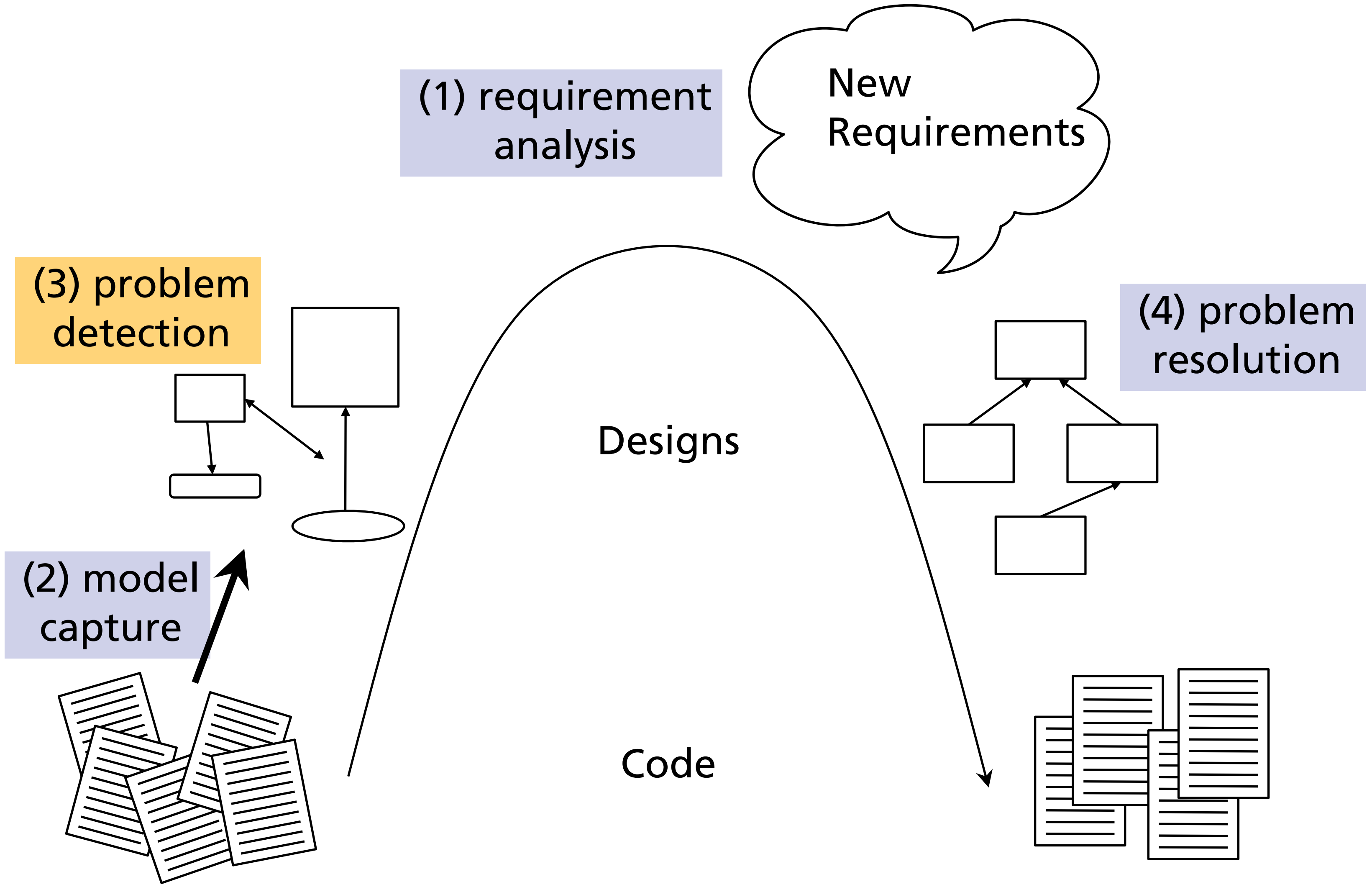
Software Reengineering Problem Detection

Martin Pinzger
Delft University of Technology

Outline

The Reengineering Life-Cycle

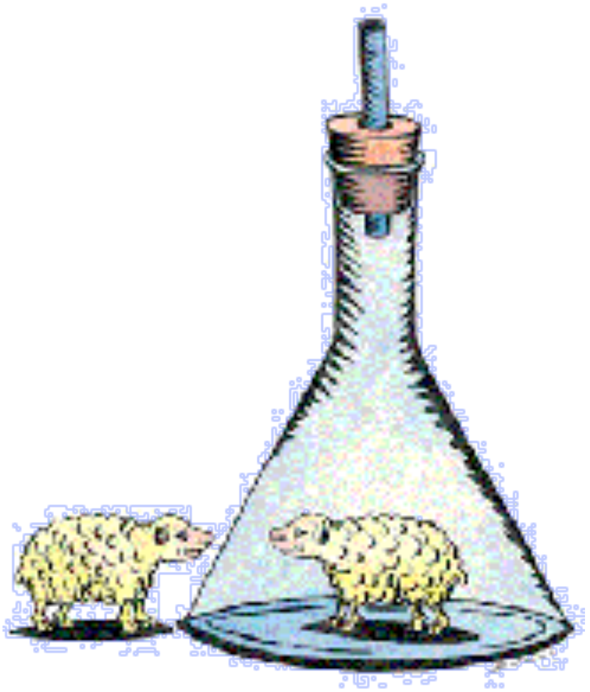


Design Problems

The most common design problems result from code that is

Unclear & complicated

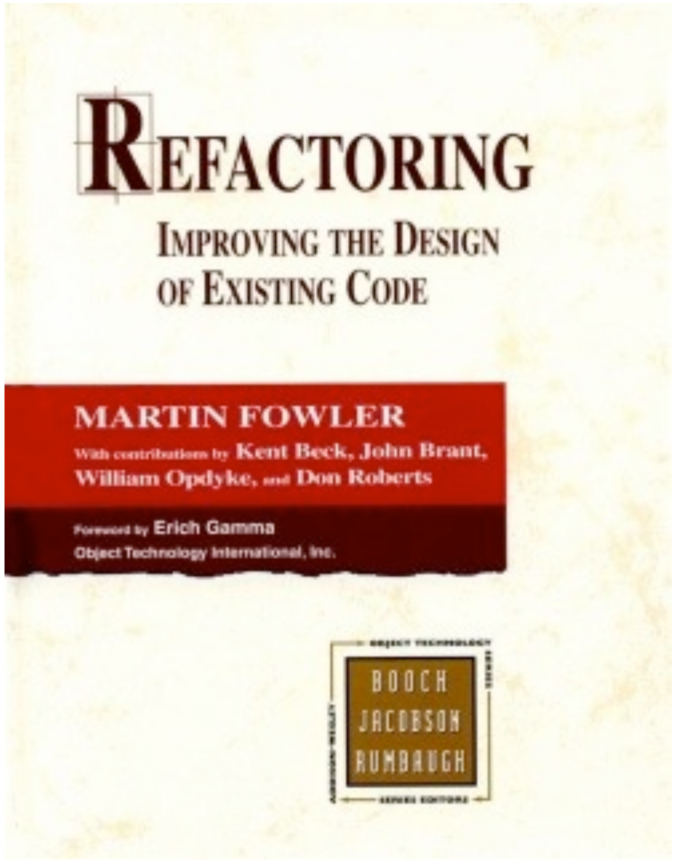
Duplicated (code clones)



Code Smells (if it stinks, change it)

A code smell is a hint that something has gone wrong somewhere in your code.

Duplicated Code
Long Method
Large Class
Long Parameter List
Divergent Change
Shotgun Surgery
Feature Envy
...


REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International, Inc.
OBJECT TECHNOLOGY SERIES
BOOCH JACOBSON RUMBAUGH
SERIES EDITORS

How To Detect?

