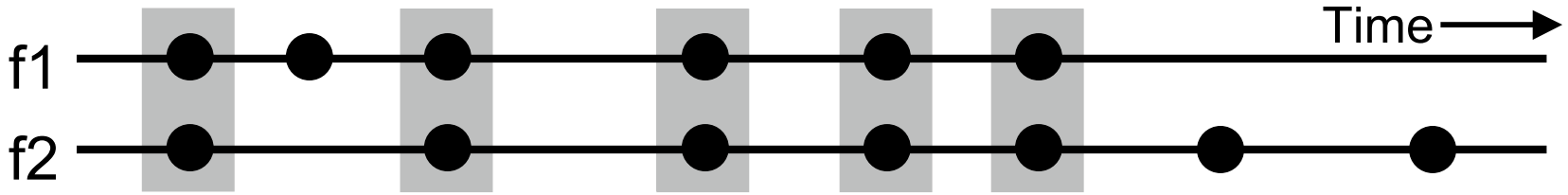
The Evolution Radar: Visualizing Integrated Logical Coupling Information

Marco D'Ambros, Michele Lanza, Mircea Lungu

- Faculty of Informatics -
University of Lugano
Switzerland

MSR 2006 - 3rd International Workshop on Mining Software Repositories, Shanghai, May 22-23, 2006

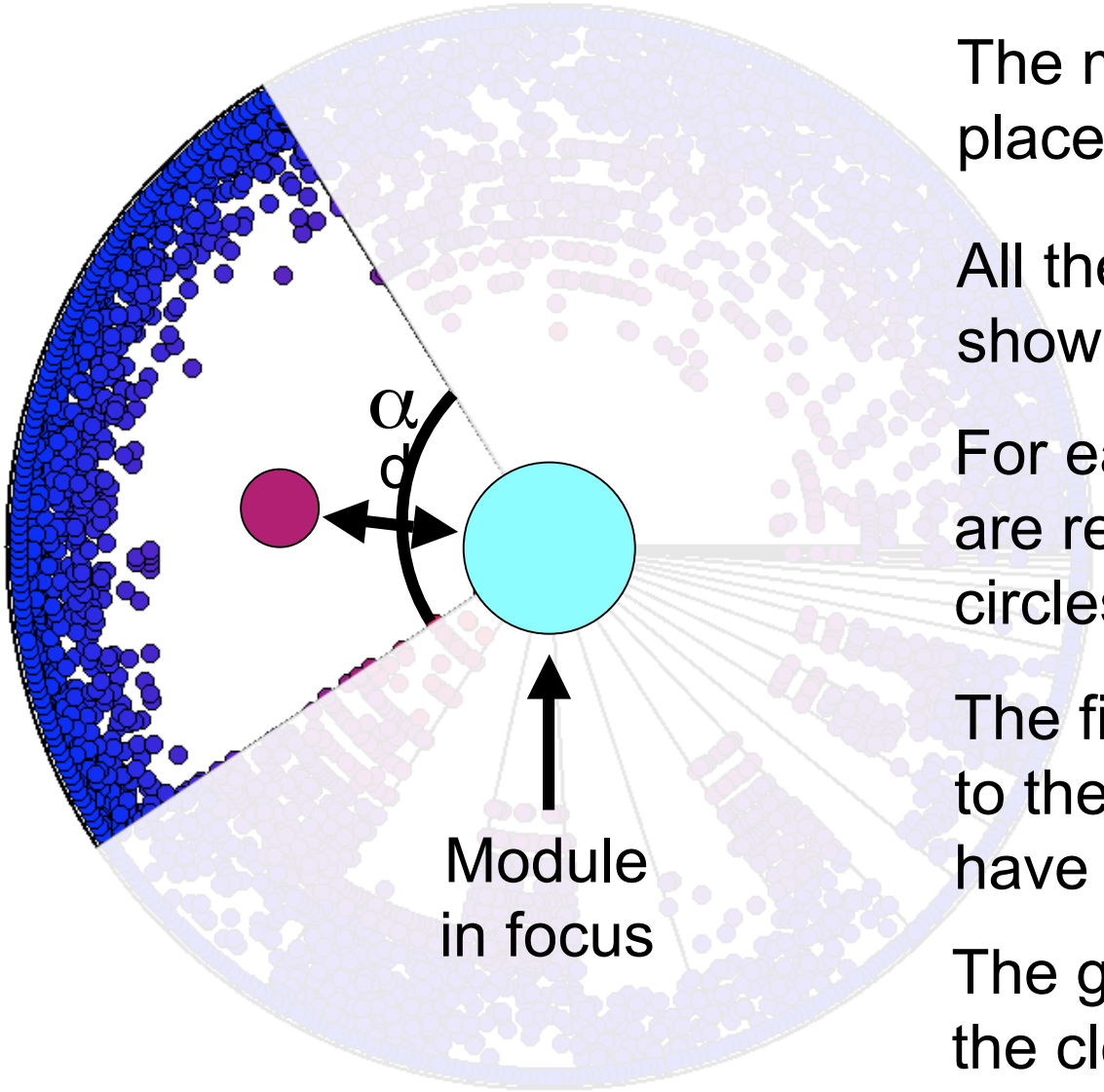
Logical Coupling



- Implicit dependencies between artifacts observed to change together
- Introduced by Gall *et. al.* in [1]
- Benefits
  - Lightweight
  - Visible only in the evolution, not in the code (or documentation)
  - Orthogonal to structural analysis
- Problems
  - Architecture level: Loss of detailed information
  - File (or finer) level: No global view of the system

