

Mendel: A Model, Metrics, and Rules to Understand Class Hierarchies

Simon Denier & Yann-Gaël Guéhéneuc
Ptidej Team – Université de Montréal

Outline

- Motivation and Problem
- Mendel Model
- Interesting Classes
- Subclassing Behaviors
- Conclusion

Motivation

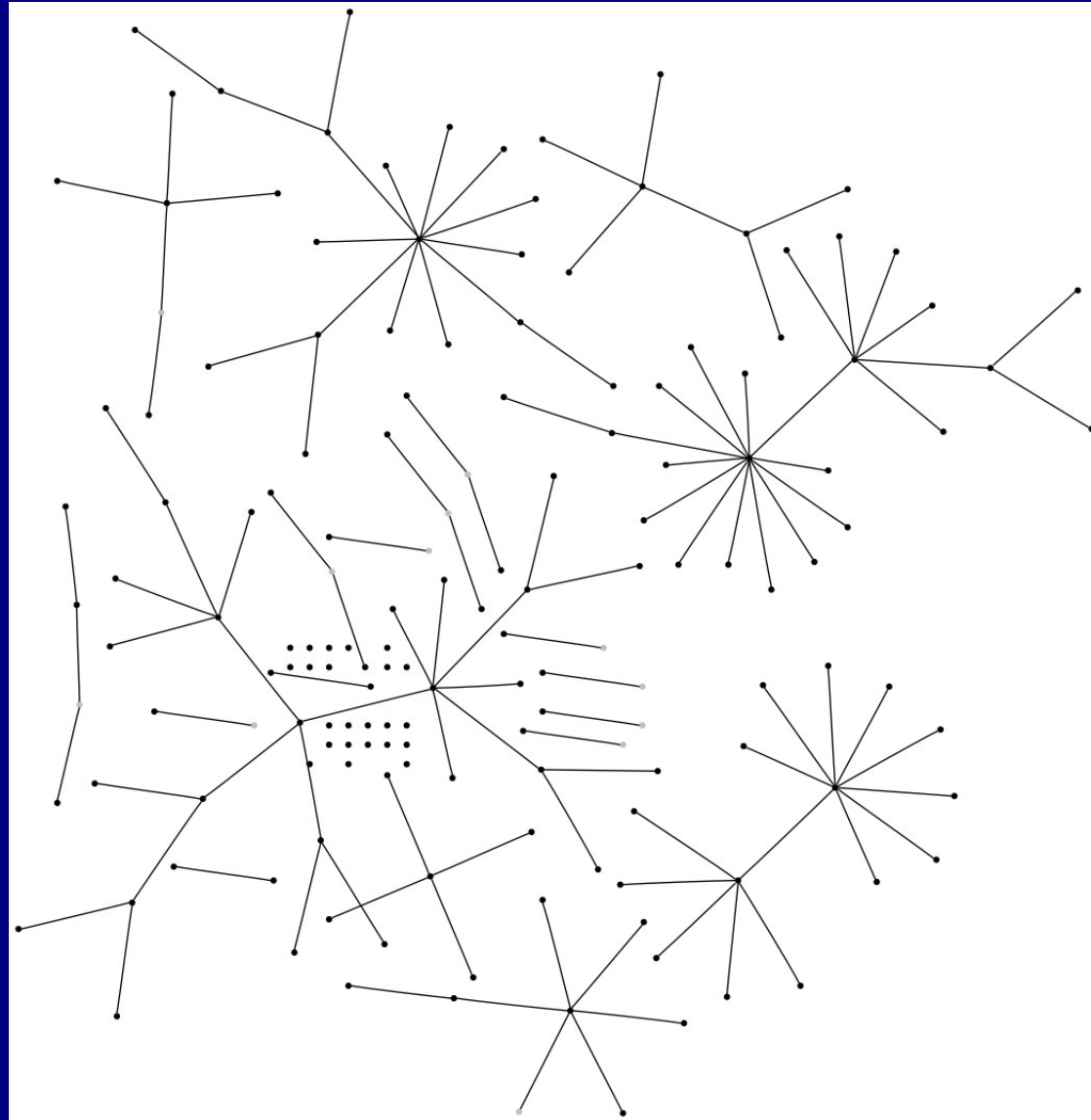
- Context: maintenance of program without documentation
 - Task: identify and understand the role of multiple classes
- Problem 1: where to start?
 - Find entry points (main(...))
 - Read large classes (by LOC or Number of Methods)
- Problem 2: what about inheritance?