Measure and visualize quality aspects of the current implementation of a system

- Source code metrics and structures

Measure and visualize quality aspects of the evolution of a system

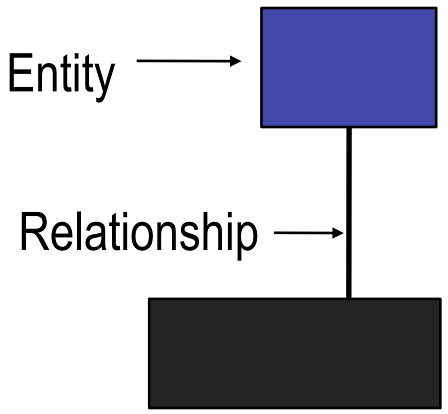
- Evolution metrics and structures

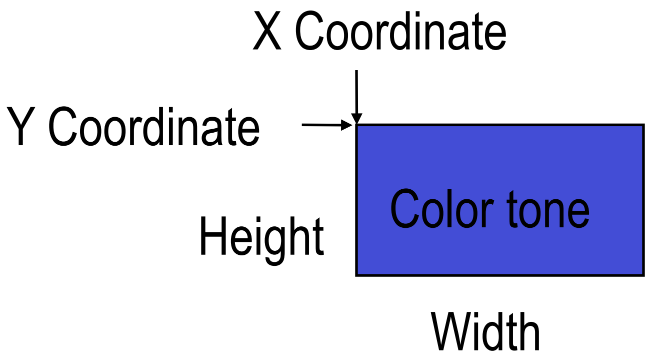
Use Polymetric Views

Polymetric Views

A combination of metrics and software visualization

- Visualize software using colored rectangles for the entities and edges for the relationships
- Render up to five metrics on one node:
  - Size (1+2)
  - Color (3)
  - Position (4+5)





Smell 1: Long Method

The longer a method is, the more difficult it is to understand it.

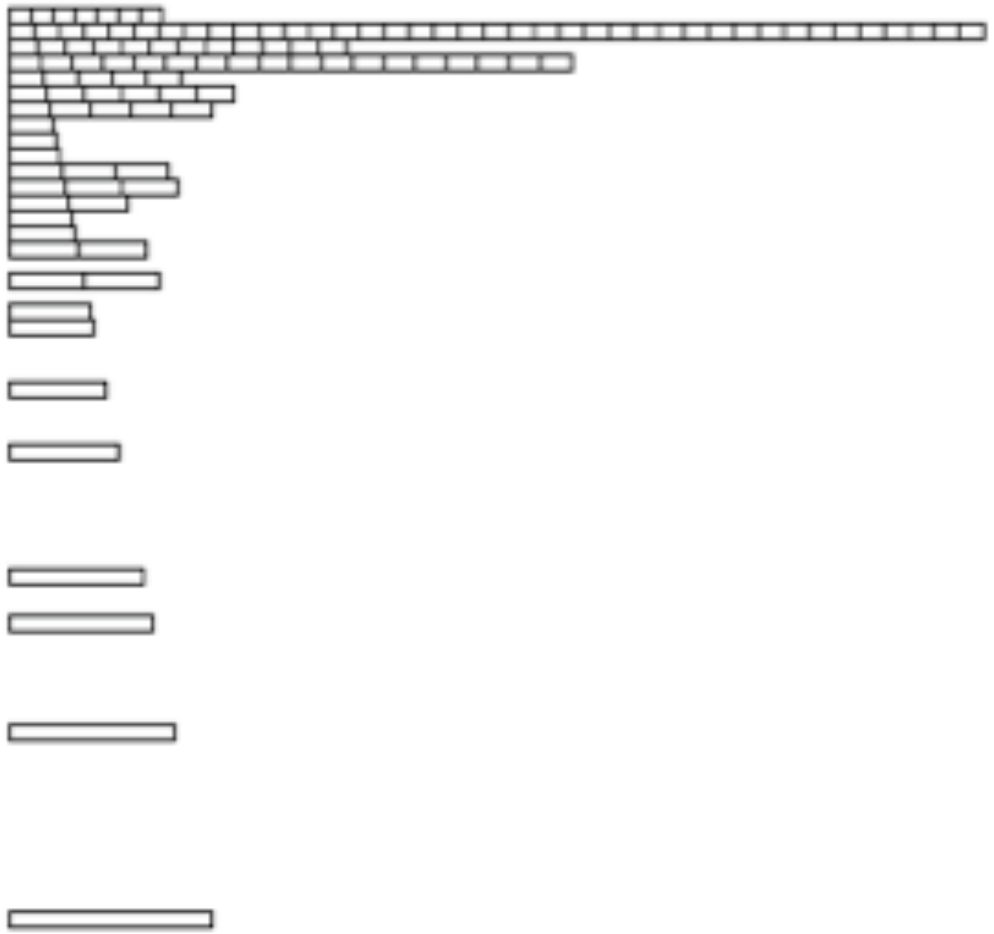
When is a method too long?

- Heuristic: > 10 LOCs (?)

How to detect?

- Visualize LOC metric values of methods
- “Method Length Distribution View”

Method Length Distribution



Metrics:
Boxes: Methods
Width: LOC
Position-Y: LOC
Sort: LOC

Smell 2: Switch Statement

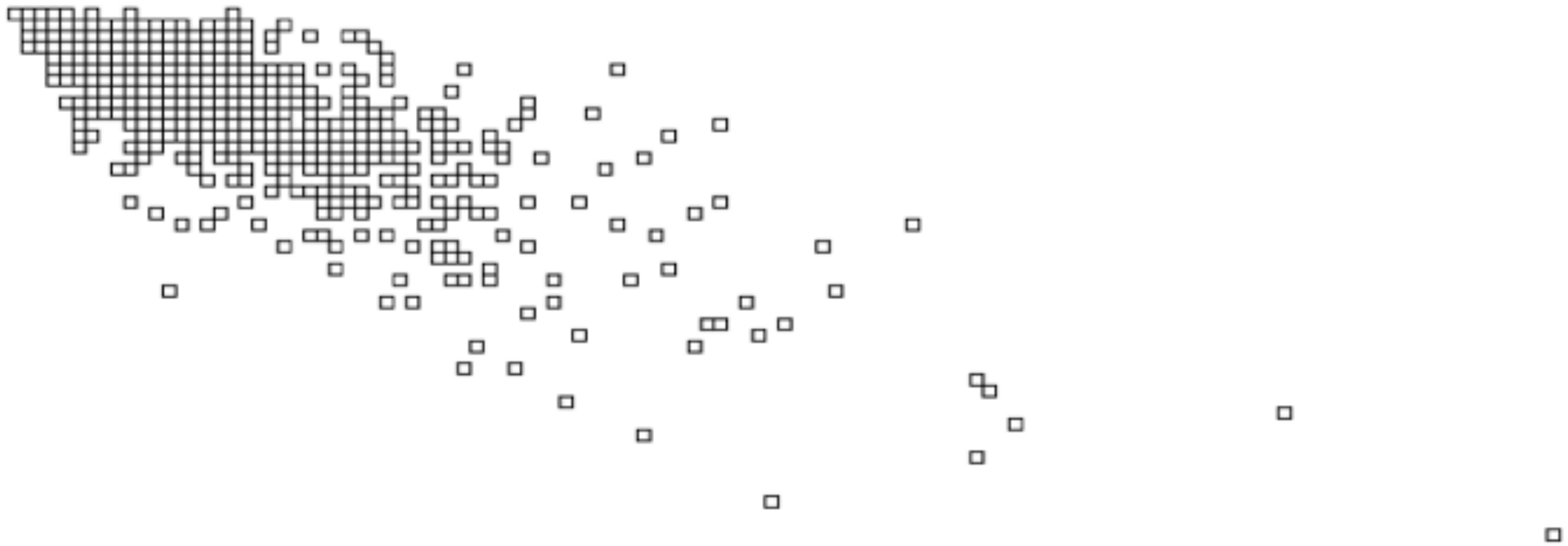
Problem is similar to code duplication

- Switch statement is scattered in different places

How to detect?

