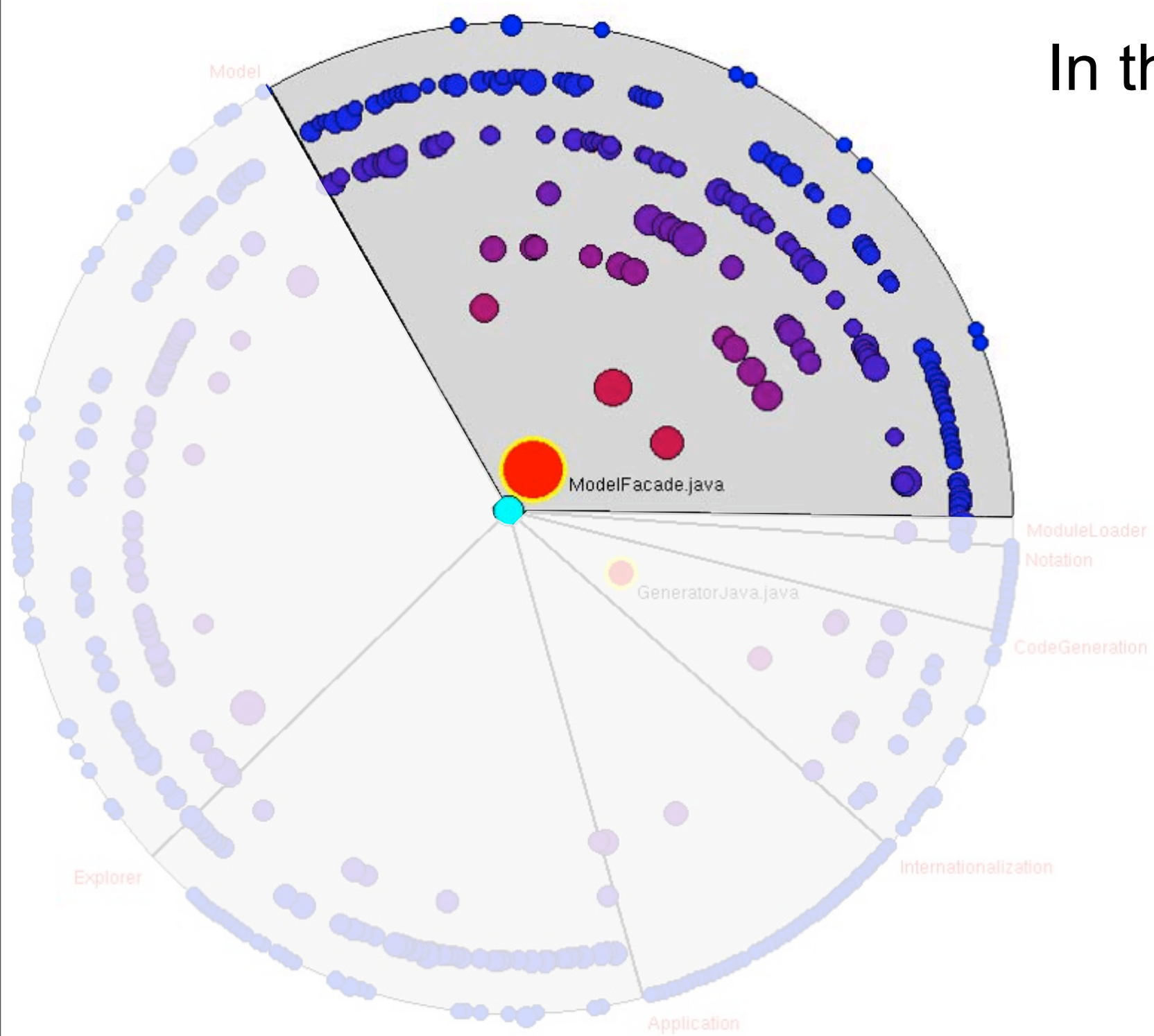
Evolution Radar Exemplified



In the radar we can see:

- CC between a module and all the other module

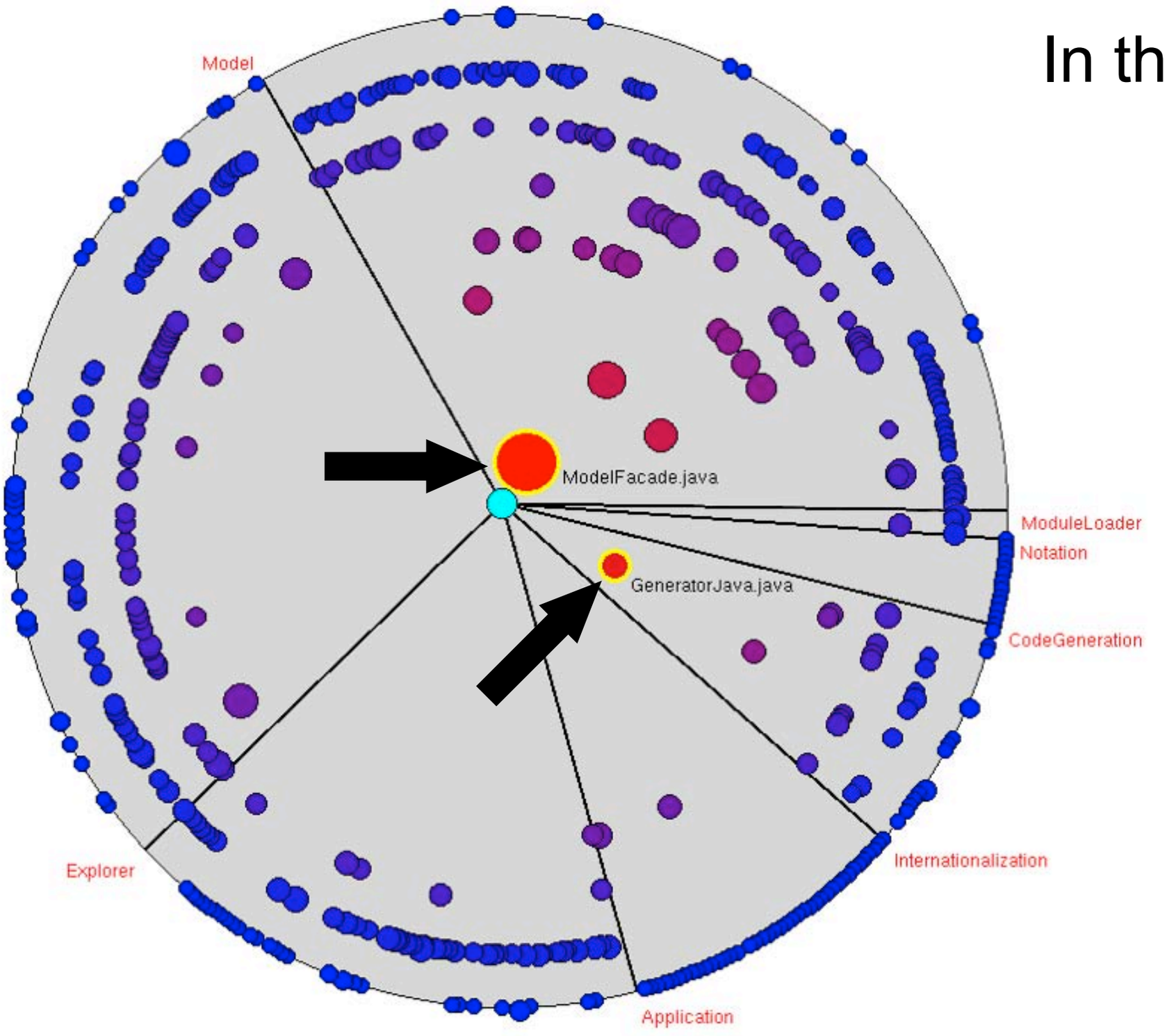
Evolution Radar Exemplified



In the radar we can see:

- CC between a module and all the other module
- How the coupling is structured in terms of classes