[1] Detection of logical coupling based on product release history. ICSM '98

The Evolution Radar



The module in focus is placed in the center

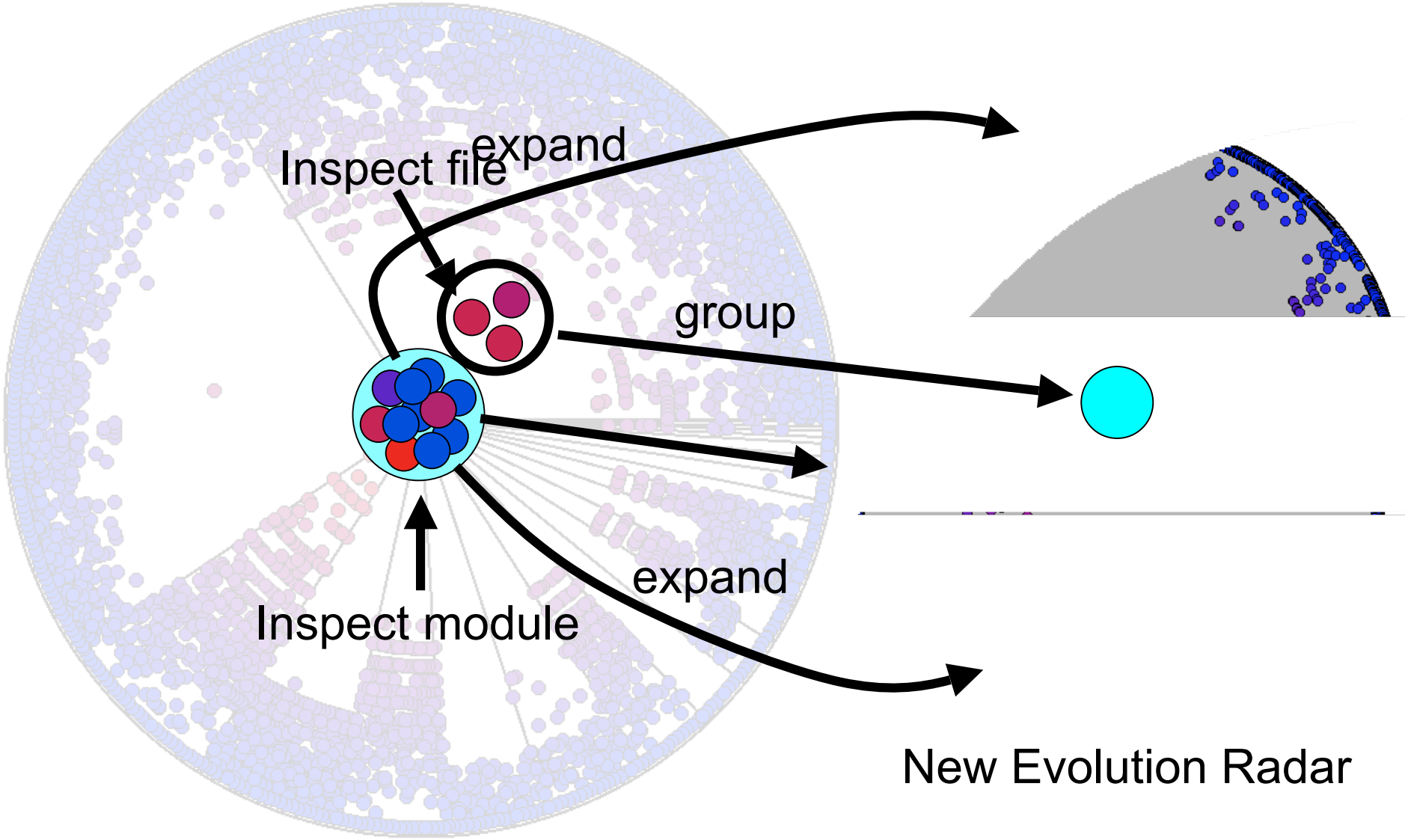
All the other modules are shown as sectors