- Visualize McCabe Cyclomatic Complexity metric to detect complex methods
- “Method Complexity Distribution View”

Method Complexity



Metrics:
Boxes: Methods
Position-X: LOC
Position-Y: MCC
Sort: -

Smell 3: System Hotspots

Classes that contain too much responsibilities

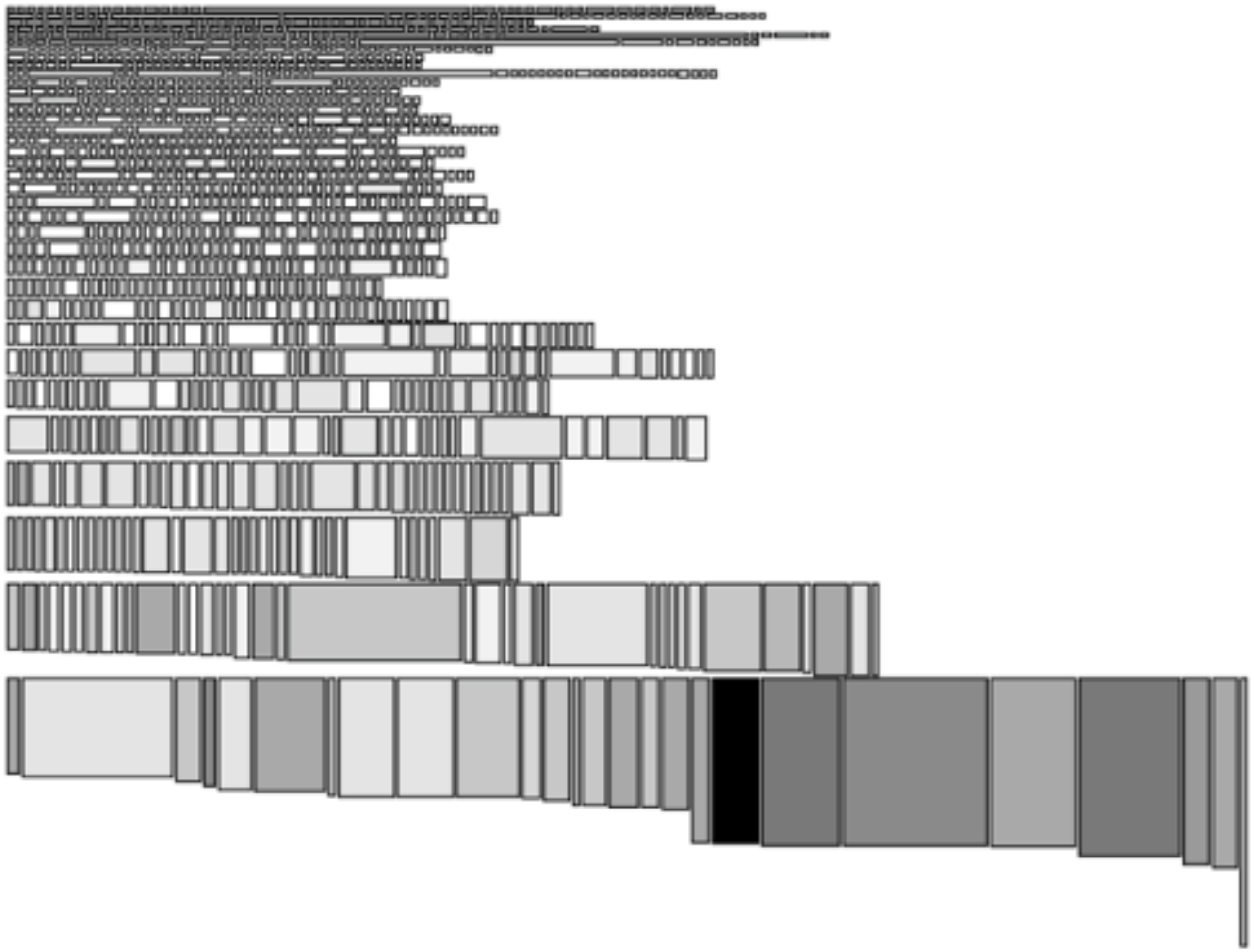
When is a class too large?

- Heuristic: > 20 NOM

How to detect?

- Visualize number of methods (NOM) and sum of lines of code of methods (WLOC)
- “System Hotspots View”

System Hotspots



Metrics:
Boxes: Classes
Width: NOA
Height: NOM
Color: LOC
Sort: NOM

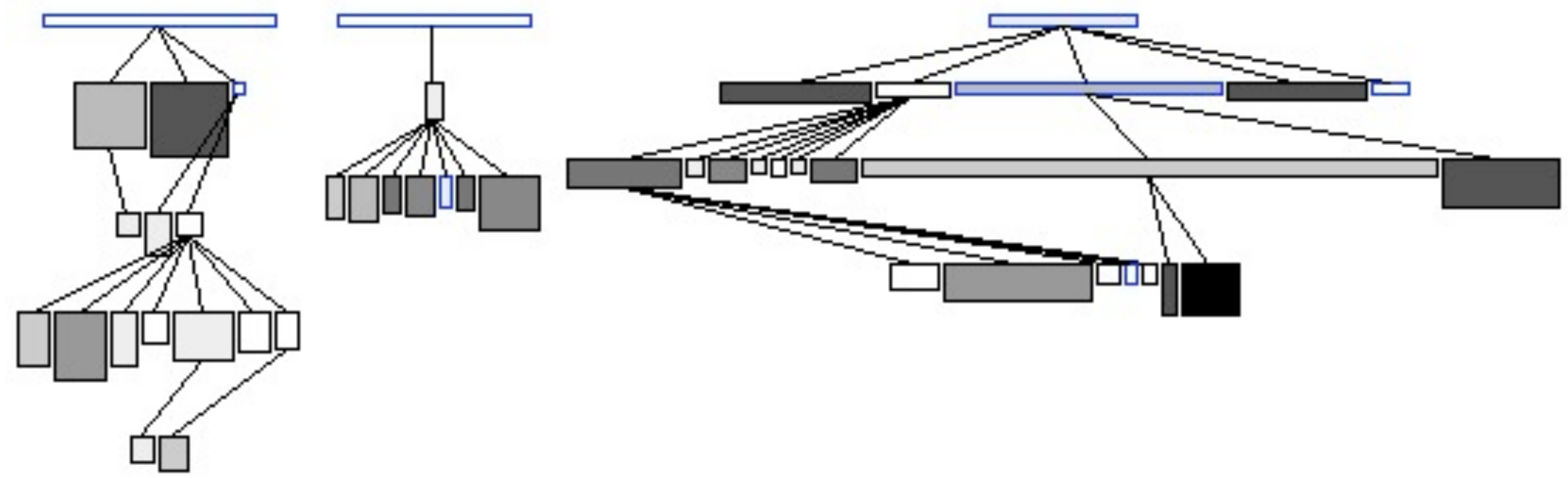
Smell 4: Lazy Sub-Class

A class that is not doing enough to pay for itself should be eliminated

How to detect?

