Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Solution

Assignment 09 — 14/11/2018 – v1.0a
Smalltalk: Software Visualization

Please submit this exercise by mail to sma@list.inf.unibe.ch before 21 November 2018, 10:15am, and **use the *file out* feature in the *System Browser* to export the Smalltalk classes you created.**

Note: For the following exercises you should use the pre-configured Moose 6.1 environments available in 32 bit flavor for Linux, Windows, and macOS. Please choose the correct version for download in accordance with your current platform.

Exercise 1: Sunburst visualization with Roassal (2 Points)

Build a Sunburst visualization as shown in Figure 1 to analyze the test coverage of the `Collection` class hierarchy. Each tile represents a specific class, and the size of the tile should represent its number of lines of code. Moreover, tested classes (i.e., classes covered by tests) should be colored in green, while other classes should remain in grey.

Hint: You can assume that test classes (i.e., classes that test other classes) use a name which closely resembles the original name of the class they test; in general, they add only the postfix `Test` to the original class name (e.g., `ByteArray` will become to `ByteArrayTest`).

Figure 1: Sunburst visualization built with Roassal

Answer:

```
| b |
b := RTSunburstBuilder new.
b layout sunburstWithRadius:  100.
b
  angularSpacing:  1;
  radialSpacing:  5.
b shape
  color:  [ :cls |
    (Smalltalk includesKey:  (cls name , 'Test') asSymbol)
      ifTrue:  [ Color green ]
      ifFalse:  [ Color gray ] ].
b leafWeight:  [:e | e numberOfLinesOfCode].
b explore:  Collection using:  #subclasses.
b view elements
  @ (RTLabeled new
    color:  Color black; fontSize:  20;
    setAsShouldBeInTheFront;
    center).
^ b
```

Exercise 2: Tree map visualization with Roassal (2 Points)

Build a tree map visualization as shown in Figure 2 to gather an overview of classes that have subclasses, and that contain the string `Array` in their names. The size of the tiles must encode the number of methods of the class they represent. Furthermore, classes that contain the string `Array` in their names and that have subclasses should be colored in green, while all remaining classes should remain in grey.

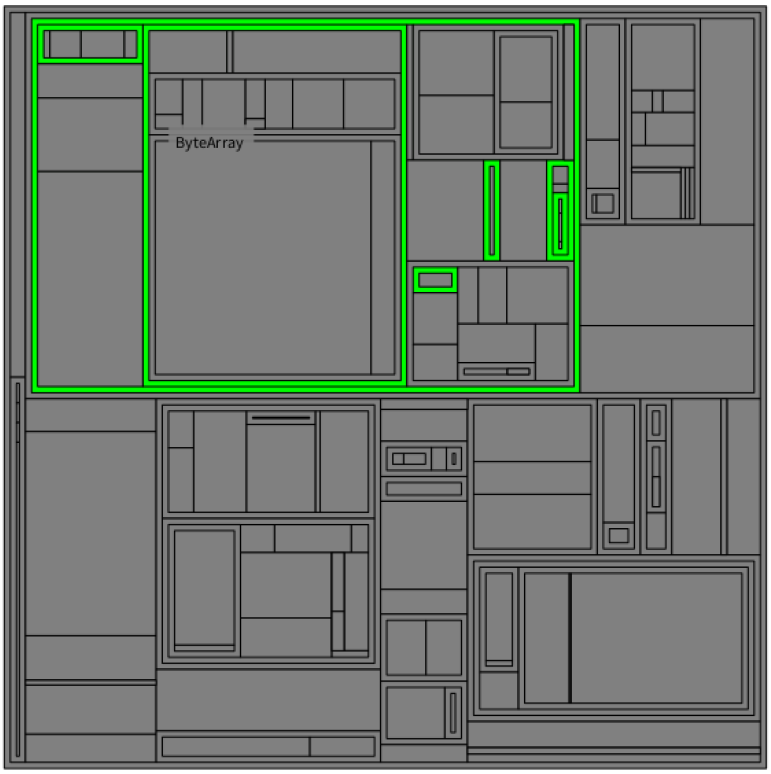

Figure 2: Tree map visualization built with Roassal

Answer:

```
| b |
b := RTTreeMapBuilder new.
b shape
  color: Color transparent;
  borderColor: Color black;
  if: [ :obj | obj isClass ]
    color: [ :cls |
      (cls subclasses notEmpty and:[('*Array*' match: cls name)])
        ifTrue: [ Color green ]
        ifFalse: [ Color gray ] ] .
b
  leafWeight: [:e | e methods size];
  explore: Collection using: #subclasses.
^ b
```