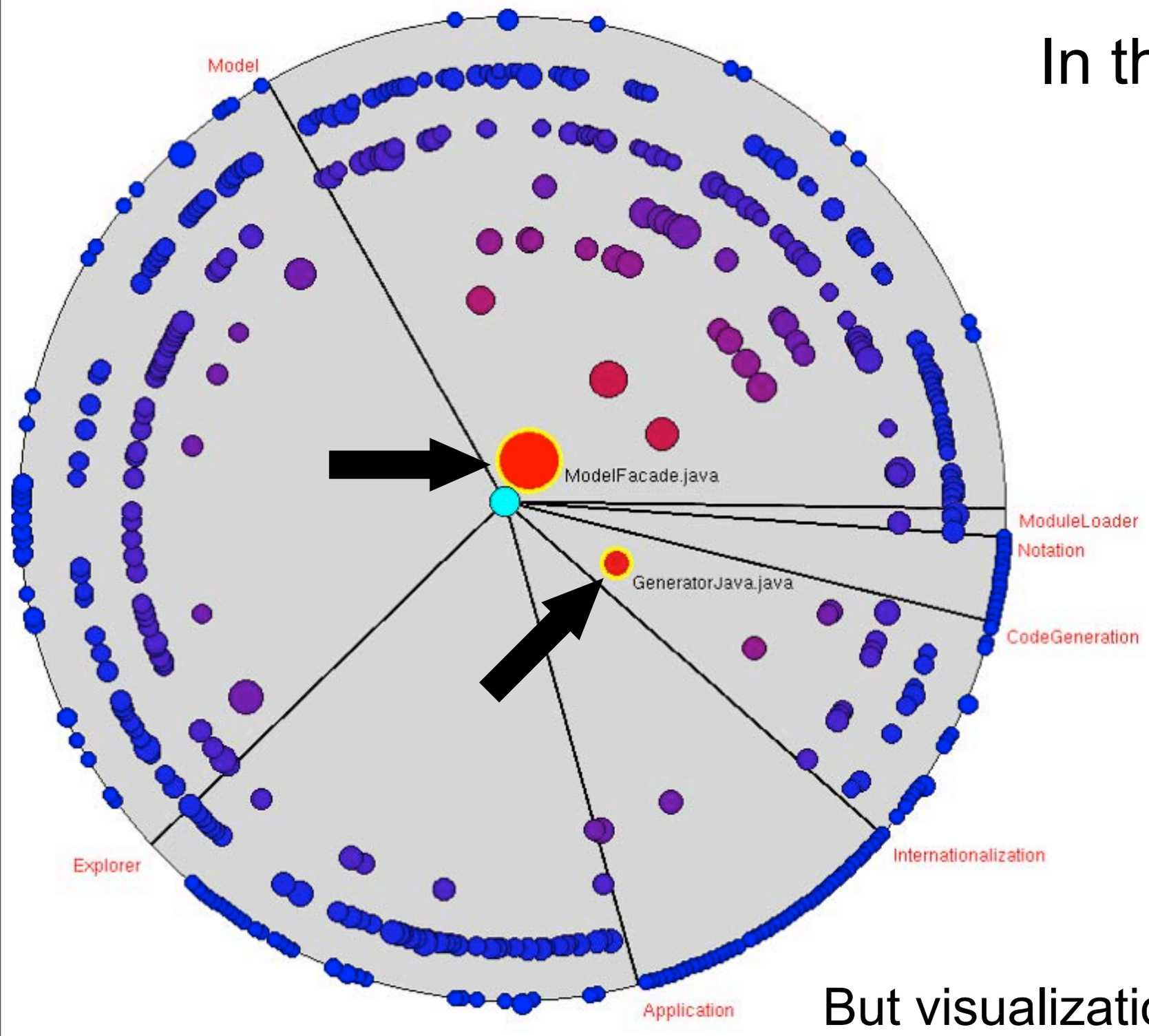
Evolution Radar Exemplified



In the radar we can see:

- CC between a module and all the other module
- How the coupling is structured in terms of classes
- Classes most coupled with the module

Evolution Radar Exemplified



In the radar we can see:

- CC between a module and all the other module
- How the coupling is structured in terms of classes
- Classes most coupled with the module

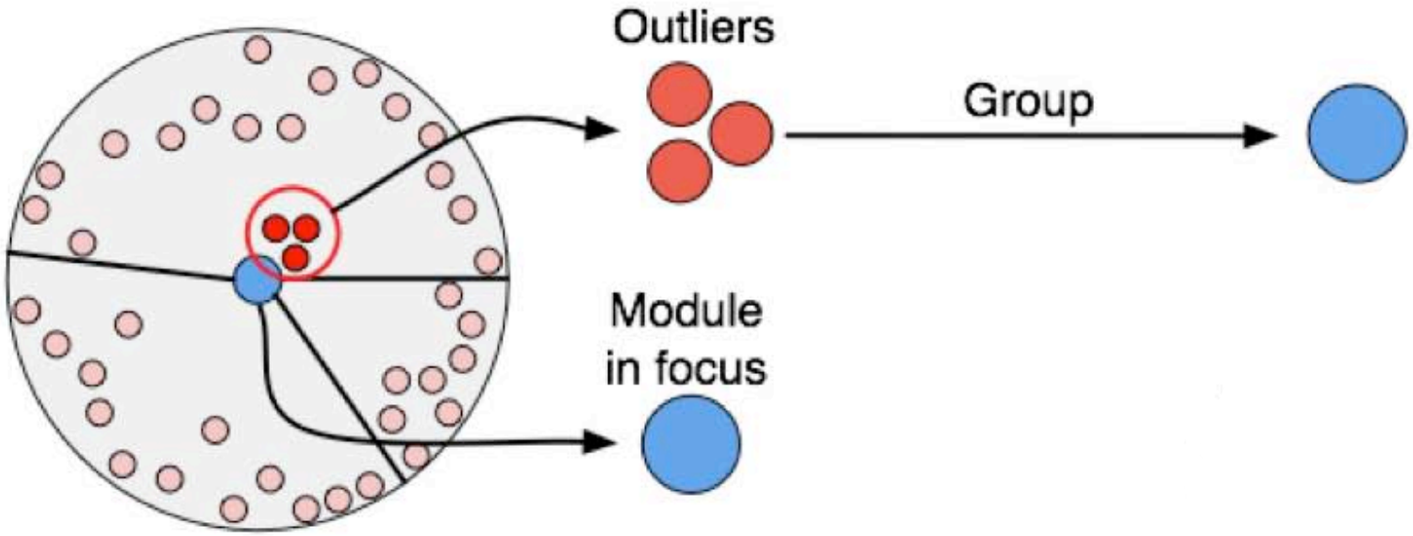
But visualization is not a silver bullet...

Interacting with the radar

- Basic interaction
  - Any entity in the visualization (classes and module in focus) can be inspected
    - Source code
    - Commit-related information
    - Contents
- Advanced interaction
  1. Spawning
  2. Moving trough time
  3. Tracking