- Visualize inheritance structure with number of methods added (NMA), overridden (NMO), and extended (NME)
- “Inheritance Classification View”

Inheritance Classification



Metrics:
Boxes: Classes
Edges: Inheritance
Width: NMA
Height: NMO
Color: NME
Sort: -

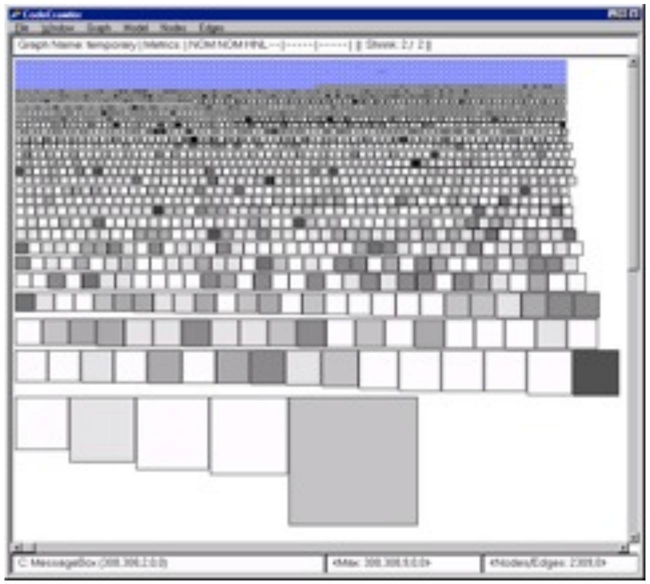
Evaluation: Polymetric Views

Pros

- Quick insights
- Scalable
- Metrics add semantics
- Interactivity makes the code “come nearer”
- Reproducible
- Industrial Validation is the acid test

Cons

- Level of granularity
- Code reading is needed



RoadMap

Introduction

Problem detection in the source code

- Code Smells
- Polymetric Views

Problem detection in the evolution

- The Evolution Matrix
- Kiviat Graphs

Conclusion

Understanding Evolution

Changes can point to design problems

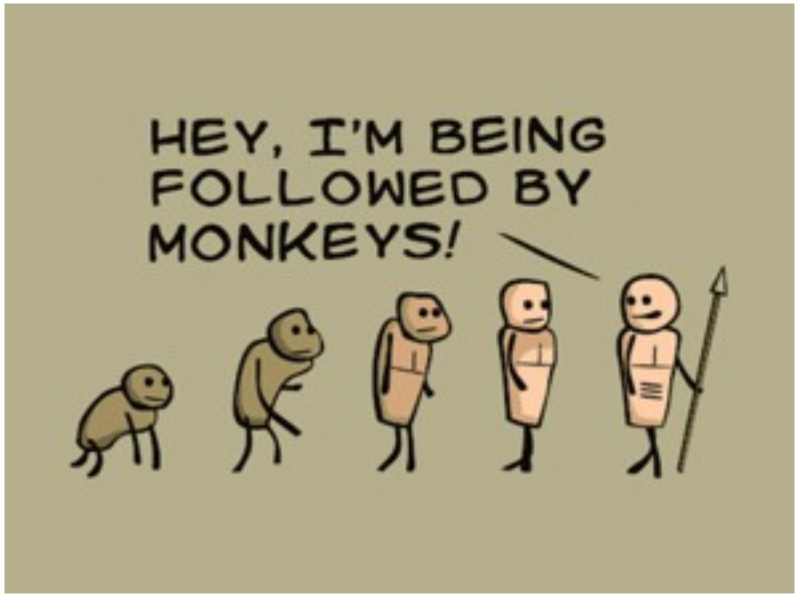
- "Evolutionary Smells"

But

- Overwhelming complexity
- How can we detect and understand changes?

Solutions

- The Evolution Matrix
- The Kiviat Graphs

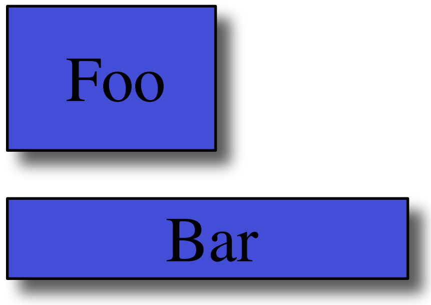


Visualizing Class Evolution

Visualize classes as rectangles using for width and height the following metrics:

NOM (number of methods)

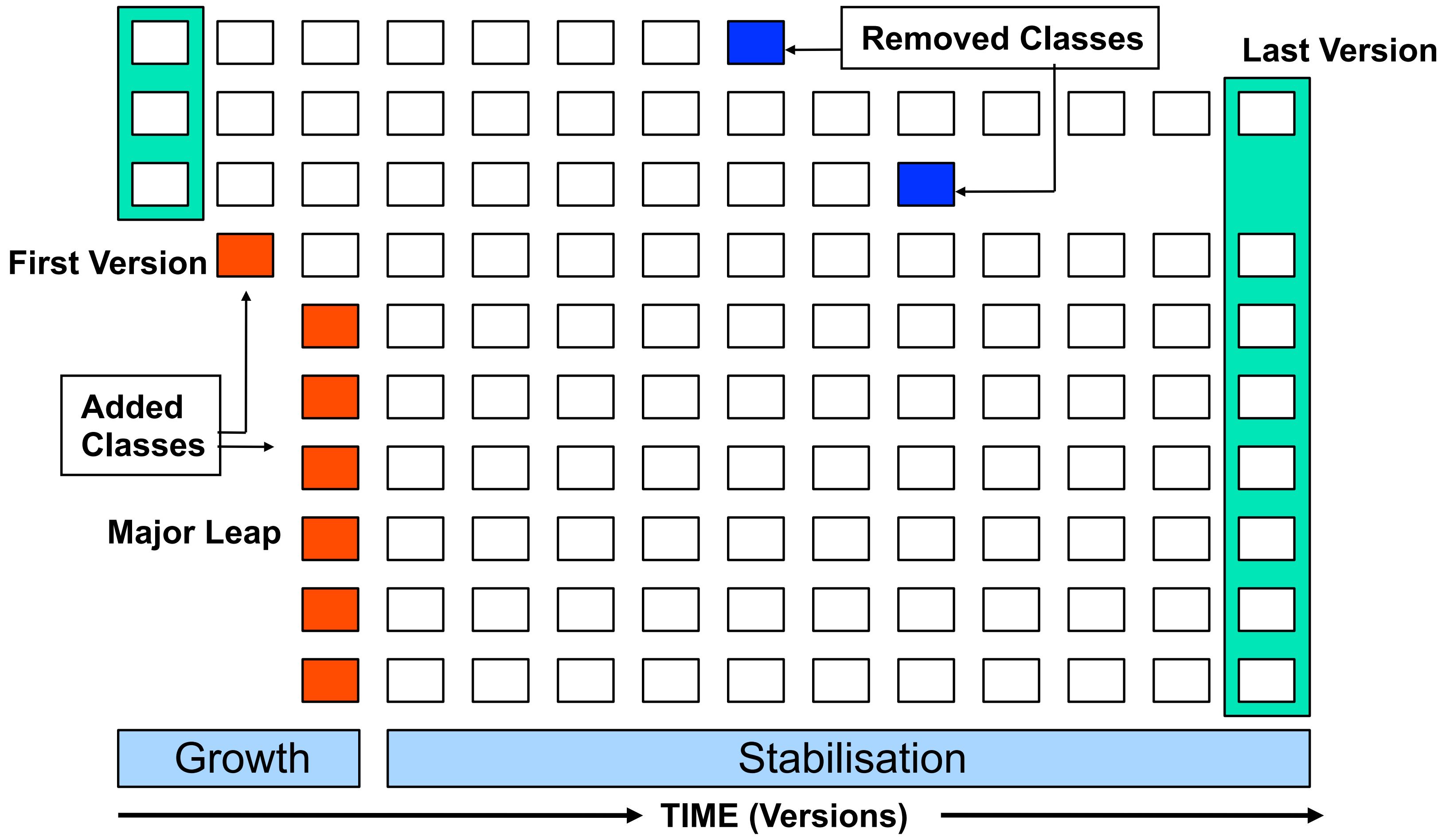
NOA (number of attributes)



The Classes can be categorized according to their “personal evolution” and to their “system evolution”

-> Evolution Patterns

The Evolution Matrix



Evolution Patterns & Smells

Day-fly (Dead Code)