Problem: Class Hierarchies

- Classes defined incrementally by inheritance
 - Class “interesting” in its own hierarchy
 - Class profiting from its hierarchy

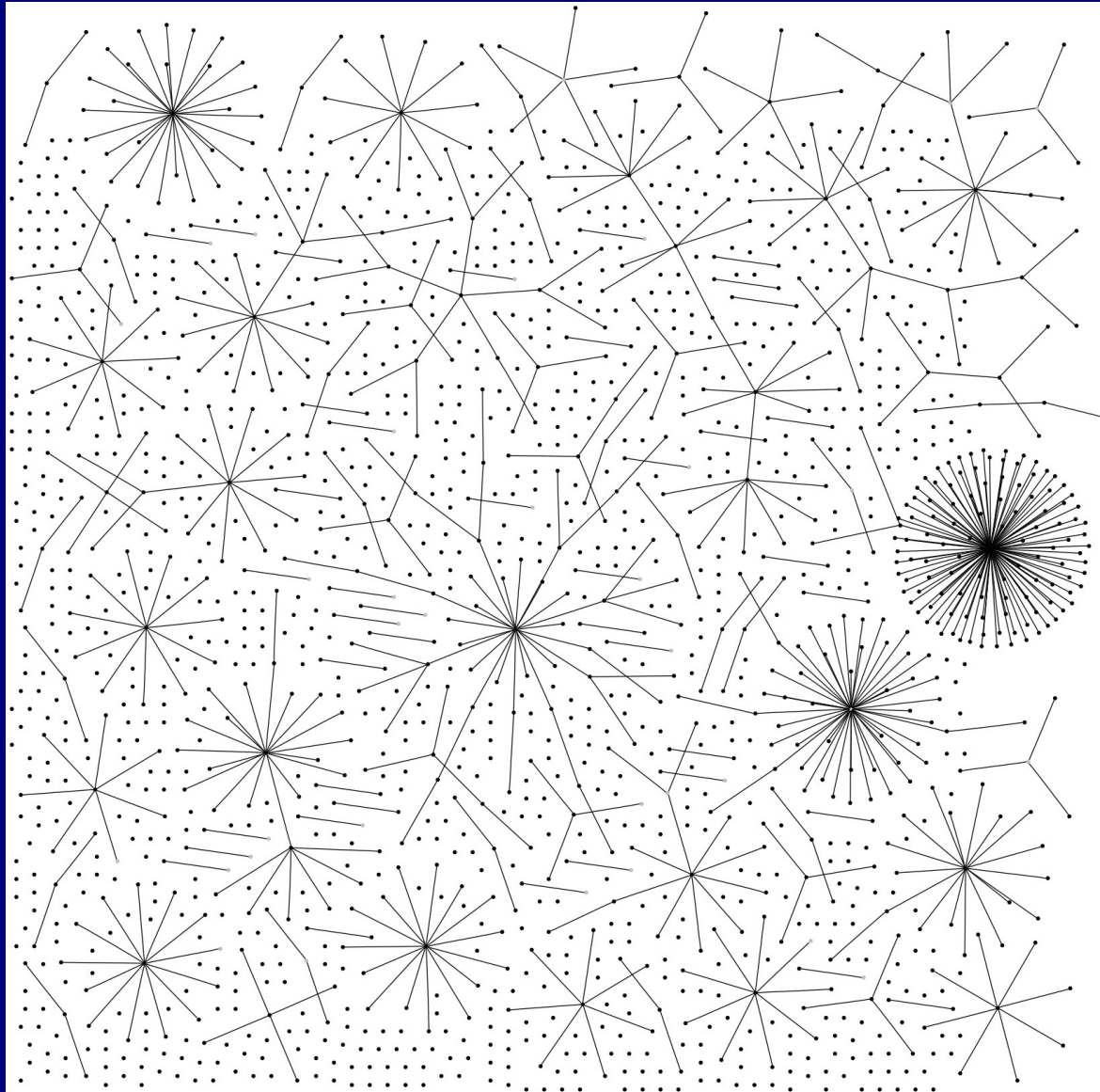
Class Hierarchies

JHotDraw
5.2



Class Hierarchies (2)

Azureus
2.4.0.2



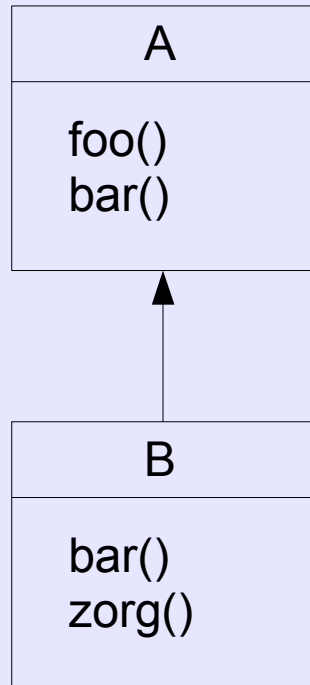
Outline

- Motivation and Problem
- **Mendel Model**
- Interesting Classes
- Subclassing Behaviors
- Conclusion