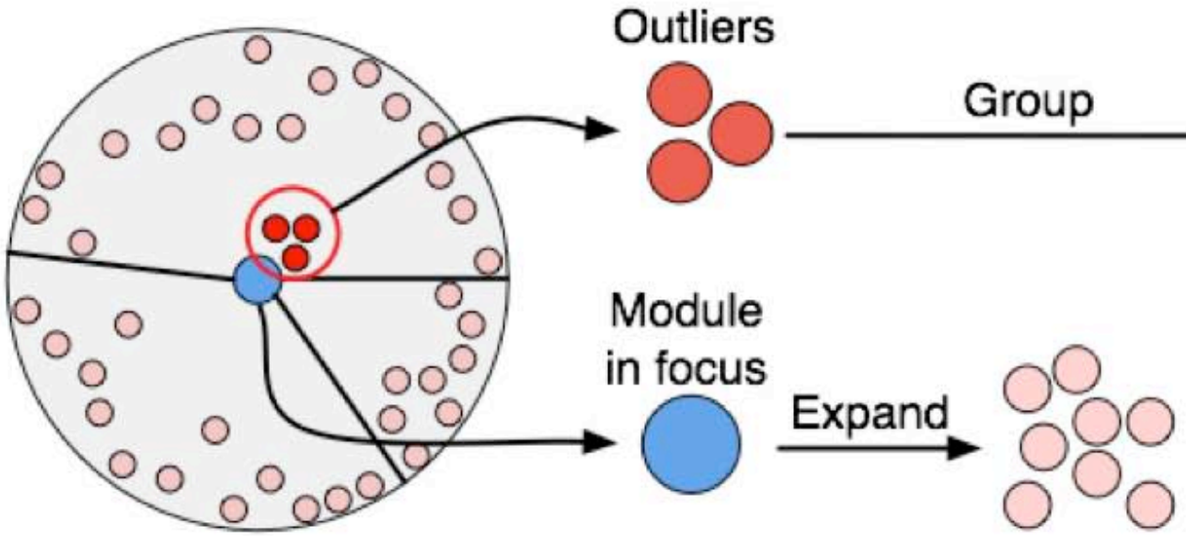
Spawning

To understand which files are coupled with selected files in the module in focus



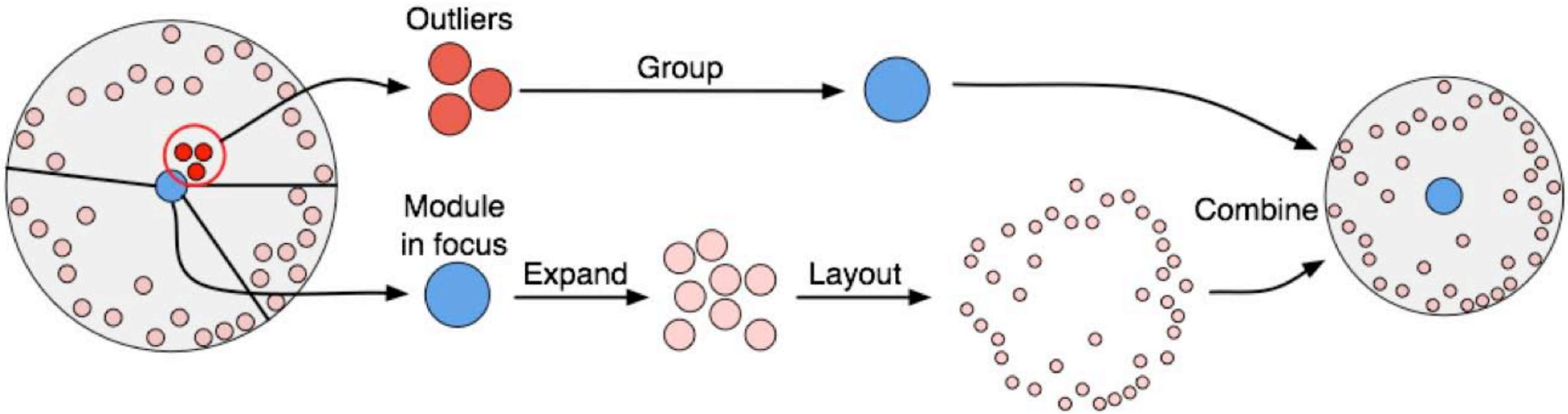
Spawning

To understand which files are coupled with selected files in the module in focus



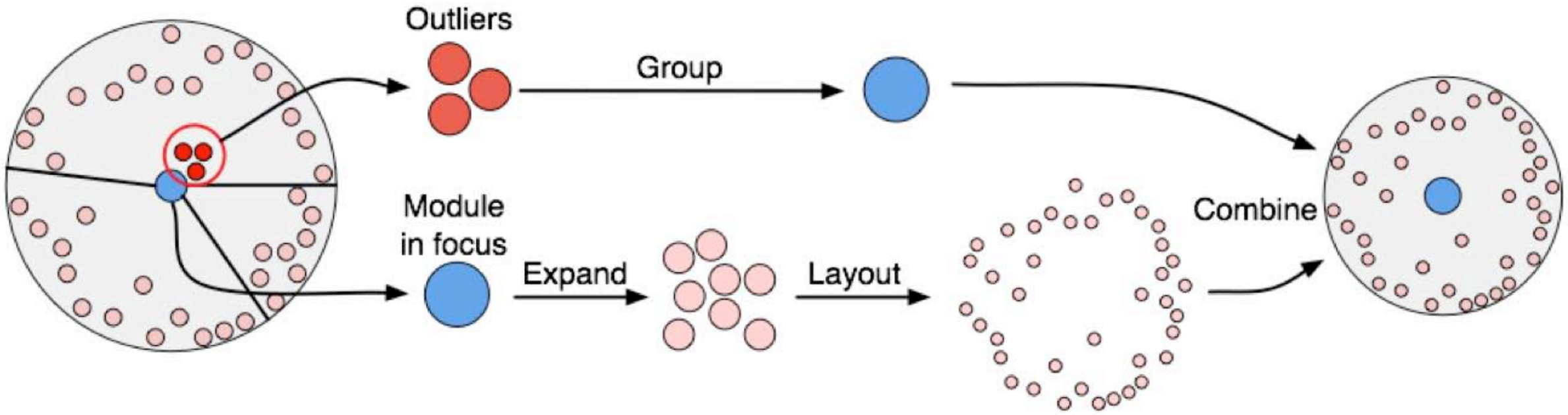
Spawning

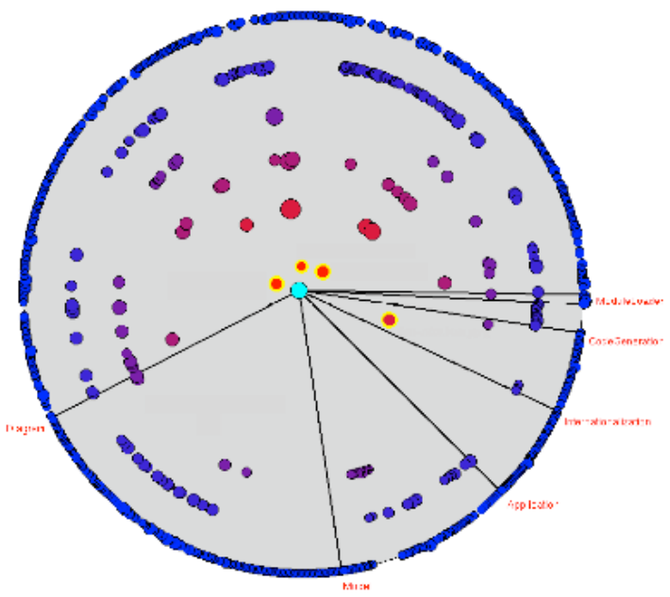
To understand which files are coupled with selected files in the module in focus



Spawning

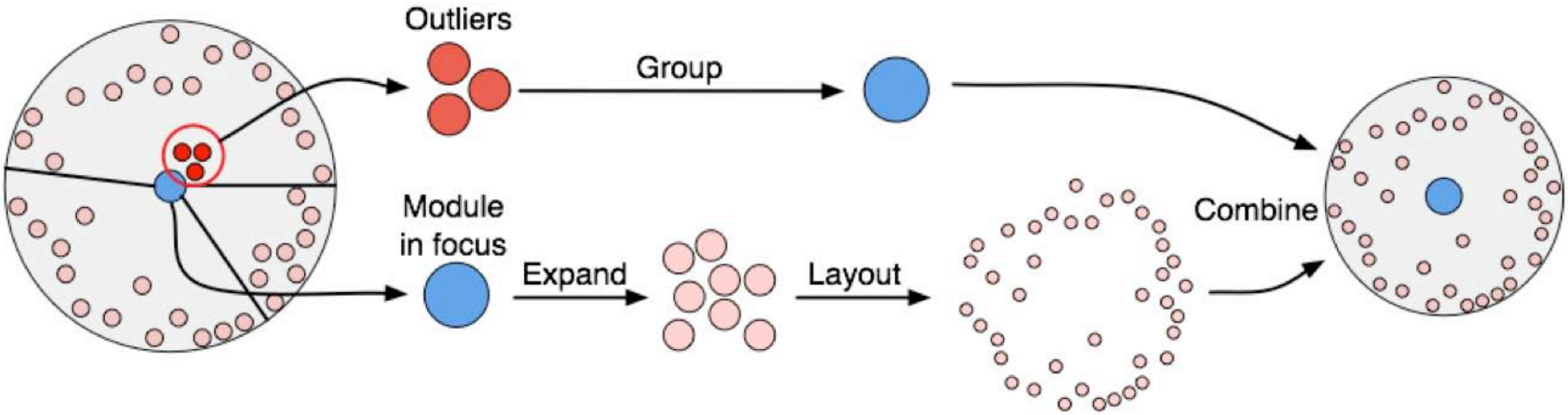
To understand which files are coupled with selected files in the module in focus





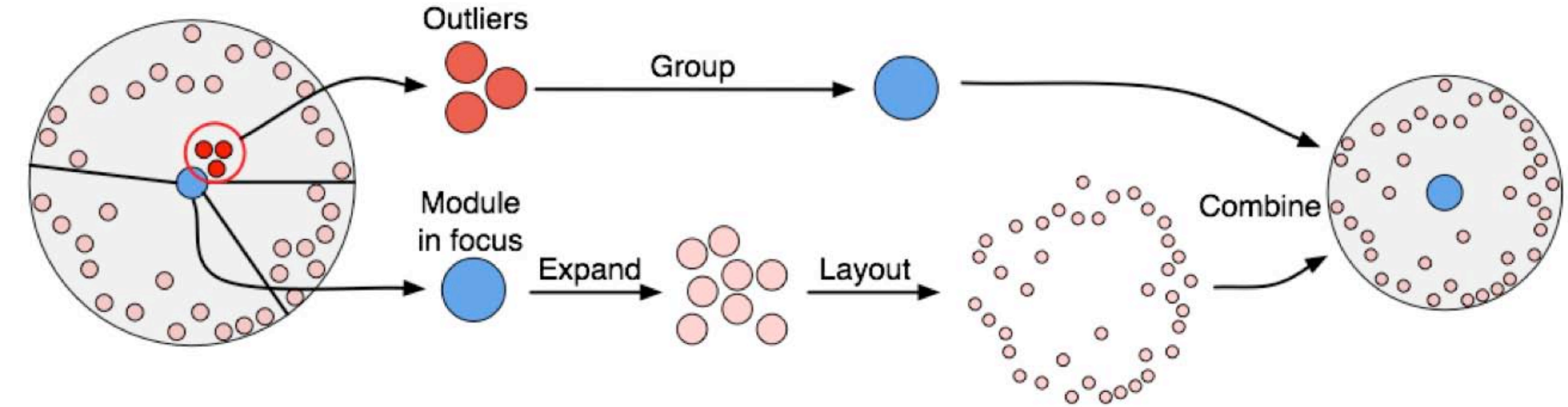
Spawning

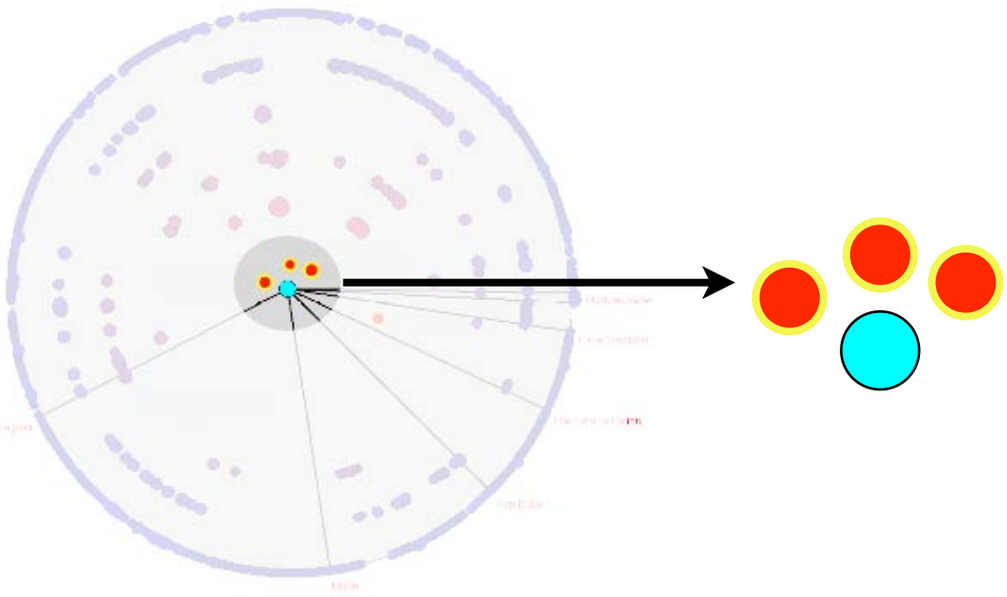
To understand which files are coupled with selected files in the module in focus