Persistent

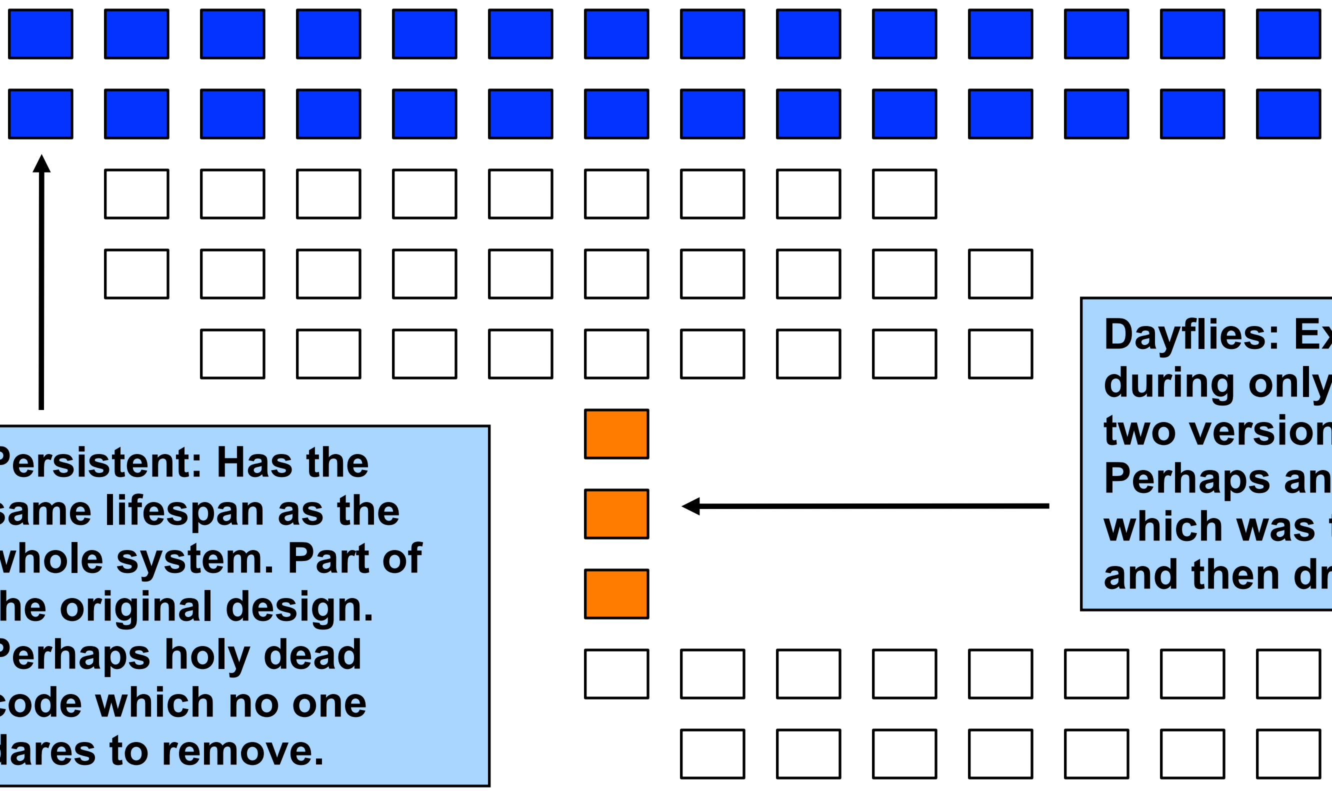
Pulsar (Change Prone Entity)

SupernovaWhite Dwarf (Dead Code)

Red Giant (Large/God Class)

Idle (Dead Code)

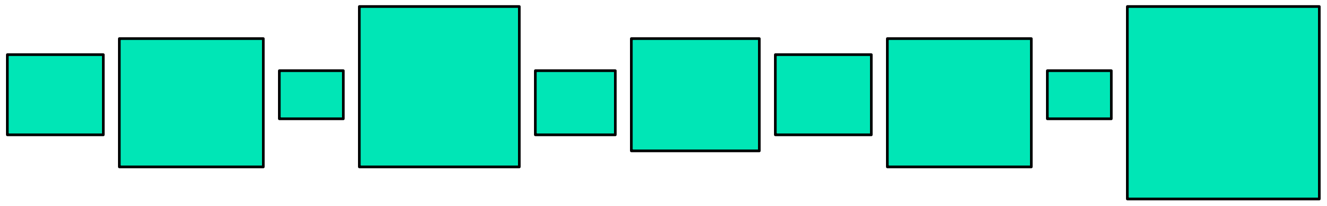
Persistent / Dayfly



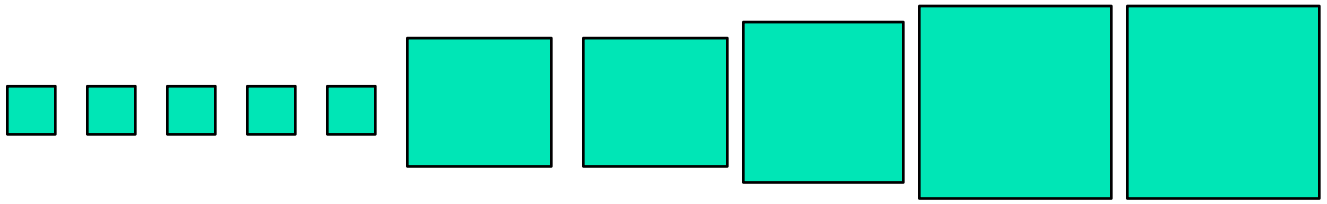
Persistent: Has the same lifespan as the whole system. Part of the original design. Perhaps holy dead code which no one dares to remove.

Dayflies: Exists during only one or two versions. Perhaps an idea which was tried out and then dropped.

Pulsar / Supernova

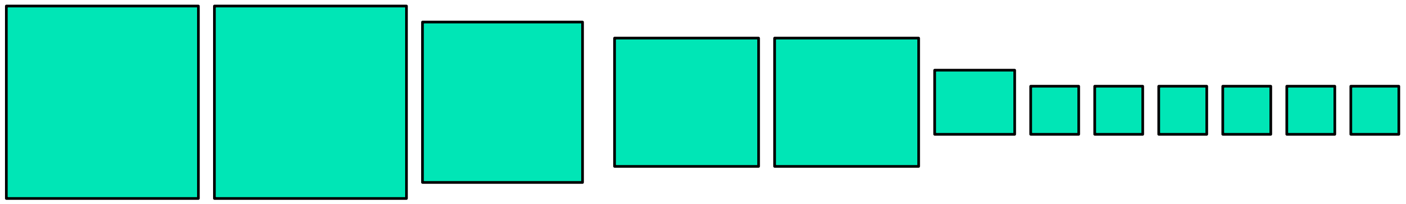


Pulsar: Repeated Modifications make it grow and shrink.
System Hotspot: Every System Version requires changes.

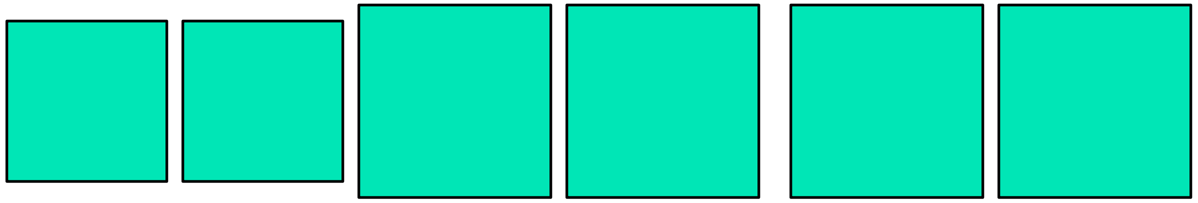


Supernova: Sudden increase in size. Possible Reasons:
- Massive shift of functionality towards a class.
- Data holder class for which it is easy to grow.
- *Sleeper*: Developers knew exactly what to fill in.

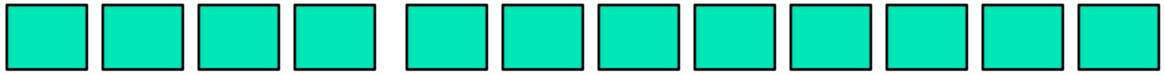
White Dwarf / Red Giant / Idle



White Dwarf: Lost the functionality it had and now trundles along without real meaning. Possibly dead code -> Lazy Class.