For each module all its files are rendered as colored circles

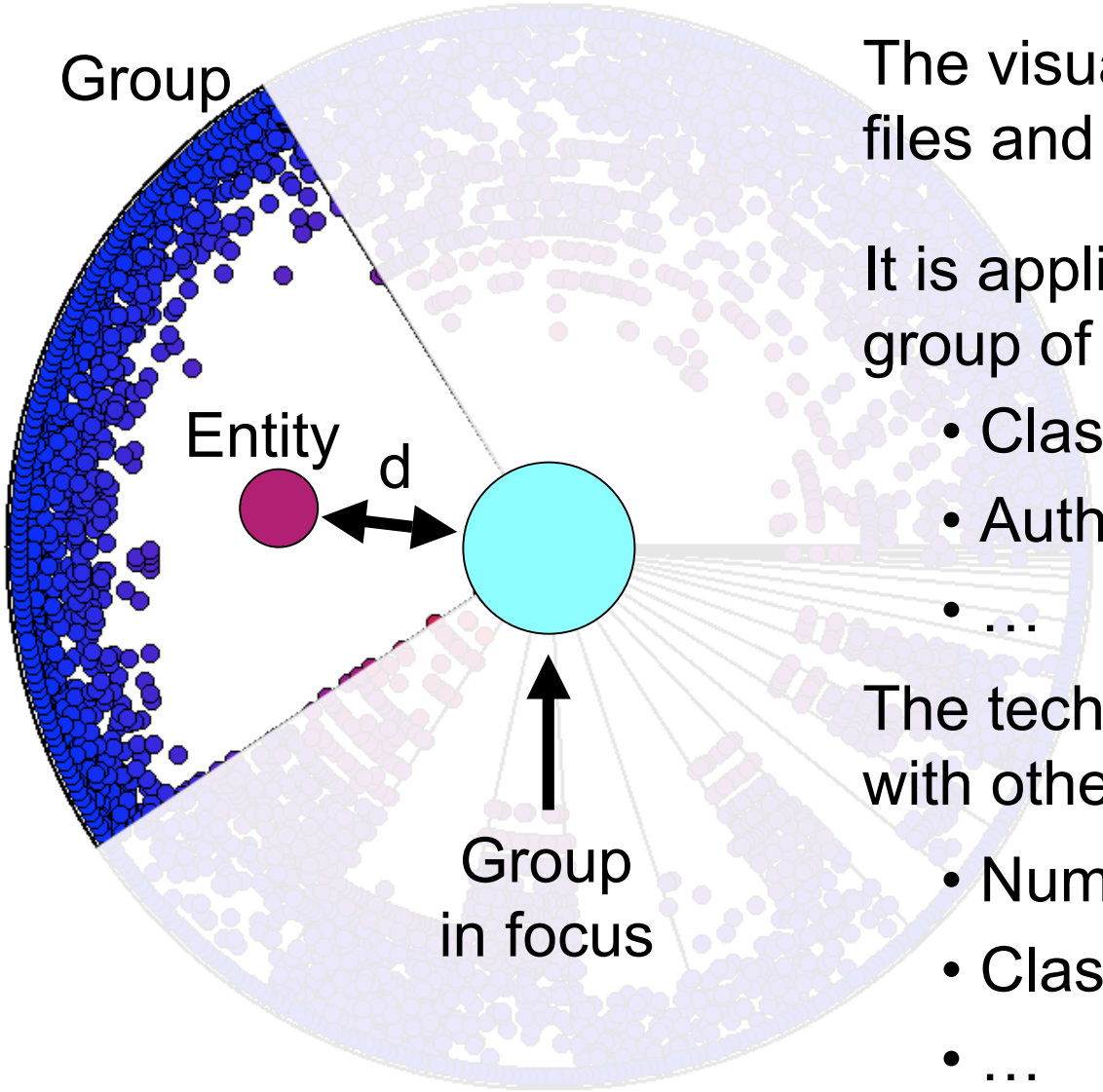
The files are placed according to the logical coupling they have with the module in focus

The greater the logical coupling the closer to the center

Interacting with the Evolution Radar



Generalizing the Evolution Radar



The visualization is not limited to files and modules

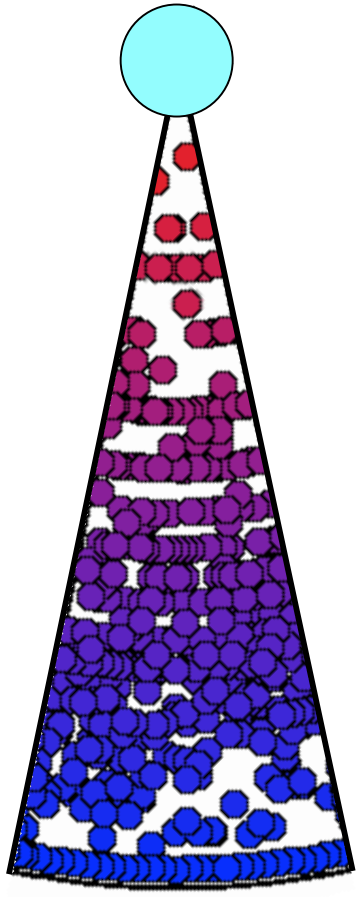
It is applicable to any entity and group of entities:

- Classes and packages
- Authors and teams
- …

The technique can be used with other metrics:

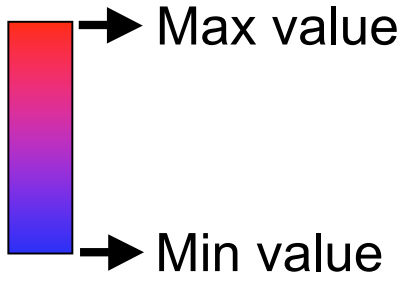
- Number of bugs shared
- Class cohesion
- …

Logical Coupling Metric Mapping



Metrics can be mapped on both position and color

Temperature mapping



Logical coupling metric

- Between files: number of “shared” commits
- Between a file and a module: defined by means of a group operator (max, avg, …)

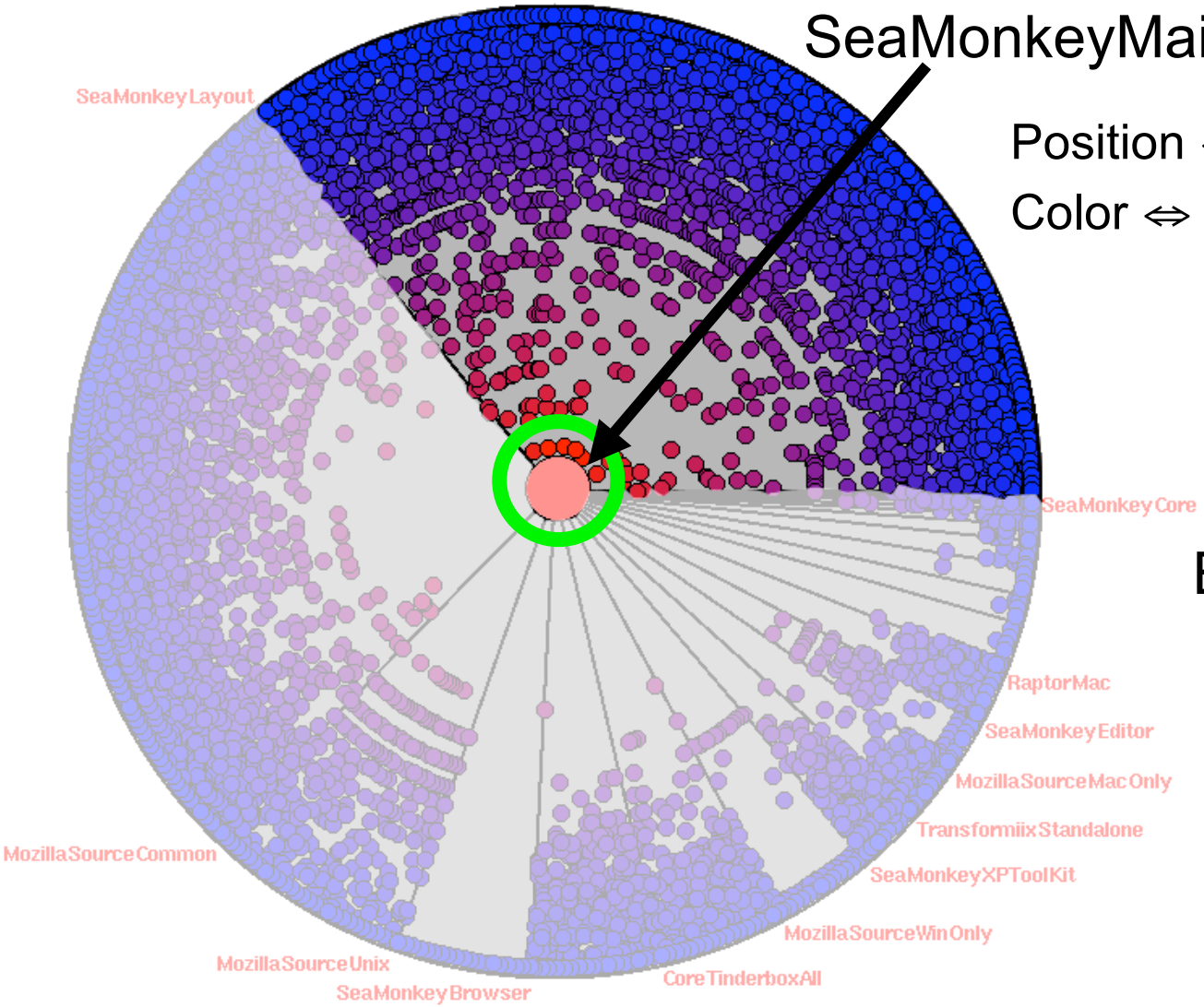
Evolution Radar Applications

Goal	Target audience
Understanding module dependencies and detecting main responsible	Analysts Project managers
Studying change impact	Developers
Understanding the evolution of dependencies	Analysts Project managers