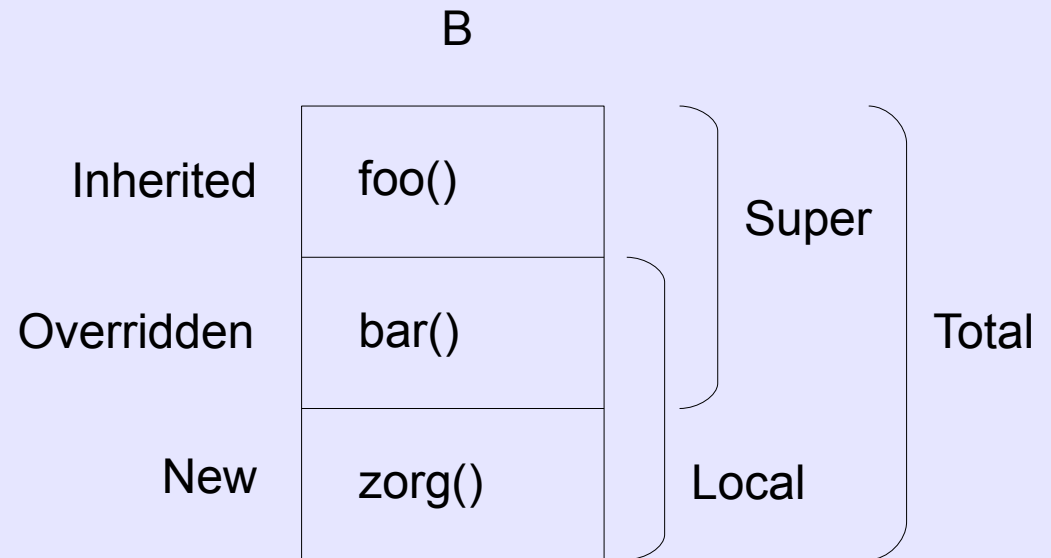
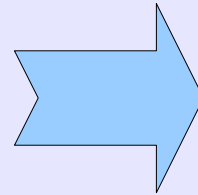
Mendel Approach

- Goal: insights on class hierarchies
 - Interesting classes in their own hierarchy
 - Subclassing behaviors: extending (adding) vs overriding
- Characteristics (requirements) of our approach
 - Early stage in maintenance
 - Intuitive
 - Fast

Mendel: Generic Model



Simple UML



Mendel

Mendel: Metrics and Rules

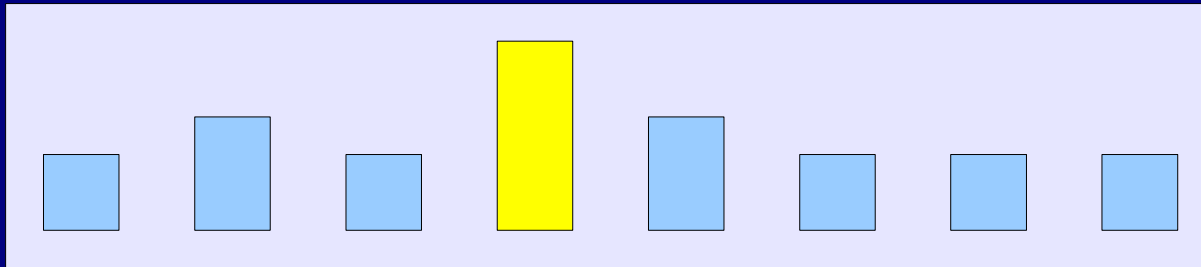
- Simple metrics:
 - Based on interfaces = sets of method signatures
 - Computed on a single class or a hierarchy
- Rules for:
 - Selection (threshold)
 - Classification
 - Comprehension (rule of thumb)

Outline

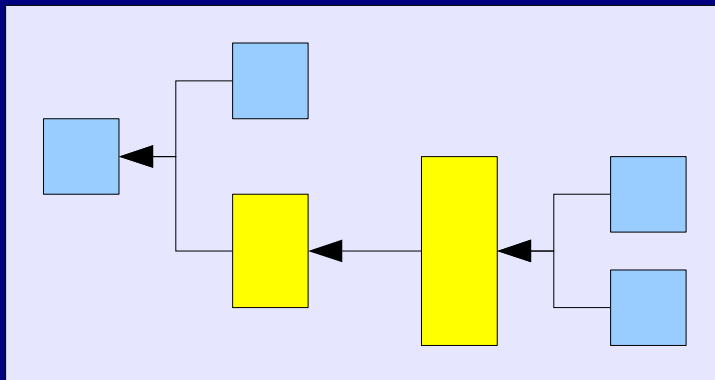
- Motivation and Problem
- Mendel Model
- **Interesting Classes**
- Subclassing Behaviors
- Conclusion