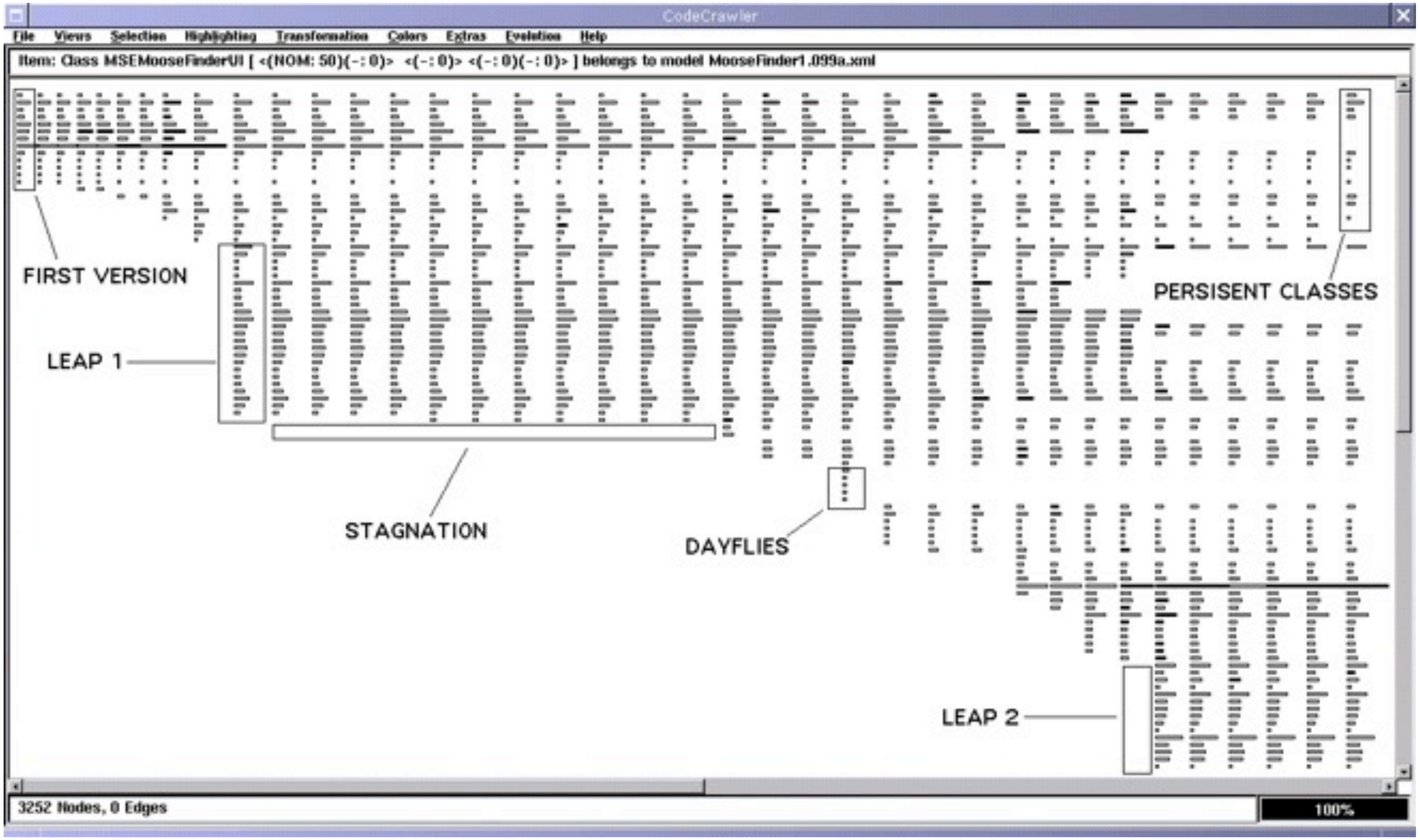


Red Giant: A permanent god (large) class which is always very large.



Idle: Keeps size over several versions. Possibly dead code, possibly good code.

Real Example: MooseFinder



Evaluation: Evolution Matrix

Pros

- Understand the evolution of a system in terms of size and growth rate
- Introduction of new classes
- Remove of classes
- Detection of Evolution Patterns & Smells
  - Dayflight, Persistent, White Dwarf, ...

Cons

- Scalability
- Limited to 3 metric values per glyph
- Fragile regarding the renaming of classes
  - What if the name of a class was changed?

Extended Polymetric Views

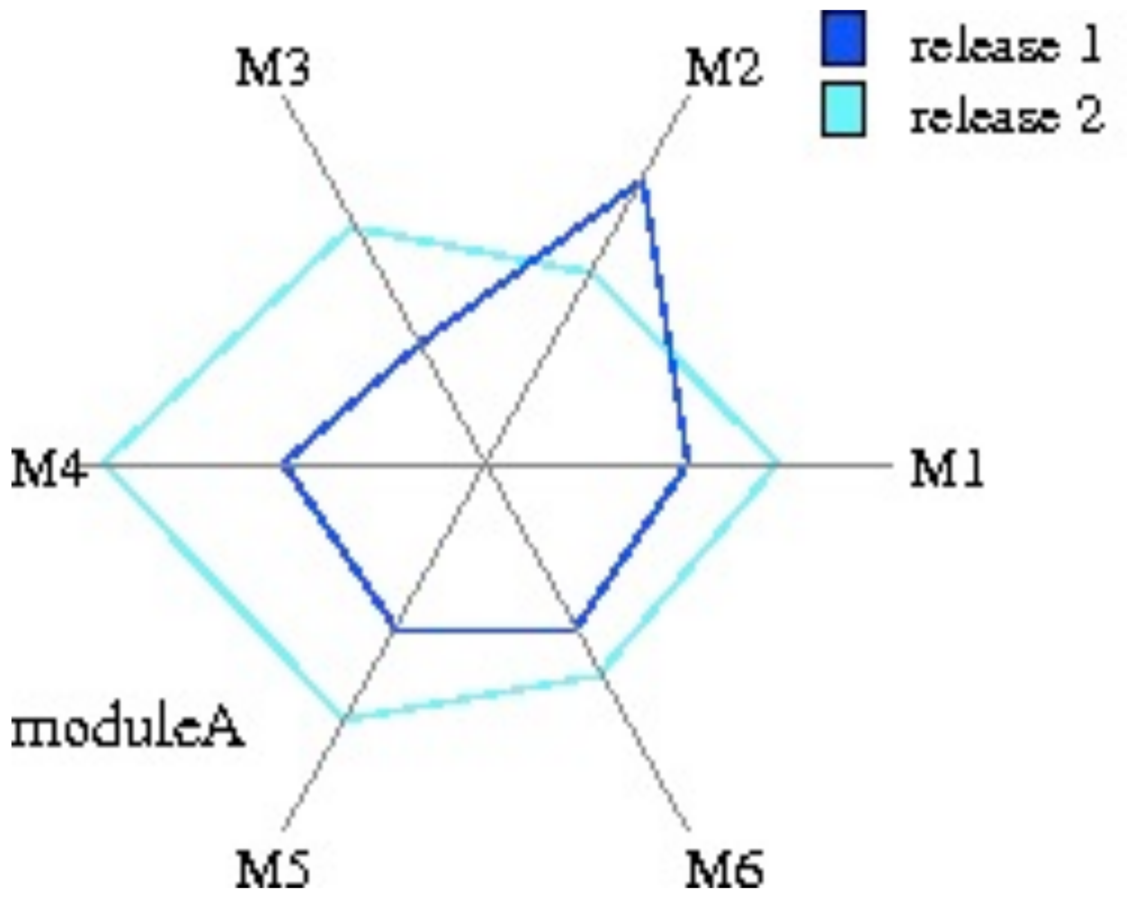
Goal:

- Visualize n metric values of m releases
  - More semantic in graphs
  - More flexibility to combine metric values