Case Study: Mozilla

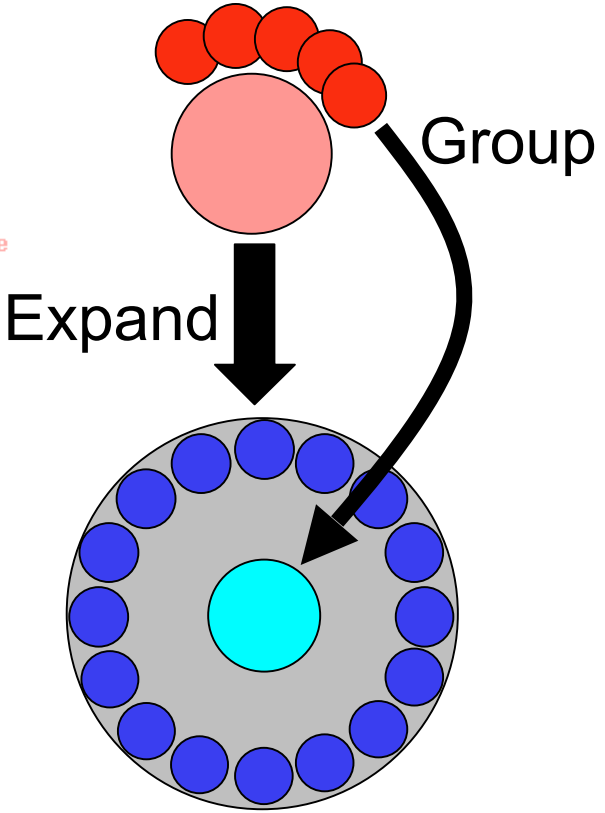
- 30.000 source code files
- 7 years of evolution, 818.000 commits