Spawning

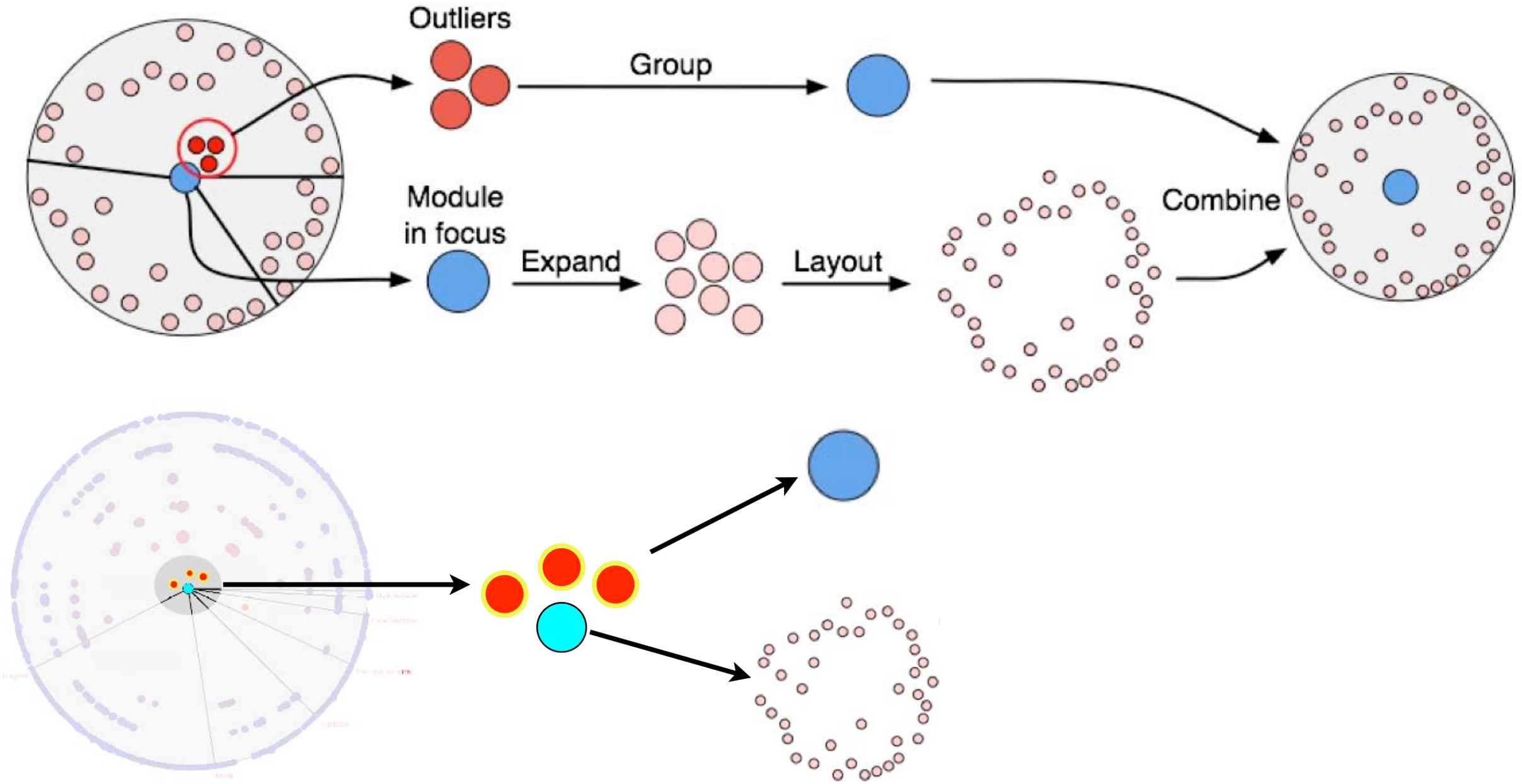
To understand which files are coupled with selected files in the module in focus