Interesting Classes

- Classes defining a lot of new behaviors
- Large class: overall system



- Budding class: with respect to its hierarchy



- Blooming class: large and budding

Large Class

- Class with a large interface
- Large Class: $|\text{Local}| > \text{threshold}(|\text{Local}|)$
- JHotDraw
 1. StandardDrawingView (inherits from JPanel), 55 methods
 2. DrawApplication (inherits from JFrame), 53 methods
 3. AbstractFigure (inherits from Object), 34 methods

Budding Class

- Class with a large interface with respect to its hierarchy
- Metrics:
 - Branch Mean Size – $bms = |Total| / (DIT + 1)$
 - **Novelty Index** – $nvi = |Local| / bms(\text{superclass})$
- Budding class: $nvi > \text{threshold}(nvi)$
- JHotDraw
 1. AbstractFigure, 3.09
 2. TextFigure, 1.84
 3. PolygonFigure, 1.59

Blooming Class

- Synthesis: combining large and budding
- **Novelty Score** – $nvs = |\text{Local}| \times nvi$
- Blooming Class: $nvs > \text{threshold}(nvs)$

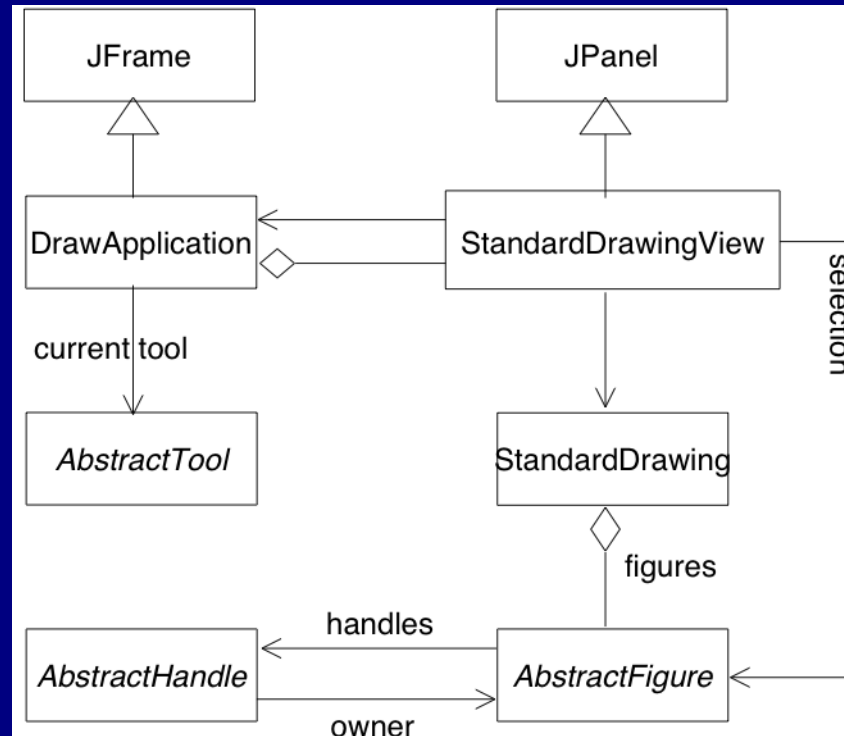
- JHotDraw
 1. AbstractFigure, 105.09
 2. TextFigure, 55.1
 3. LineConnection, 49.55

JHotDraw Overview with Mendel

- JHotDraw 5.2: 148 classes
 - 13 large (9%)
 - 18 budding (12%)
 - 11 blooming (7%)
- Among 11 blooming classes
 - 8 in AbstractFigure hierarchy
 - + DrawApplication, StandardDrawingView, ConnectionTool