#1: Understanding module dependencies

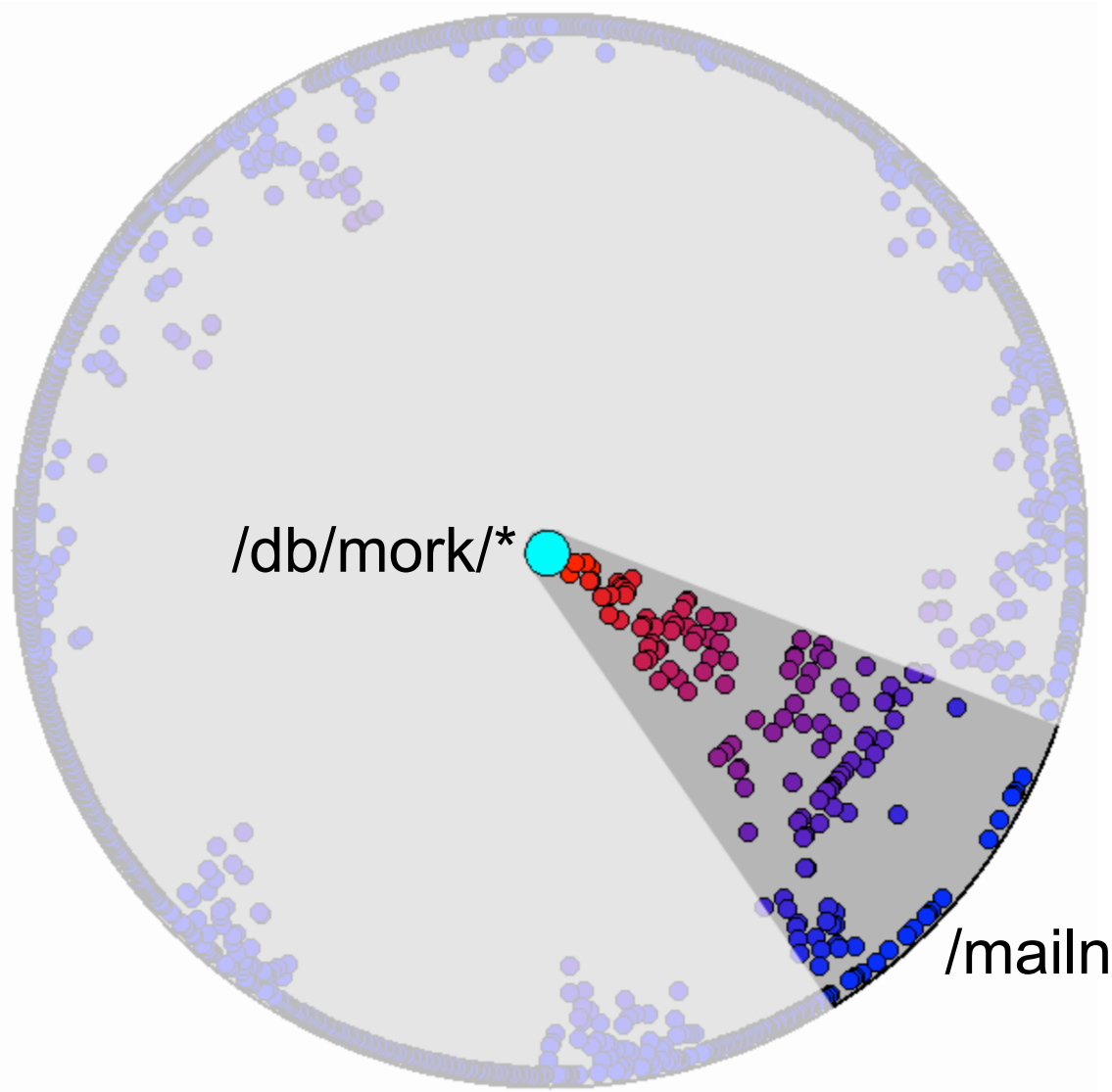


Position ⇔ complete history

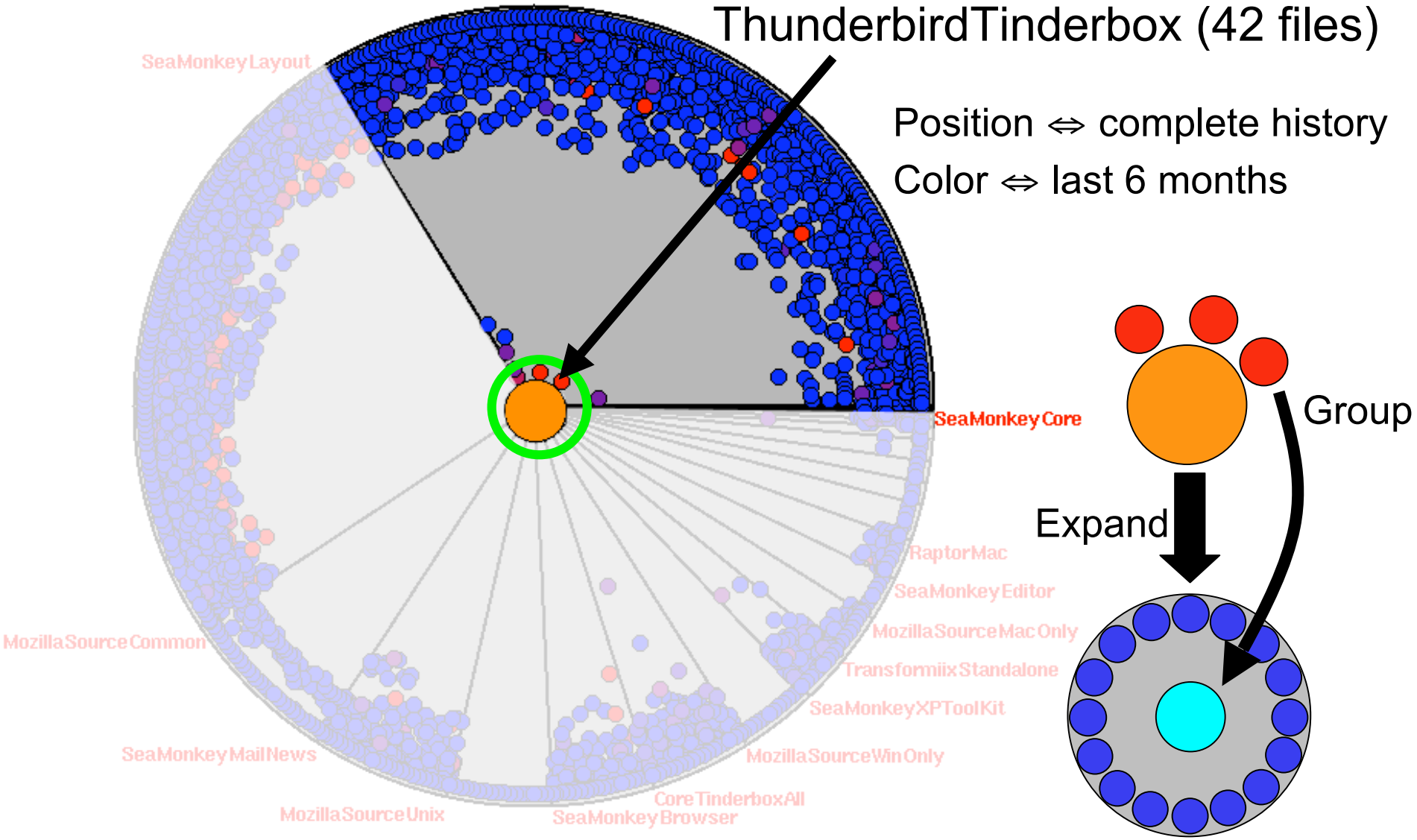
Color ⇔ complete history



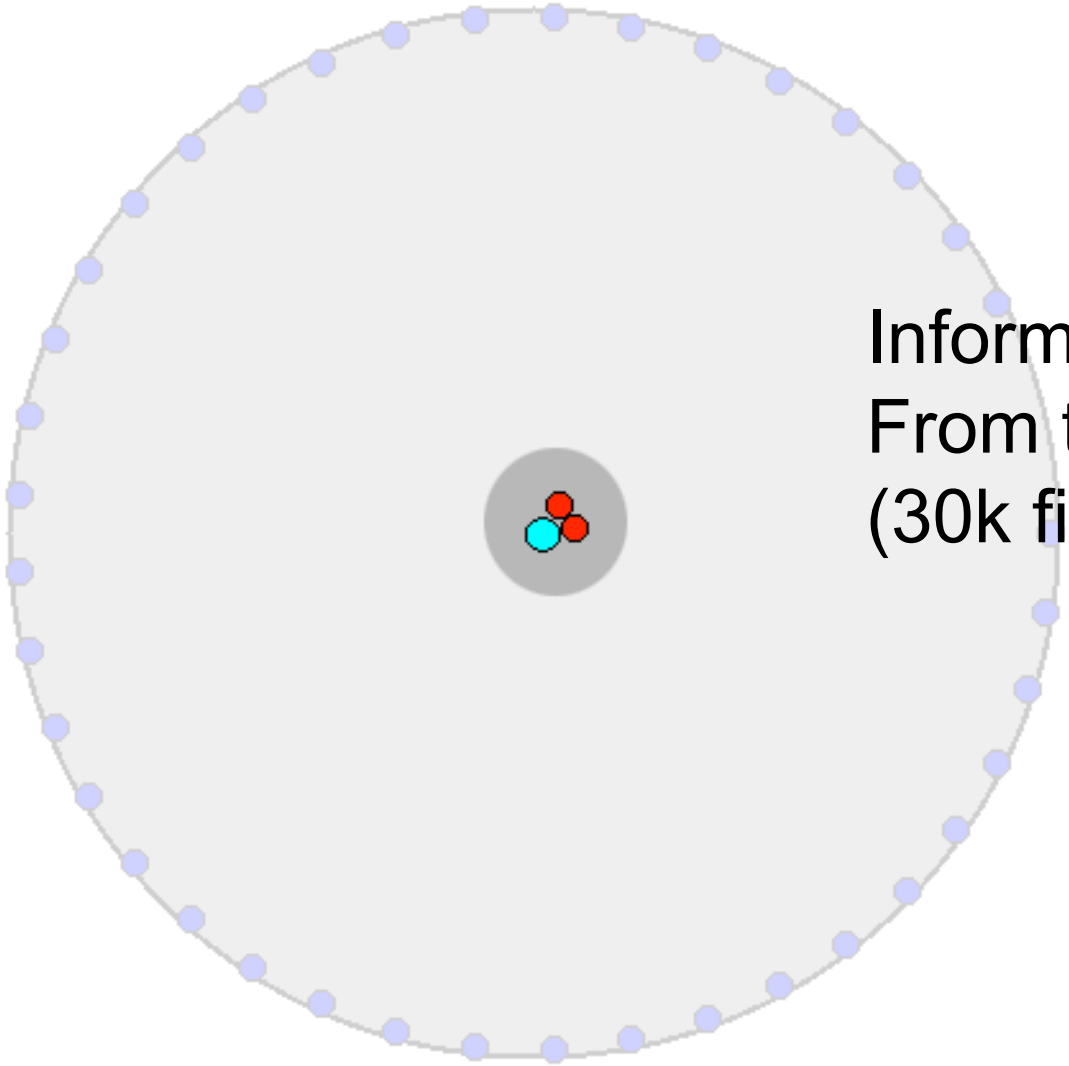
Detecting Responsible for Dependencies



#2: Studying Change Impact



Logical Coupling Details