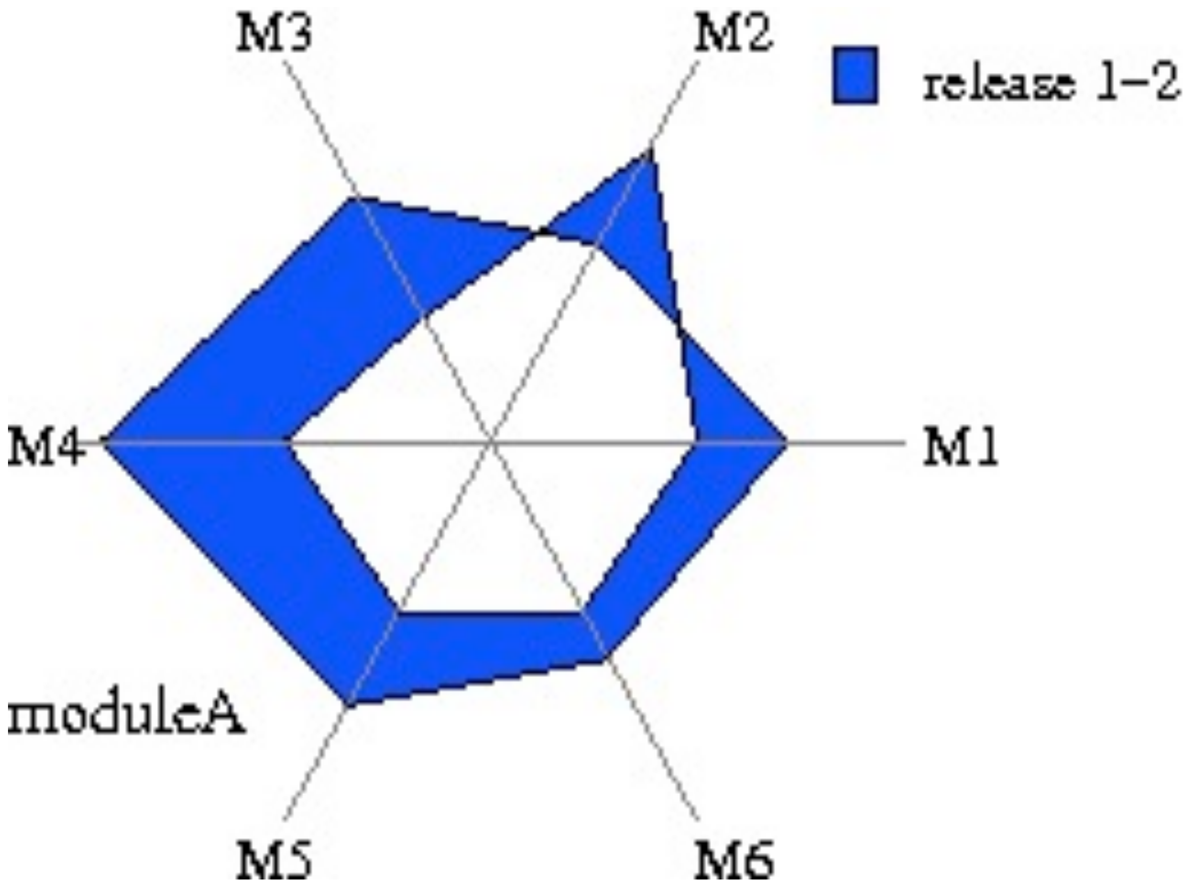
Solution: Kiviat Diagrams (Radar Charts)