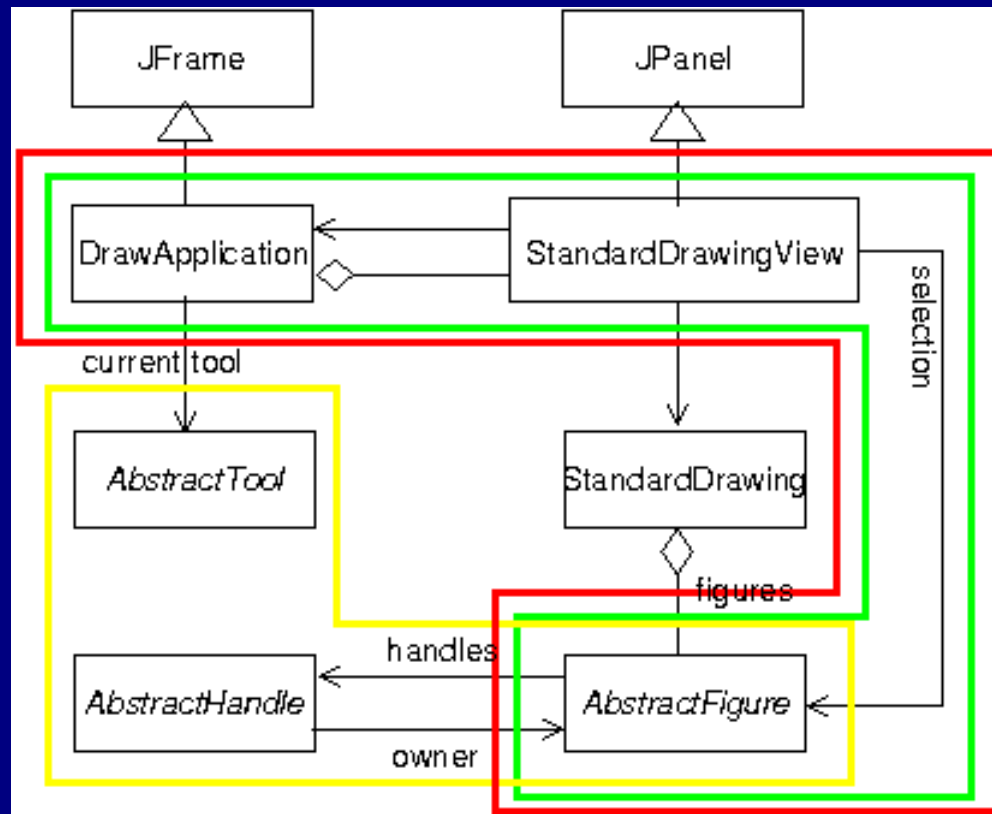
JHotDraw Base Classes

- Given documentation: 6 base classes



Comparison with Documentation

- 5 out of 6 in at least one set
 - AbstractFigure in all 3 sets



Large

Blooming

Budding

What about StandardDrawing?

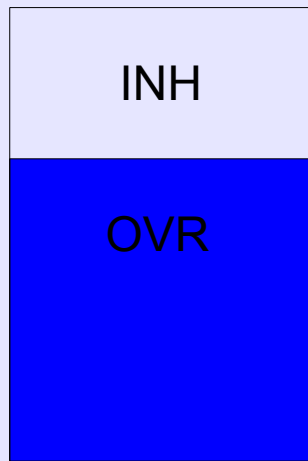
- Max rank: 16th by |Local| order
- Reason: overrider of CompositeFigure (4th blooming class)
- Conclusion: StandardDrawing to be understood in relation with CompositeFigure

Outline

- Motivation and Problem
- Mendel Model
- Interesting Classes
- **Subclassing Behaviors**
- Conclusion