Information crystallization:
From the whole system (30k files) to 5 files

#3: Studying the Evolution of Dependencies

Module in focus

PhoenixTinderbox (74 files)

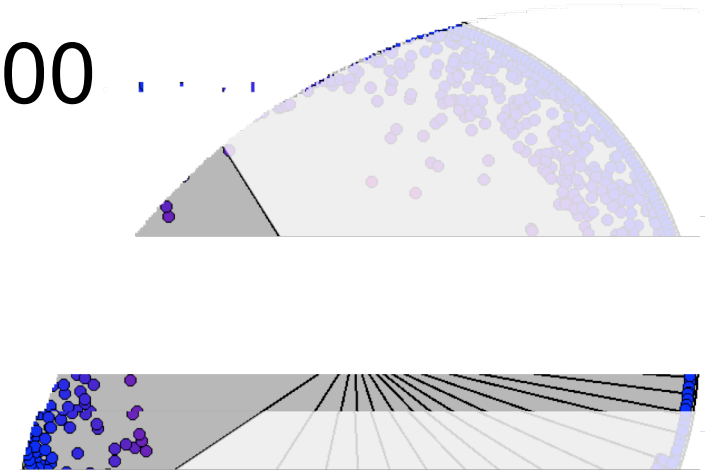
Methodology

Draw an evolution radar for each 2 years of evolution, where the logical coupling mapping is:

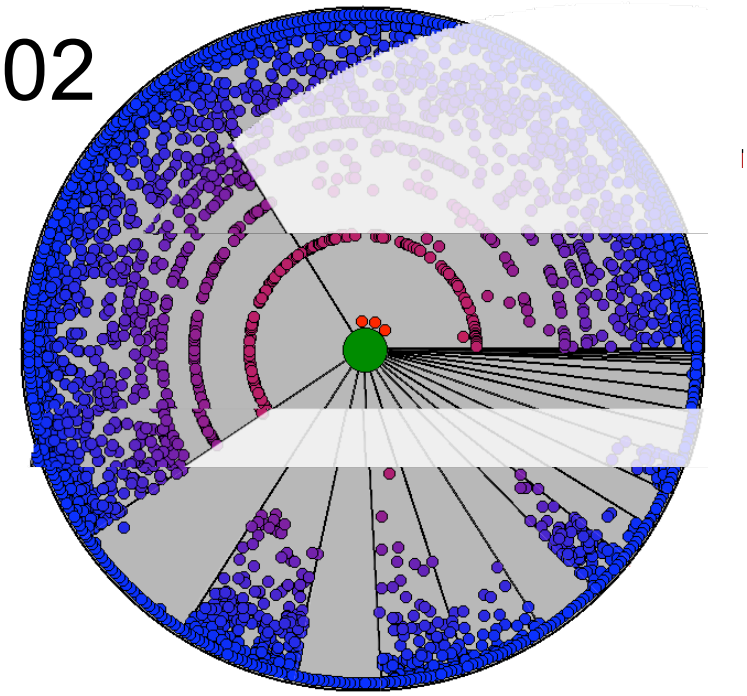
Position ⇔ 2 years

Color ⇔ 2 years

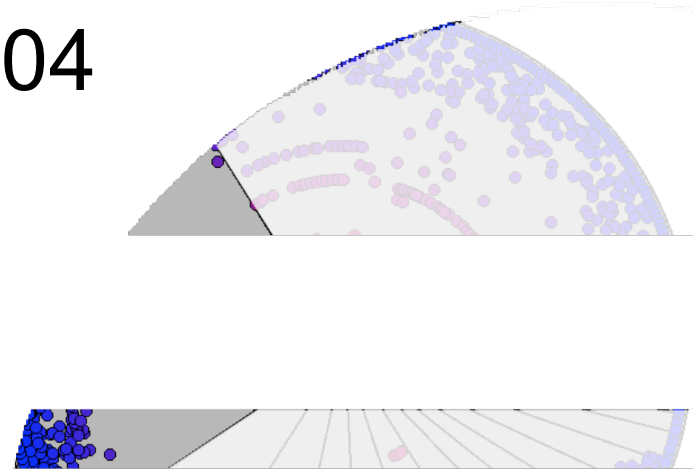
‘98- ‘00



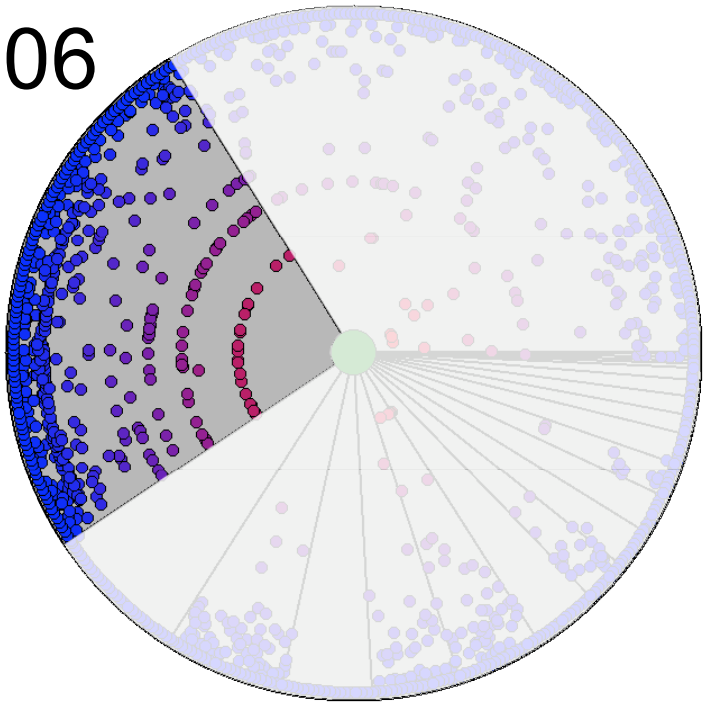
‘00- ‘02



‘02- ‘04



‘04- ‘06



Conclusion

The Evolution Radar visualizes integrated logical coupling information. It shows

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

Pros:

+ Scalable (> 30k files, >800k commits)
+ Interactive
+ Visual expressiveness benefits
  - Rotation invariant
  - Does not suffer from overplotting
  - Occupies a settable amount of space

Cons:

- Need some tuning for the outliers problem
- Fixed time window used to compute the logical coupling

LC Measure Discussion

Outliers (files with LC >> average) can deform the visualization by pushing all the other figures at the boundary

Possible solution	Pro	Cons
Log scale Log(LC)	Simple	Can still suffer from outliers
Percentage LC / noc[1]	No outliers	Files with 2 and 100 commits can have the same value
Percentage and log (LC / noc) * log(noc)	No outliers	
Percentage with query engine to detect files with noc < threshold	No outliers	Manual removal of files

[1] noc = number of commits