Understanding the Use of Inheritance with Visual Patterns

Luca Cometta
Motivation

- Number of hierarchies
- Depth
- Size
- Important classes
- Relationship between classes
VERSOS

- 3D objects
- 2D plane
- Classes as 3D boxes
Definitions of the Inheritance Notion

- **Parent class**: a class which has subclasses
- **Childless class**: a class which has no subclasses
- **Family**: the set of direct subclasses of a parent
- **Hierarchy**: the full set of subclasses of a parent
- **Hierarchy root**: the parent class of a Hierarchy whose parent is `Object`
- **Ghost class**: a class which is defined outside the project
Definitions of the Sunburst Layout

- **Slice**: a part of the visualization delimited by edges
- **Subslice**: a slice on the outer edge of the current slice
- **Full Slice**: a slice and all its subslices
- **Eye**: the innermost slice of the visualization
Basic Metrics

• Depth of Inheritance Tree:
  – The number of parent classes up to the root
  – The greater the DIT is, the farther the class is in the inheritance hierarchy

• Number of Children:
  – The greater the NOC is, the more responsibility the class has

• Number of Methods
  – The greater the NOM is, the more services the class can offer to the program
  – Mapped to height
# Subclassing Behaviour

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
<th>Colour</th>
<th>Sample Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure Extender</td>
<td>Adding only new methods</td>
<td>Bright red</td>
<td>A</td>
</tr>
<tr>
<td>Extender</td>
<td>Adding more methods than overriding</td>
<td>Red</td>
<td>C</td>
</tr>
<tr>
<td>Other</td>
<td>No method defined</td>
<td>Purple</td>
<td>X</td>
</tr>
<tr>
<td>Overrider</td>
<td>Overriding more methods than adding</td>
<td>Blue</td>
<td>Y</td>
</tr>
<tr>
<td>Pure Overrider</td>
<td>Only overriding methods</td>
<td>Bright blue</td>
<td>B, D, Z</td>
</tr>
</tbody>
</table>
Children Similarity

- Based on similarity of the method signatures
- Prototype is computed
- Similarity to prototype is computed

\[
sim(i_1, i_2) = \frac{|i_1 \cap i_2|}{|i_1 \cup i_2|}
\]

<table>
<thead>
<tr>
<th>Class</th>
<th>(m_1)</th>
<th>(m_2)</th>
<th>(m_3)</th>
<th>(m_4)</th>
<th>(m_5)</th>
<th>(\sim)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_1)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>2/5</td>
</tr>
<tr>
<td>(C_2)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(C_3)</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>Prototype</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

- Mapped to Twist
- Similar children are vertical
- Dissimilar children are horizontal
A First Example: JHotDraw

A: Swing framework extension
B: High use of overriding for polymorphism
C: Main hierarchy of class Figure
Visual Patterns

• **Large Root**: root slice with a large radius
• **Big Hierarchy**: full slice with deeply nested slices and slices with many entities or subslices
• **Common Family**: red parent and mostly blue children
Polymorphic Hierarchies

- **Large Family**: a slice with many entities, often childless
- **Similar Children**: a slice with many vertical entities
- **Nested Families**: nesting of the previous patterns
- **Library / Framework Extension**: full slice with nested ghost parents parenting entities of mixed properties
- **Exception Hierarchy**: Full slice with nested ghost parents and a majority of purple entities
Program Analysis

A three-step process:

1. An overview of inheritance use is provided to answer general questions
2. Focus on particular classes or hierarchies to identify visual patterns
3. For each particular case it’s possible to request more information
Case Study 1: JFreeChart

- Many nested slices
- Rather small hierarchies
- Many red entities
- Three big hierarchies
- Entries display mixed properties
- Two big slices
- Many tall classes in A
- C: Swing framework extension
- Horizontal twist of purple entities in the eye
Case Study 2: Xalan

• Many slices
• The eye is relatively small
• Slices are deeply nested with large root slices
• Entities are mostly blue
• A and E show strong polymorphic patterns
• B, C and D display mixed polymorphic hierarchies
• F matches the exception hierarchy pattern
Case Study 3: Azureus

- Big eye
- Few big hierarchies
- Most hierarchies are shallow
- Blue and red entities are mixed
- A is a big hierarchy used to encode basic datatypes
- The other big hierarchy is the exception hierarchy E
- C is unusual: similar children implementing the same interface (TableCellRefreshListener)
- D and F are occurrences of polymorphic families
Conclusion

- Scalable inheritance visualization
- Display many characteristics of inheritance at program as well as at class level
- Developed a new metric for child similarity

- Major limitation: single inheritance
- Sunburst layout can display improper effects
Questions?