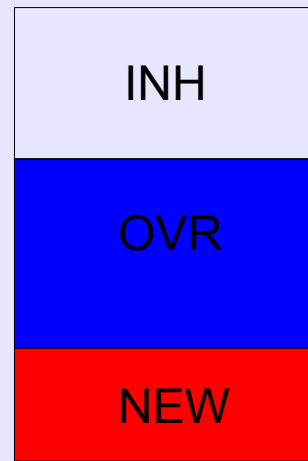
Subclassing Behaviors

Pure Overrider



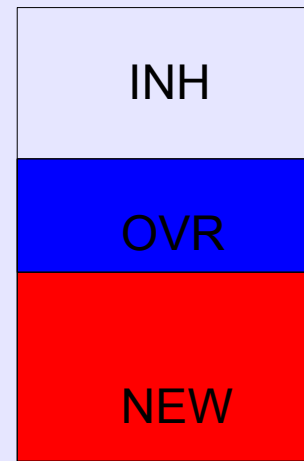
$$|\text{NEW}|=0$$

Overrider



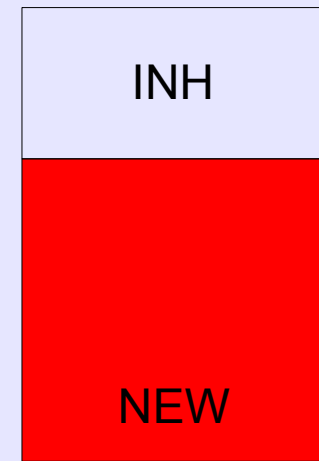
$$|\text{OVR}| \geq |\text{NEW}|$$

Extender



$$|\text{NEW}| > |\text{OVR}|$$

Pure Extender



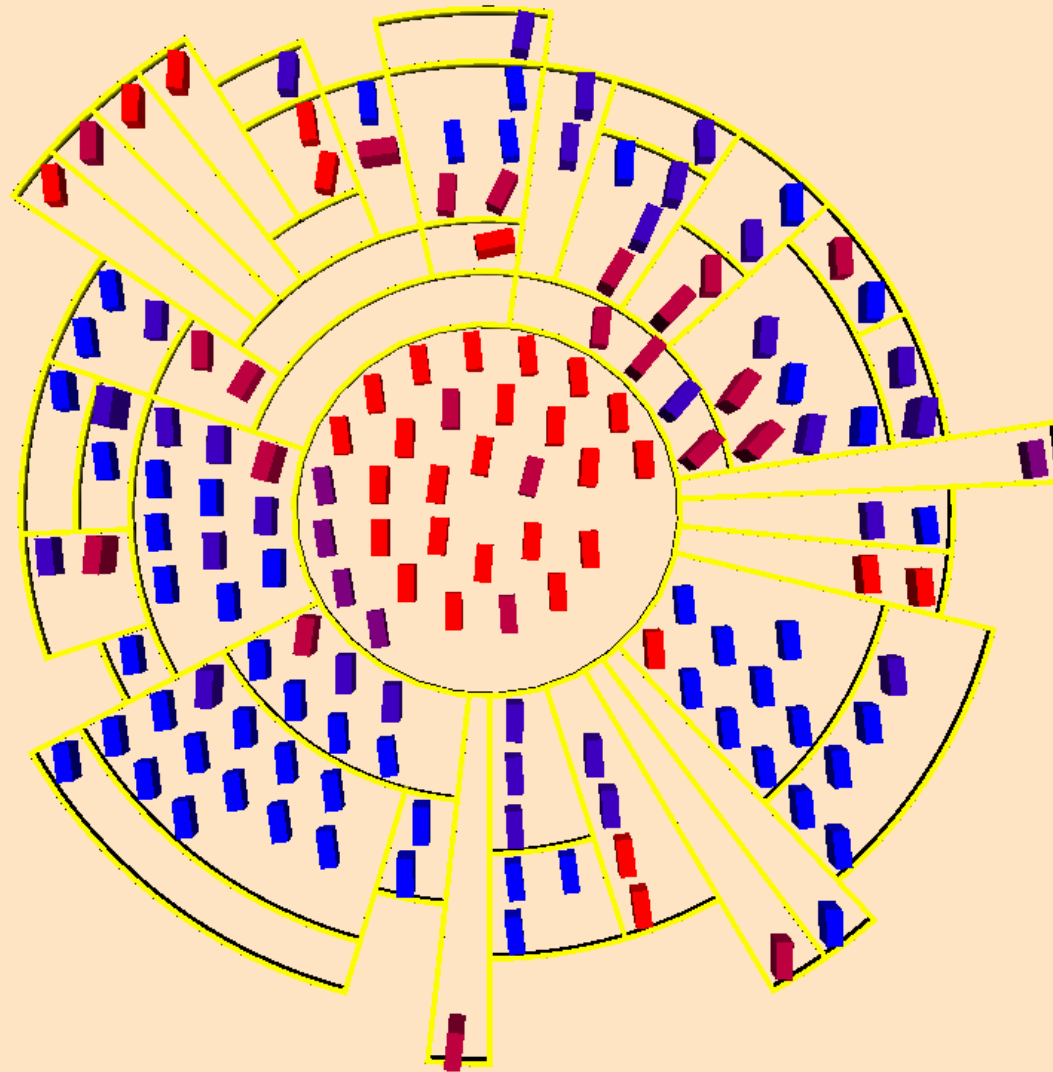
$$|\text{OVR}|=0$$

Extender vs Overrider

- Rules of thumb
- Pure extender
 - Adding new methods: no access through superclass
 - Used for itself
 - Implementation inheritance
- Pure overrider
 - Overriding methods
 - Used through substitution to superclass
 - Interface inheritance