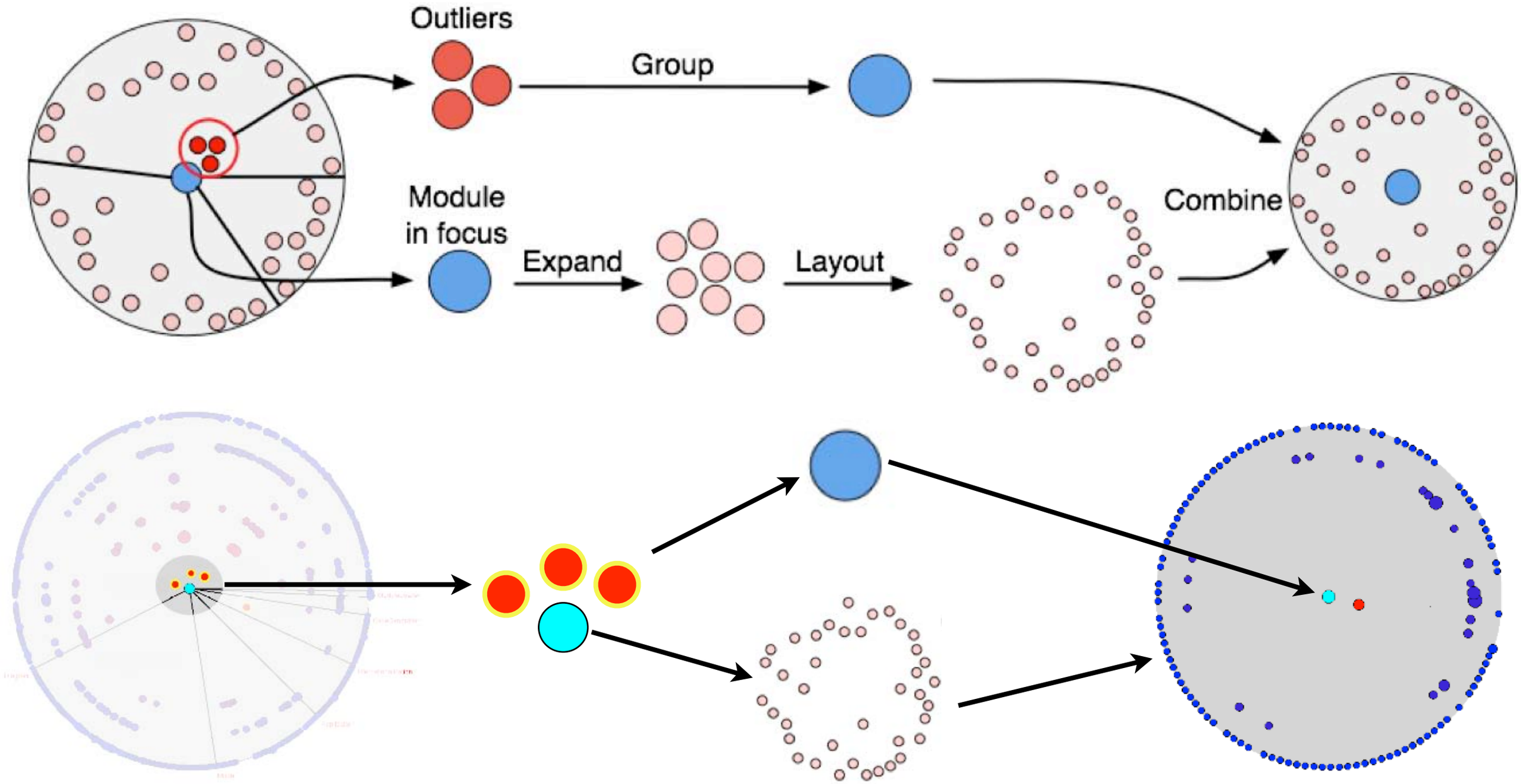
Spawning

To understand which files are coupled with selected files in the module in focus



Spawning

To understand which files are coupled with selected files in the module in focus

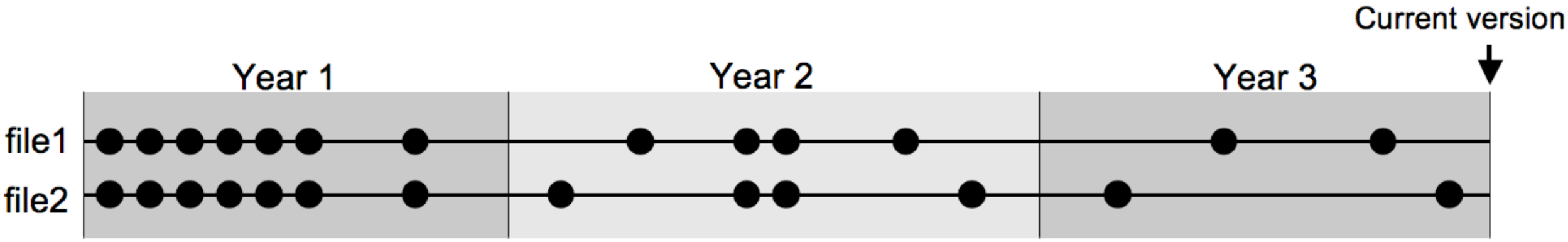


Moving Through Time & Tracking

Problem: The coupling value depends on the time window considered

Moving Through Time & Tracking

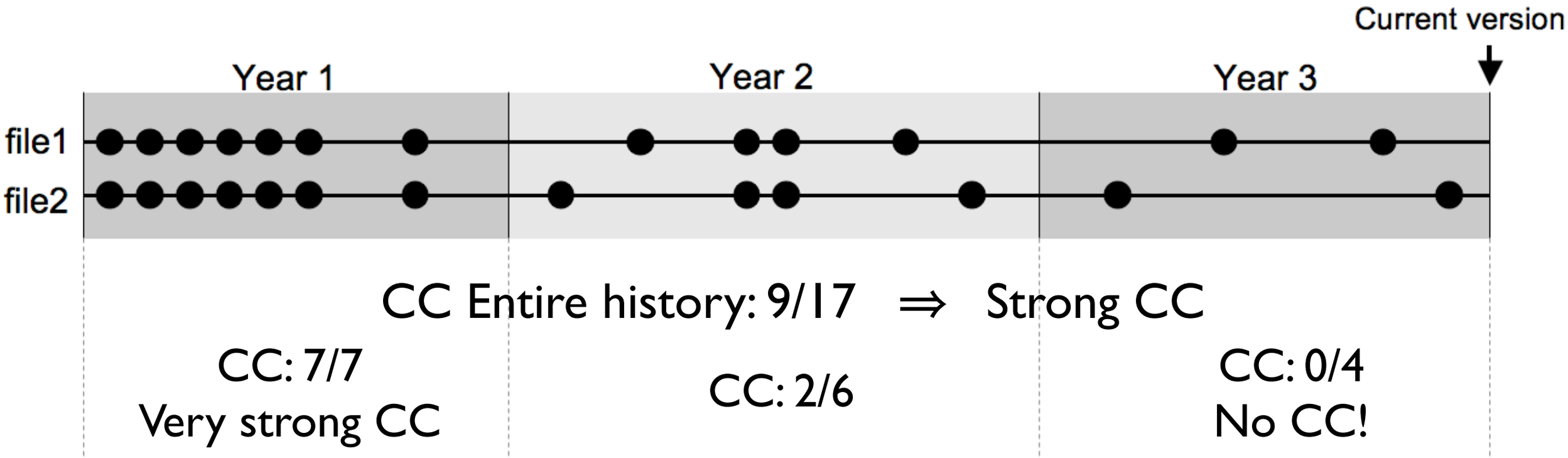
Problem: The coupling value depends on the time window considered



CC Entire history: 9/17 ⇒ Strong CC

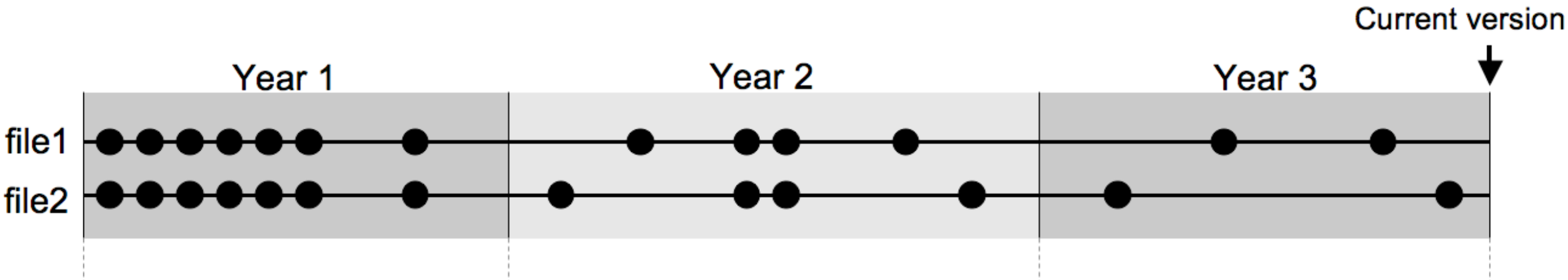
Moving Through Time & Tracking

Problem: The coupling value depends on the time window considered



Moving Through Time & Tracking

Problem: The coupling value depends on the time window considered

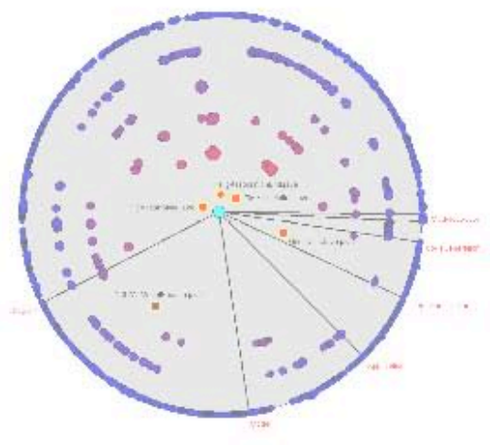


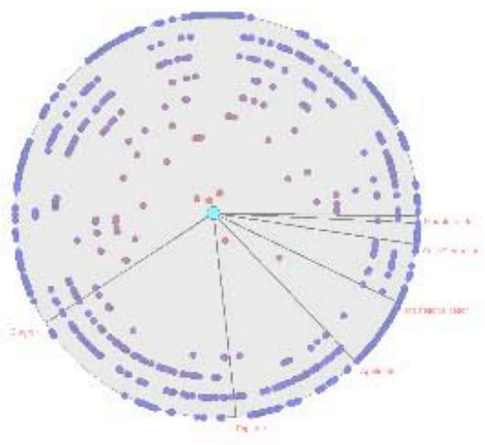
CC Entire history: 9/17 ⇒ Strong CC

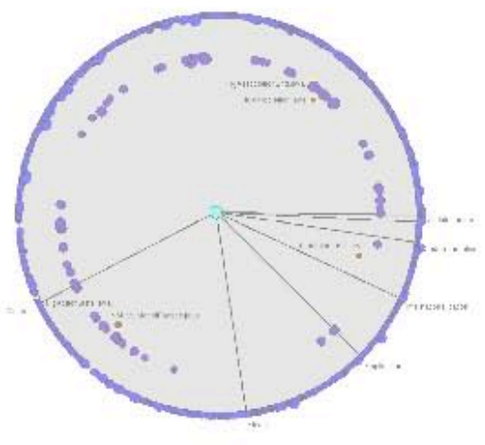
CC: 7/7
Very strong CC

CC: 2/6

CC: 0/4
No CC!