- Each ray represents a metric
- Encode releases with different colors

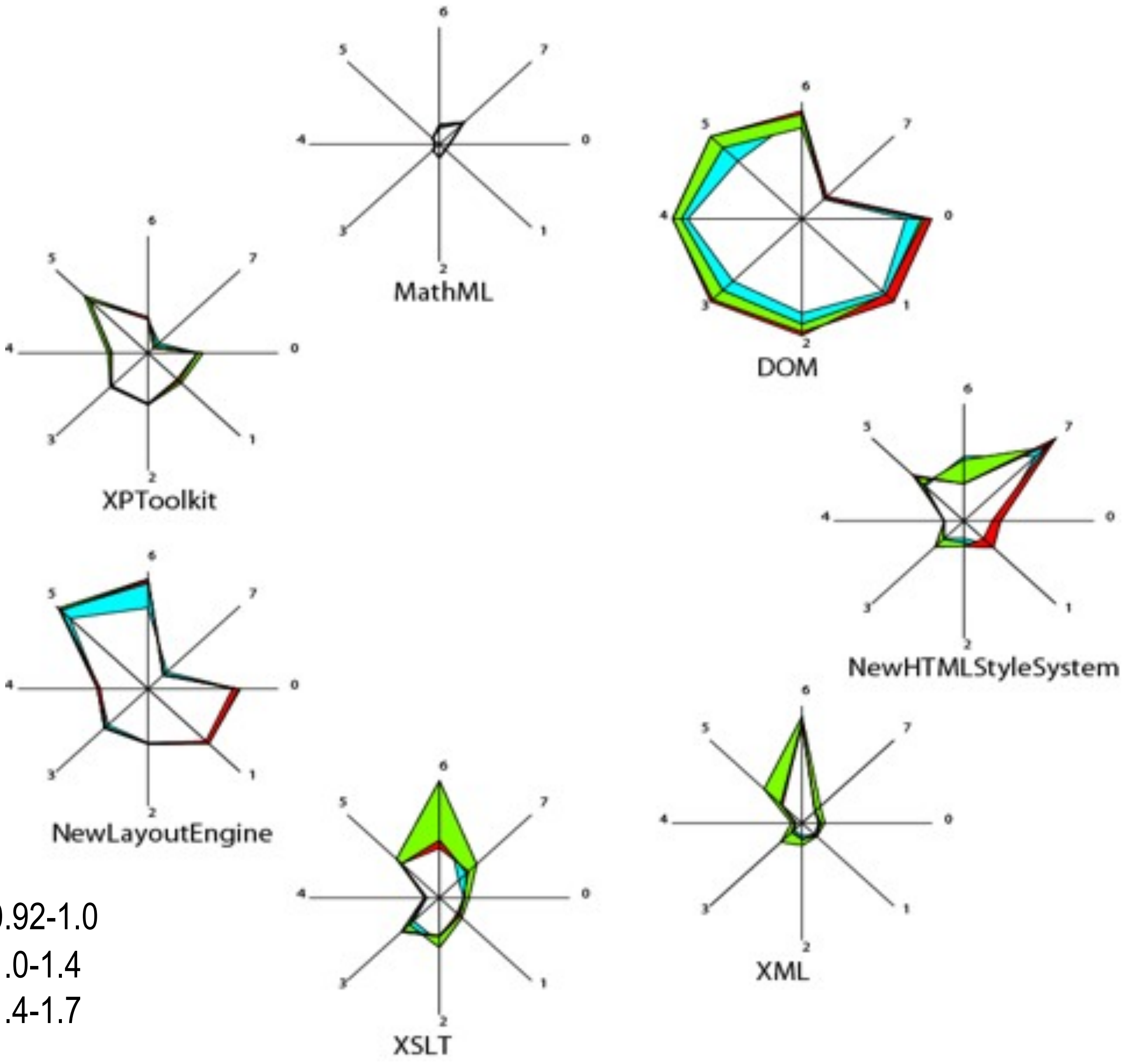
Kiviat Diagram



Highlight the Change



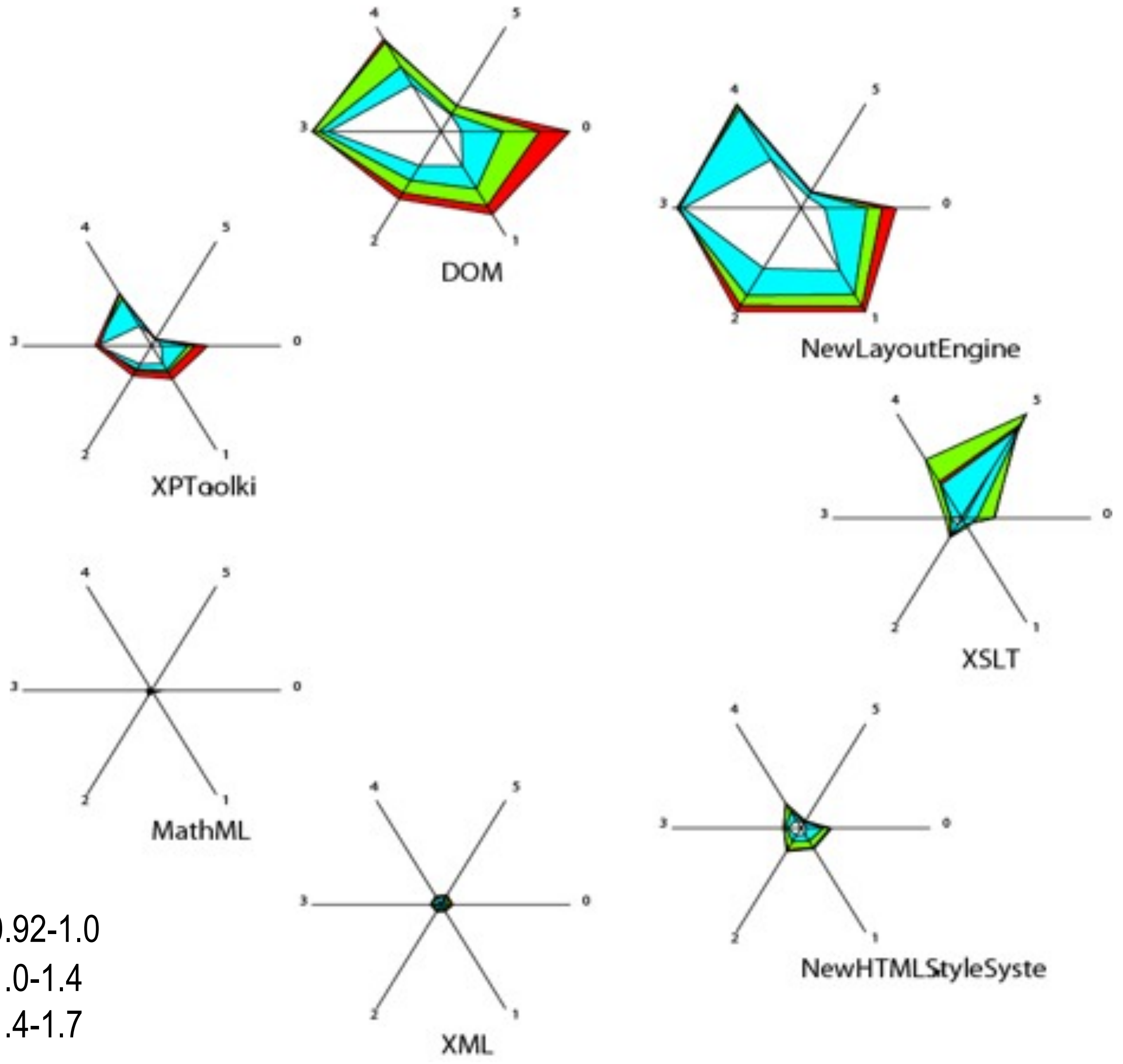
Size & Complexity Metrics



🟦 release 0.92-1.0
🟩 release 1.0-1.4
🟥 release 1.4-1.7

Metrics:
0:nrStmts
1:CCMPLX
2:nrFiles
3:nrClasses
4:nrMeths
5:nrAttrs
6:nrGlobFuncs
7:nrGlobVars

Problem Report Metrics



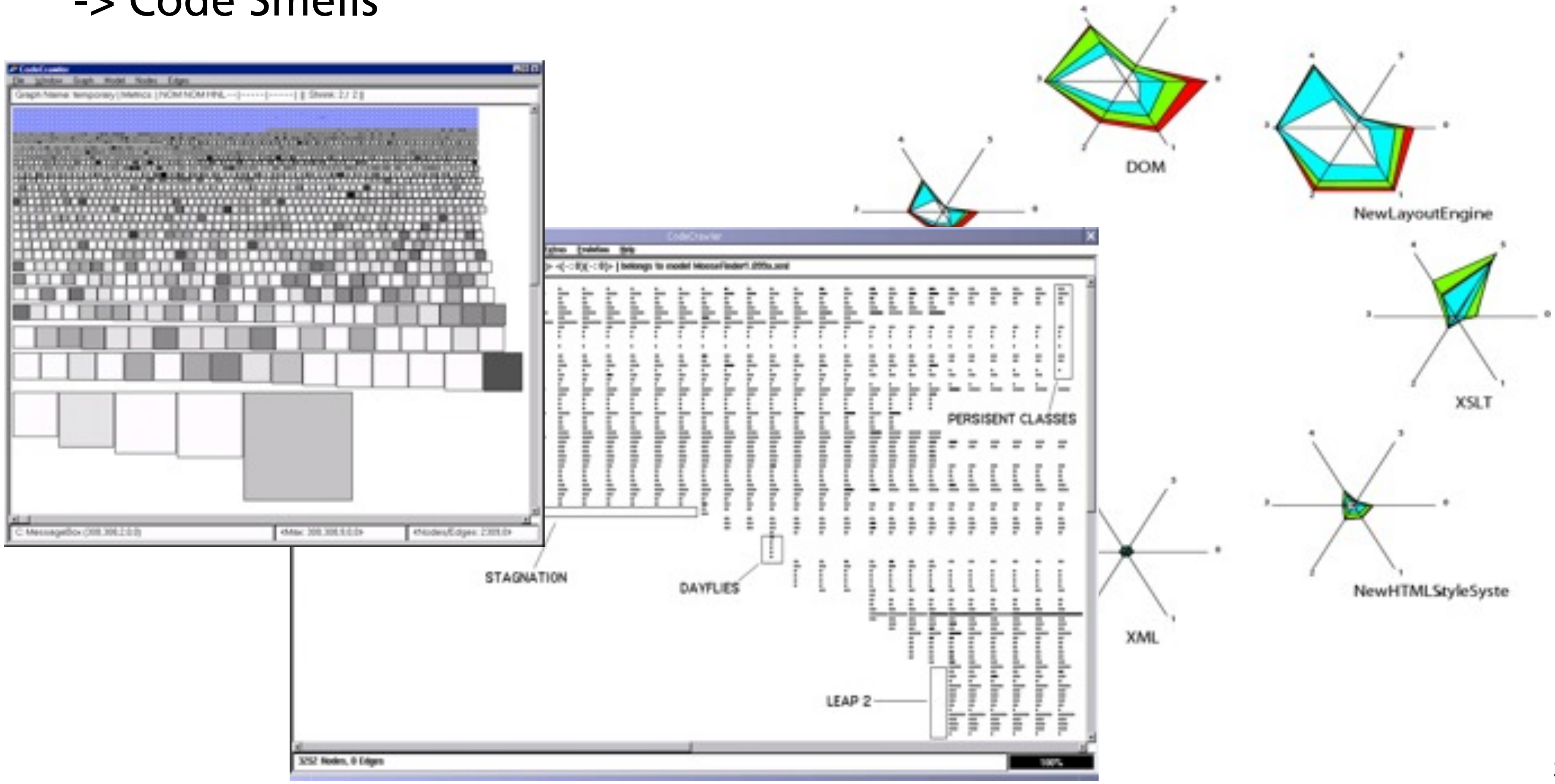
🟦 release 0.92-1.0
🟩 release 1.0-1.4
🟥 release 1.4-1.7

Metrics:
0:nrPrio_undef
1:nrPrio_1
2:nrPrio_2
3:nrPrio_3
4:nrPrio_4
5:nrPrio_5

Conclusions

Design Problems

Result from duplicated, unclear, complicated source code
-> Code Smells


DOM
NewLayoutEngine
XSLT
PERSISENT CLASSES
STAGNATION
DAYFLIES
XML
NewHTMLStyleSyste
LEAP 2