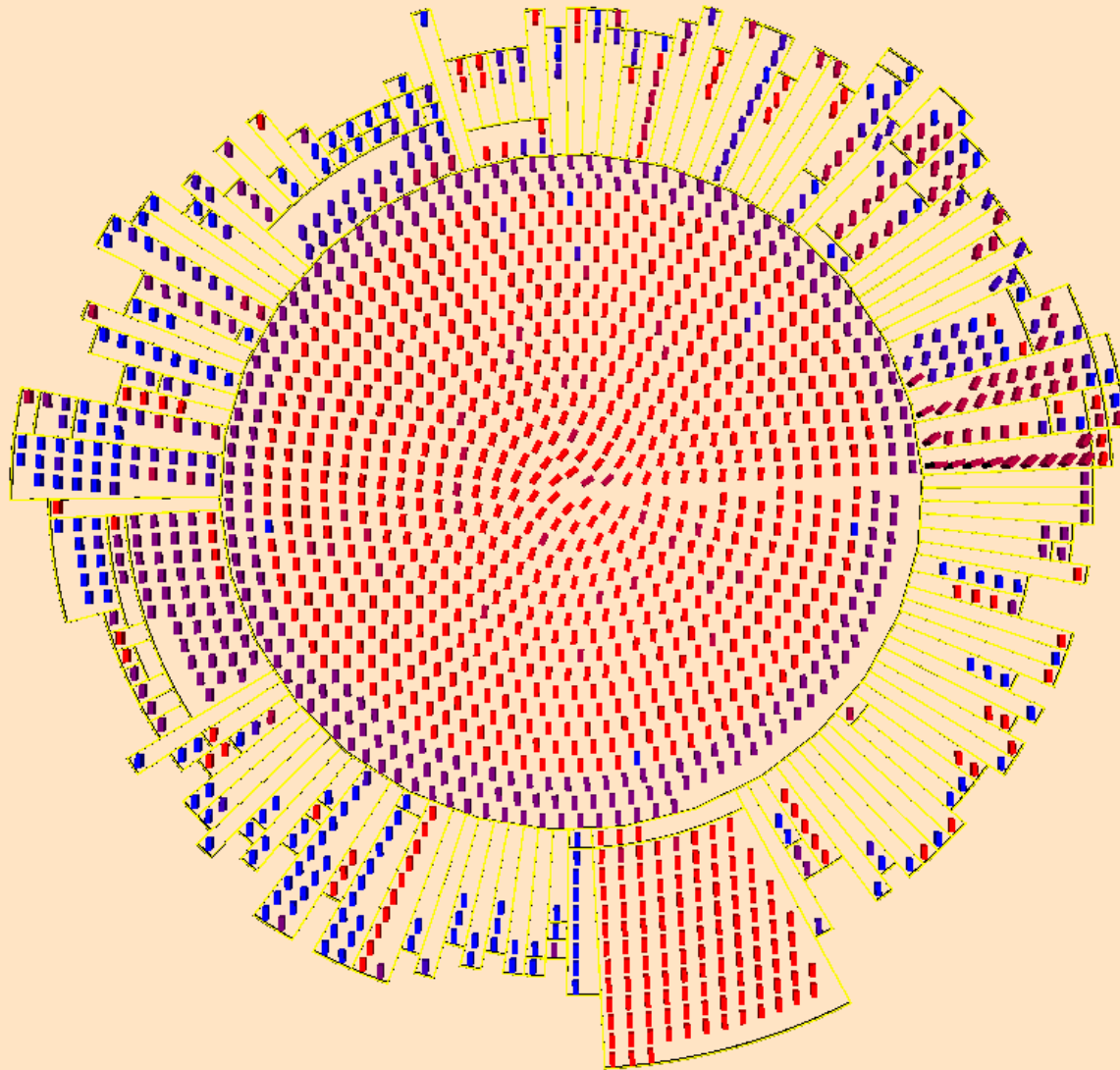
JHotDraw in Verso

fps: 0,4295302



Azureus in Verso

fps: 12,8



Implementation in Java

- Tailored for Java
 - BCEL as class files parser
- Performance for Azureus (1681 classes)
 - Standard desktop machine (2Ghz processor, 1Go memory)
 - 9 seconds

Outline

- Motivation and Problem
- Mendel Model
- Interesting Classes
- Subclassing Behaviors
- **Conclusion**