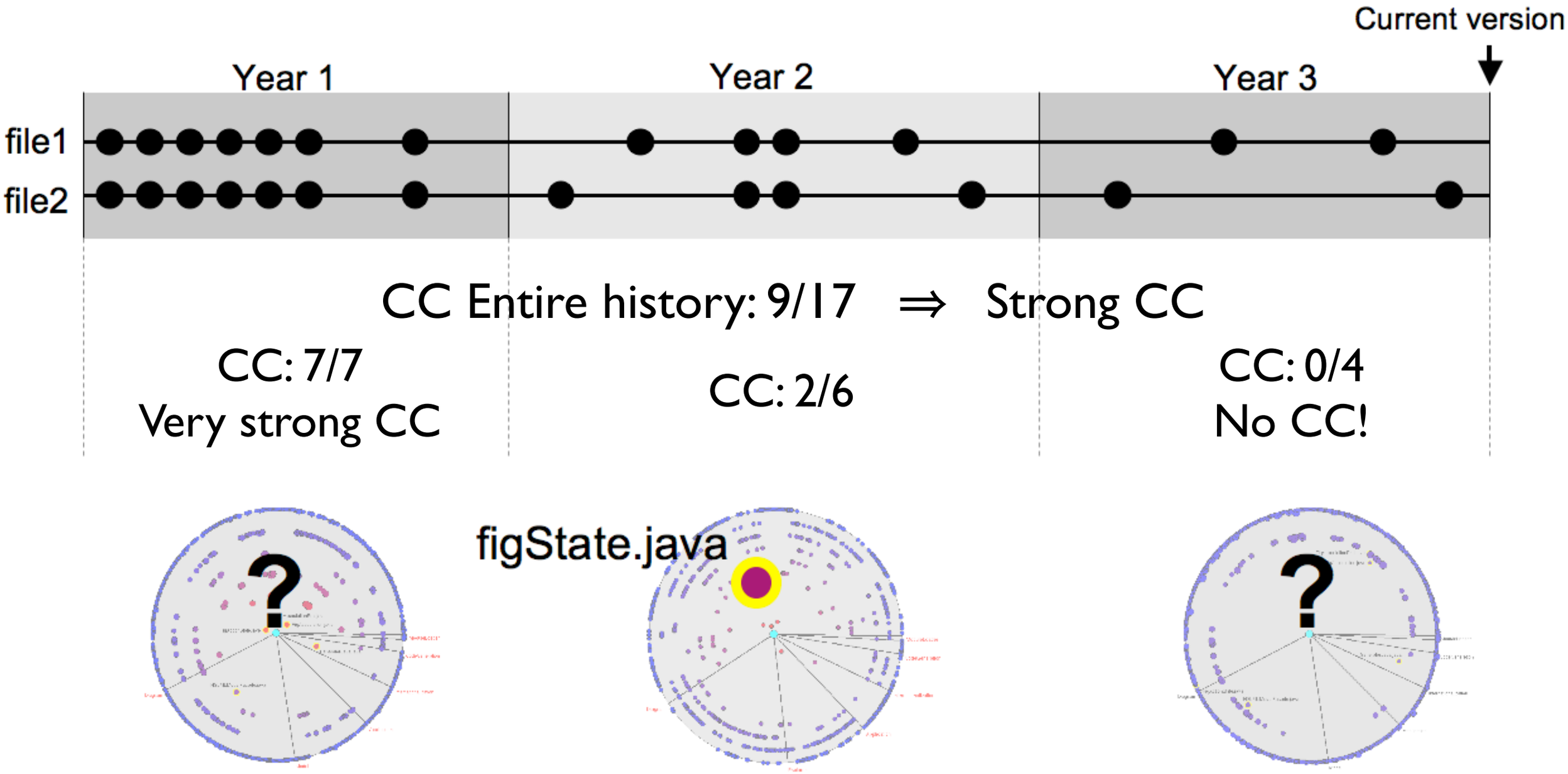






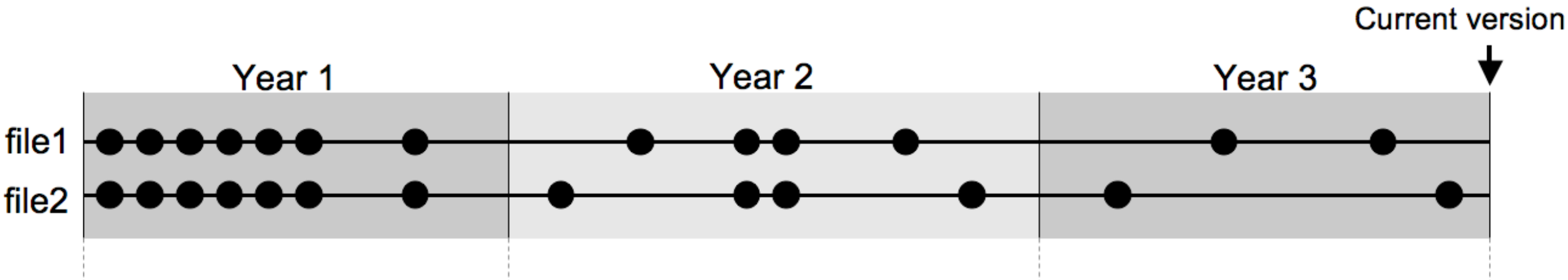
Moving Through Time & Tracking

Problem: The coupling value depends on the time window considered



Moving Through Time & Tracking

Problem: The coupling value depends on the time window considered

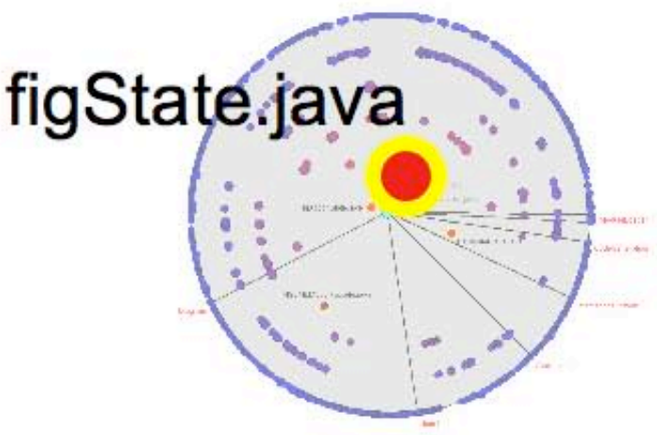


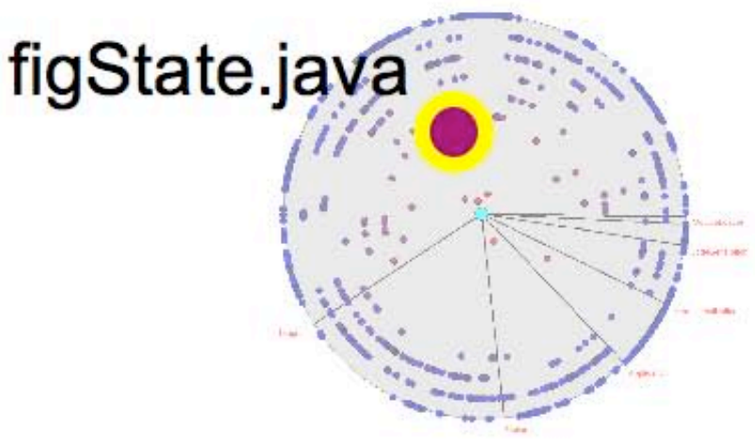
CC Entire history: 9/17 ⇒ Strong CC

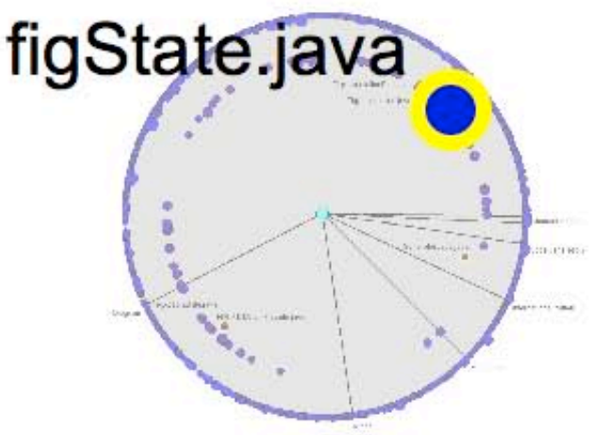
CC: 7/7
Very strong CC

CC: 2/6

CC: 0/4
No CC!