Exercise 3: Node-link visualization with Roassal (3 Points)

Node-link diagrams as shown in Figure 3 can provide valuable insights of the relationships amongst the classes in a system. In this exercise you have to create a visualization using the `Mondrian` builder to analyze the class dependencies between the `Collection` class hierarchy and the `RTLayout` class hierarchy. To this end, you have to:

i) Visualize the classes of both hierarchies using circles (i.e., `RTEllipse`)

ii) Use a different color to differentiate the classes of each hierarchy.

iii) Add edges to depict the class hierarchy, while using the `RTClusterLayout`

iv) Add blue Bézier edges to depict class dependencies

Provide the Roassal code to implement such a visualization, and use it to identify the classes in each hierarchy that have the highest number of dependencies.

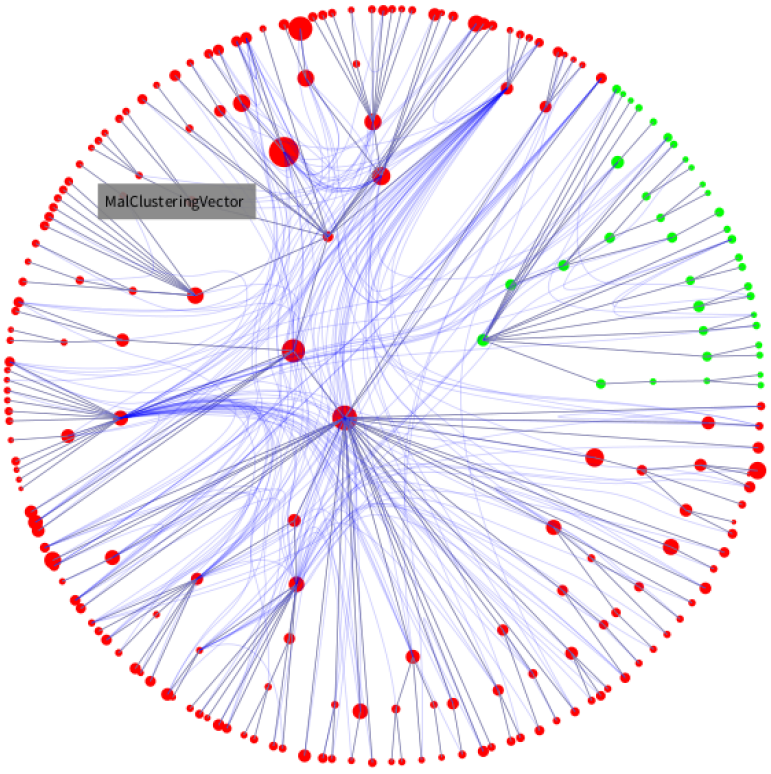

Figure 3: Node-link visualization built with Roassal

Answer:

```
| classes b |
classes := RTLayout withAllSubclasses , Collection withAllSubclasses.
b := RTMondrian new.
b shape circle; color:[:cls | (RTLayout withAllSubclasses includes:  cls)
ifTrue:[Color green] ifFalse:[Color red]].
b nodes:  classes.
b edges connectFrom:  #superclass.
b shape
  bezierLineFollowing:  #superclass;
  color:  (Color blue alpha:  0.2).
b edges
  notUseInLayout;
  connectToAll:  #dependentClasses.
b normalizer
  normalizeSize:  #numberOfMethods using:  #sqrt.
b layout cluster.
b build.
^ b
```

Exercise 4: Discussion (3 Points)

Comment on the strenghts and limitations of each visualization you just created.

Answer:

Sunburst visualization. *The sunburst visualization provides a nice representation for hierarchies; each ring from the center moving outwards corresponds to a level in the hierarchy. The rings are sliced into numerous tiles, each of them revealing the relationship to the parent tile. Besides those advantages, anything related to the covered area of specific tiles is hard to evaluate manually, since humans cannot accurately estimate areas of circular sections.*

Tree map visualization. *The advantages are similar to those of the sunburst visualization, however, the area of each tile is much easier to compare, since the area of the tiles is always rectangular. Moreover, this eases very much part-to-whole comparisons. One drawback is the increased difficulty of comparing elements in different tiles, but at the same hierarchy level. Another downside is the more difficult manual inspection of the plot: the parent elements are very hard to hit with the cursor due to their small size.*

Node-link visualization. *Node-link visualizations combine the best of both worlds: they support multiple overlaid relation visualizations (in our example using Bézier curves and straight lines) and represent the hierarchy level of each element in an intuitive manner. Unfortunately, these plots require a very high resolution to be of any use, because of their heavy utilization of tiny connections between the different shapes.*