Conclusion

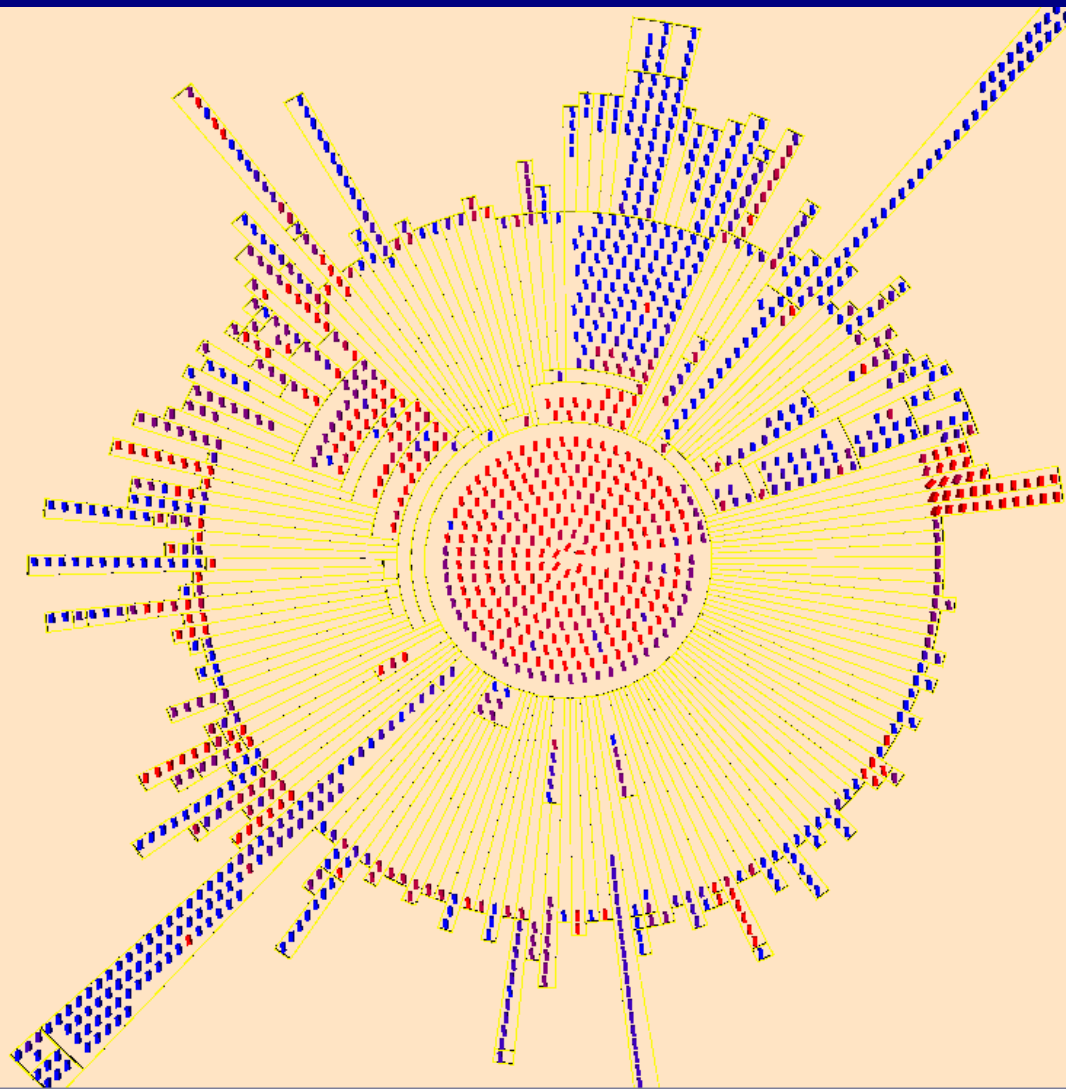
- Mendel
 - Generic model
 - Simple metrics
 - Rules
- Results
 - Quick insights on interesting classes and concerns
 - Large, budding, blooming: different views of interesting classes
 - Concerns: keywords in classnames
 - Different usage of inheritance in programs
 - Some leaned to extending (Log4J, Azureus)
 - Others to overriding (JHotDraw, ArgoUML)

Future Work

- Family analysis
 - Family = class + children
 - Common behavior and interface
- Visualisation
 - Hierarchy representation
 - Visual patterns

ArgoUML in Verso

fps: 0,3615819



Blooming Classes Selectivity

- Log4J: 2 of 206 (1%)
 - LogBrokerMonitor (104 methods), Category
 - Budding set: Appender hierarchy
- ArgoUML: 18 of 1431 (1%)
 - 2 very big classes: NSUMLModelFacade, FacadeMDRImpl
 - Design choice: Facade design pattern
 - Keywords in class names: two products, normal and MDR
- Azureus: 54 of 1681 (3%)
 - Keywords in class names: three concerns, download, peer, DHT

Behaviors Discrimination

- Two categories
 - JHotDraw and ArgoUML: overriding behaviors
 - Large proportions of *Pure Overriders*
 - Azureus and Log4J: extending behaviors
 - Azureus: 58% of classes are *Pure Extenders*