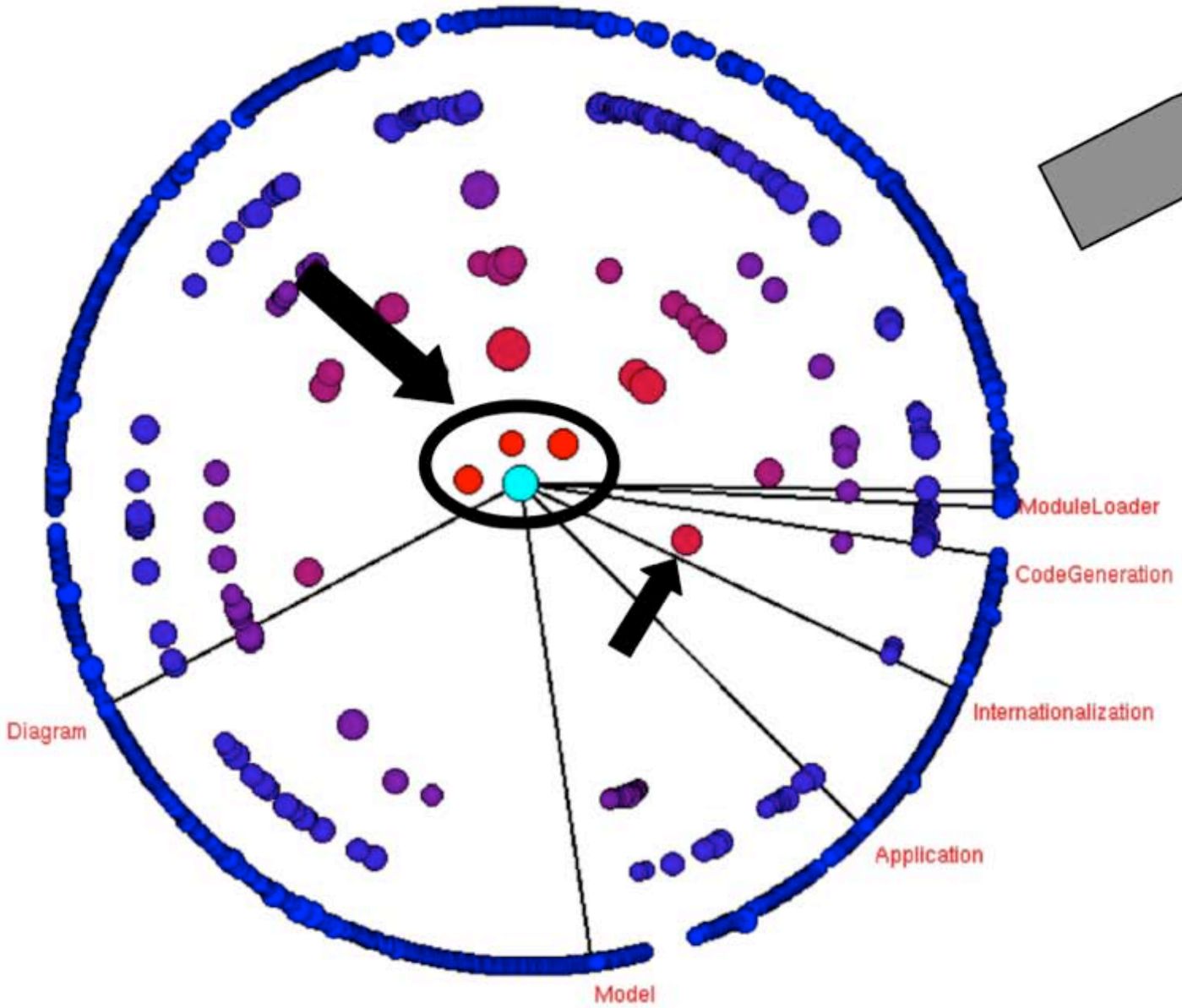


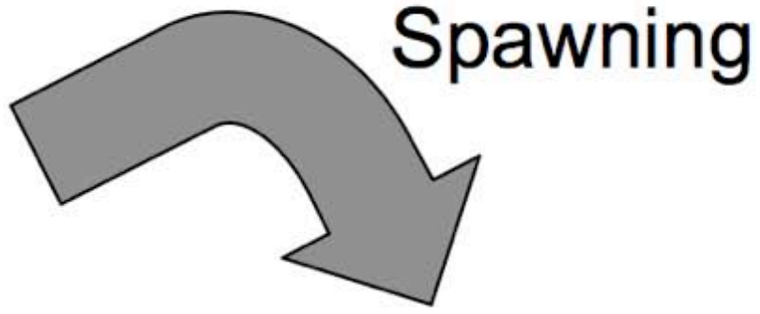
Validation: ArgoUML

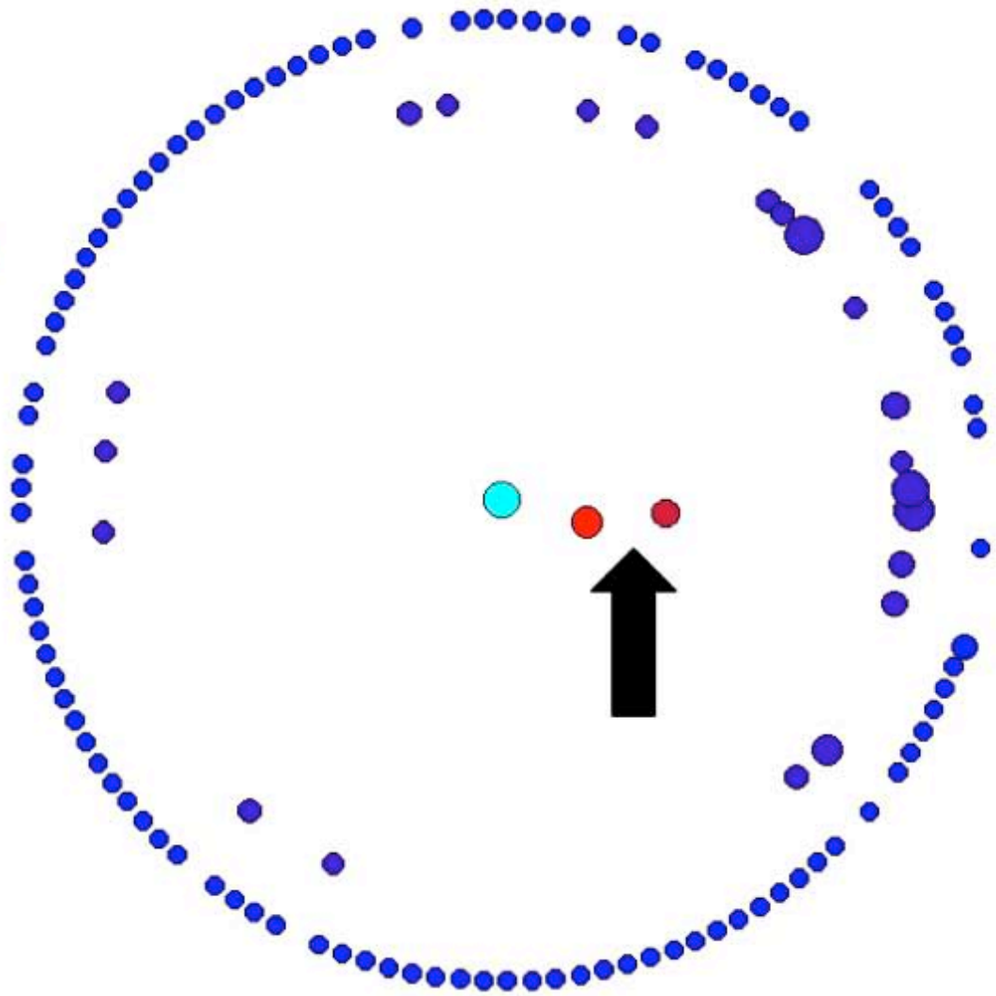
LOC	#Classes	#Commits	Time Interval
~ 220K	~ 2500	~ 50'000	2000 - 2005

- Methodology
  - Consider time interval of 6 months
  - Apply one radar per time interval

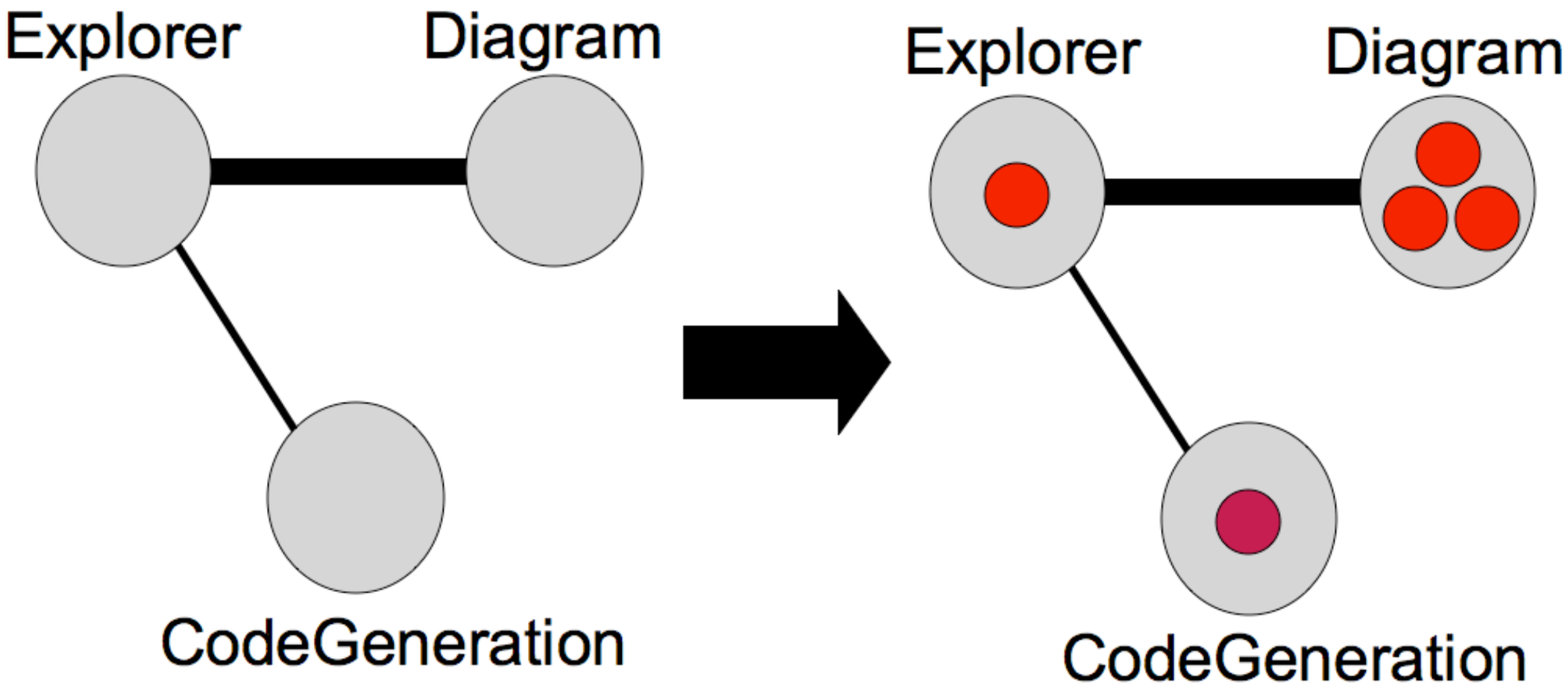
Explorer: August - December '05





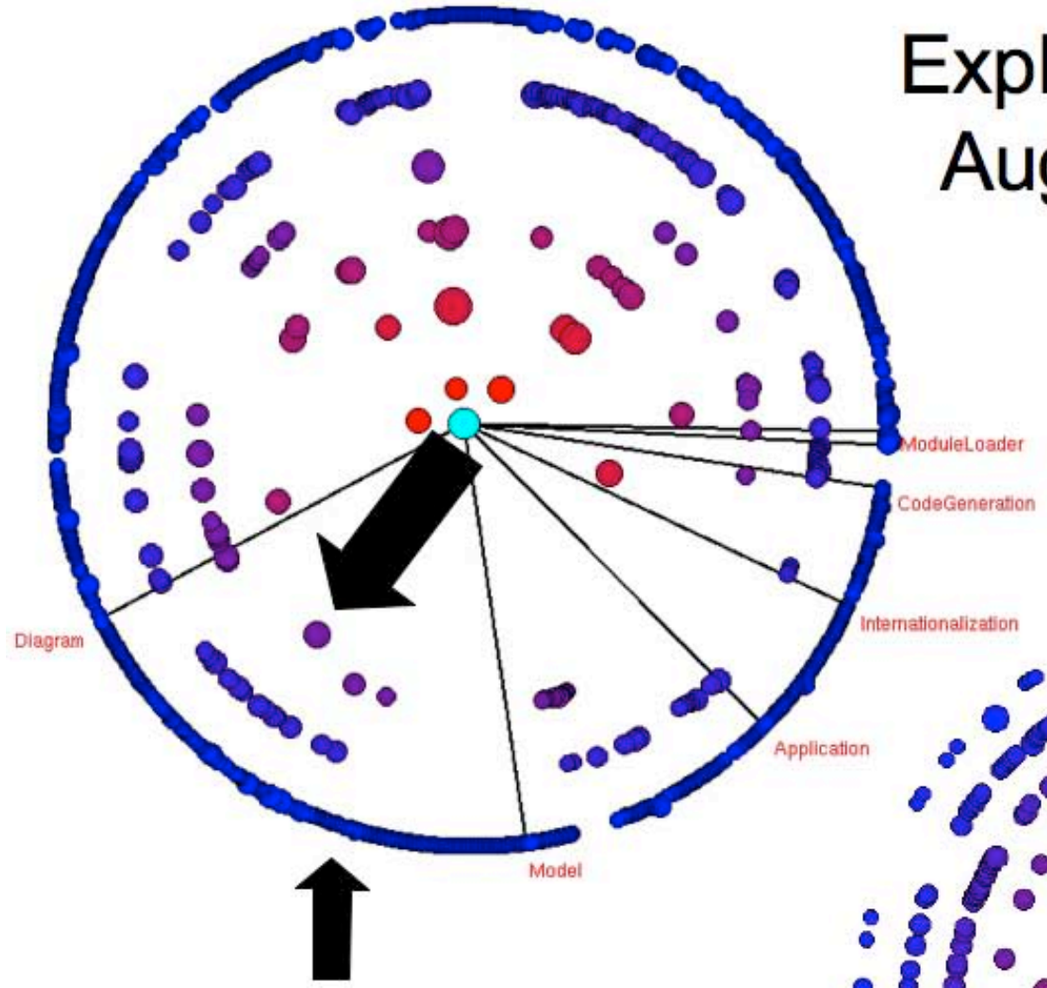


Information Crystallization



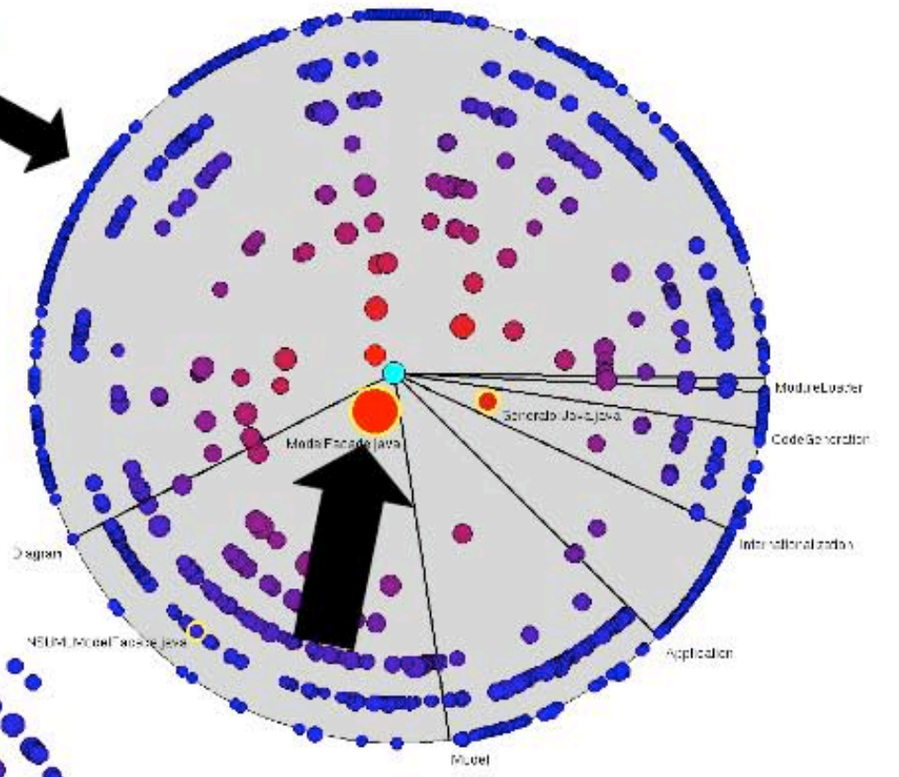
- Dependencies between modules are simplified to dependencies between small sets of classes.
- These classes are candidates for reverse engineering

The Evolution of Model Facade



Explorer in focus
Aug-Dec 2005

Explorer in focus
Aug-Dec 2004



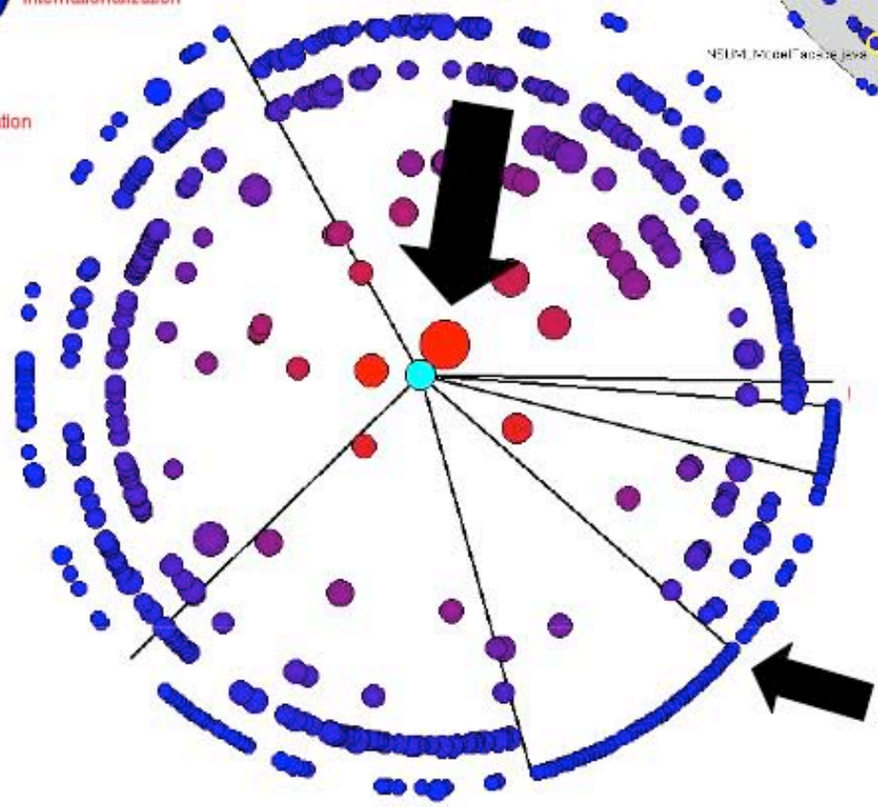


Diagram in focus
Aug-Dec 2004

Evolution Radar Discussion

The Evolution Radar visualizes **integrated** change coupling information. It shows:

- Dependencies at the module level
- The structure of these dependencies in terms of classes, by rendering the classes themselves

Evolution Radar Discussion

The Evolution Radar visualizes **integrated** change coupling information. It shows:

- Dependencies at the module level
- The structure of these dependencies in terms of classes, by rendering the classes themselves

Pros	Cons
• Interactivity and control of time • Scalability • Does not suffer from overplotting • General technique applicable to any groups of entities given a distance measure	• Need an authority system decomposition • The visualization has to be learnt • May suffer from the outliers problem

Epilogue: From study to control Software Evolution

Change coupling (as well as bug and change history) is helpful to detect architecture shortcoming but is **only one aspect** of the evolution of a system

Epilogue: From study to control Software Evolution

Change coupling (as well as bug and change history) is helpful to detect architecture shortcoming but is **only one aspect** of the evolution of a system

*The goal is to aggregate all the pieces of information to tell the story of the system and then to monitor and **control** its evolution*

References

Detection of Logical Coupling Based on Product Release History

Harald Gall, Karin Hajek, Mehdi Jazayeri. In Proceedings of ICSM 1998 (International Conference on Software Maintenance), pp. 190 - 198, IEEE CS Press, 1998.

Mining version histories to guide software changes

Thomas Zimmermann, Peter Weisgerber, Stephan Diehl, Andreas Zeller. In Proceedings of ICSE 2004 (26th International Conference on Software Engineering), pp. 563 - 572, IEEE Computer Society, 2004.

Visualizing Multiple Evolution Metrics

Martin Pinzger, Harald Gall, Michael Fischer, Michele Lanza. In Proceedings of SoftVis 2005 (2nd International ACM Symposium on Software Visualization), pp. 67 - 75, ACM Press, 2005.

Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships

Marco D'Ambros, Michele Lanza. In Proceedings of CSMR 2006 (10th European Conference on Software Maintenance and Reengineering), pp. 227-236 , IEEE Computer Society, 2006.

Reverse Engineering with Logical Coupling

Marco D'Ambros, Michele Lanza. In Proceedings of WCRE 2006 (13th Working Conference on Reverse Engineering), pp. 189 - 198, IEEE Computer Society, 2006.

"A Bug's Life" - Visualizing a Bug Database

Marco D'Ambros, Michele Lanza, Martin Pinzger. In Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis), pp. 113 - 120, IEEE CS Press, 2007.

References

Detection of Logical Coupling Based on Product Release History

Harald Gall, Karin Hajek, Mehdi Jazayeri. In Proceedings of ICSM 1998 (International Conference on Software Maintenance), pp. 190 - 198, IEEE CS Press, 1998.

Mining version histories to guide software changes

Thomas Zimmermann, Peter Weisgerber, Stephan Diehl, Andreas Zeller. In Proceedings of ICSE 2004 (26th International Conference on Software Engineering), pp. 563 - 572, IEEE Computer Society, 2004.

Visualizing Multiple Evolution Metrics

Martin Pinzger, Harald Gall, Michael Fischer, Michele Lanza. In Proceedings of SoftVis 2005 (2nd International ACM Symposium on Software Visualization), pp. 67 - 75, ACM Press, 2005.

Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships

Marco D'Ambros, Michele Lanza. In Proceedings of CSMR 2006 (10th European Conference on Software Maintenance and Reengineering), pp. 227-236 , IEEE Computer Society, 2006.

Reverse Engineering with Logical Coupling

Marco D'Ambros, Michele Lanza. In Proceedings of WCRE 2006 (13th Working Conference on Reverse Engineering), pp. 189 - 198, IEEE Computer Society, 2006.

"A Bug's Life" - Visualizing a Bug Database

Marco D'Ambros, Michele Lanza, Martin Pinzger. In Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis), pp. 113 - 120, IEEE CS Press, 2007.

Analyzing Software Repositories to Understand Software Evolution

Marco D'Ambros, Harald Gall, Michele Lanza, Martin Pinzger
In "**Software Evolution**", Tom Mens, Serge Demeyer, eds. Springer 2008.