Onion Graphs for Focus+Context Views of UML Class Diagrams

Huzefa Kagdi
(PhD Candidate)

Dr. Jonathan I. Maletic
(Advisor)
A Typical Software Evolution Case

• Software developers “focus” on a small subset of a software artefact
  – e.g., source code, UML design, etc.
• Design a system with a number of small UML class diagrams
  – e.g., a feature, a concept
• Classes specific to an evolution case
  – e.g., a bug-fix, a feature enhancement
Focus Necessary but may not be Sufficient

- Individually created diagram may need to be interpreted in the "context" of the overall design
  - e.g., comprehension, requirements coverage

- Impact of changes to the specified classes may need to be considered in the "context" of the rest of the system
  - e.g., broken dependencies, ripple effects, discover unknown dependencies
The Problem

The focus area is needed in detail. The remainder of the model (the context) is needed at various level of detail.
Focus+Context Views [Kadmon’78]

- Analogous to wide-angle camera lens
  - Distorted views
- Fisheye Views [Furnas’81]
- Perspective wall [Mackinlay’91]
- Document Lens [Robertson’93]
- Hyperbolic browsers [Lamping’95, Munzner’95]
Our Approach

Given a UML class diagram (model)

- determine the focus (i.e., a subset of the model)
  - manually
  - automatic via UML model slicing [Kagdi’05]

- represent the focus in standard UML notations

- visually abstract the context (i.e., rest of the diagram)
  - what to abstract?
  - how to represent visually?
  - transformation technique
What to Abstract? …Edges

• Common graph drawing algorithms [Purchase’97]
  – Minimize edge crossings and bends, and maximize symmetry

• Subjects preference in UML class and collaboration diagrams [Purchase’00]
  – Fewer bends and crosses, shorter edge lengths, and an orthogonal structure

• UML class diagram comprehension
  – Inconclusive except for minimizing bends

Need to eliminate/reduce Edges
How to Represent Visually? ... Onion Notations

- **Onion notations** [Sindre’93]
  - Nested graphs i.e., nodes within nodes
  - Alternative to using edges
  - Mostly used for representing hierarchical relations

- **Onion graph**
  - Combination of onion and pure graph notations

- **Different types of relations can be distinguished by shapes, color, line types, or annotations**
Examples of Onion Graphs

Single Onion

Hicon
Onion Notations for UML Class Diagrams

Pure-onion Notations

Mix-onion Notations

Class (a)
2-Relation Class (b)
Generalization (c)
Dependency (d)
Association (e)
Aggregation (f)
Combinations (g)

Classes and Classes with Generalizations Partitions (a)
Classes with Generalizations and Classes Partitions (b)
Classes with Associations and Classes with Generalizations and Classes Partitions (c)
Empty, Classes with Generalizations, and Empty Partitions (d)
An Example
Transformation Technique

- Selective aggregation algorithm
- Two alternating operations
  - sibling-order compaction (i.e., reduction at the same levels)
  - level-order compaction (i.e., reduction across different levels)
  - constrains preserving UML structure and semantics of the represented class diagram
Initial UML class Diagram
Intermediate Views

Diagram showing a hierarchy of classes and transformations.
Multiple Inheritance
A Focus+Context View
## Evaluation

<table>
<thead>
<tr>
<th>Onion Graph</th>
<th>Edges</th>
<th>Nodes</th>
<th>Edge Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5</td>
<td>10</td>
<td>11</td>
<td>0.48</td>
</tr>
<tr>
<td>Figure 6</td>
<td>8</td>
<td>9</td>
<td>0.47</td>
</tr>
<tr>
<td>Figure 7</td>
<td>6</td>
<td>7</td>
<td>0.46</td>
</tr>
<tr>
<td>Figure 8</td>
<td>5</td>
<td>6</td>
<td>0.45</td>
</tr>
<tr>
<td>Figure 9</td>
<td>4</td>
<td>5</td>
<td>0.44</td>
</tr>
<tr>
<td>Figure 10</td>
<td>3</td>
<td>4</td>
<td>0.43</td>
</tr>
<tr>
<td>Figure 11</td>
<td>21</td>
<td>18</td>
<td>0.53</td>
</tr>
<tr>
<td>Figure 12</td>
<td>21</td>
<td>18</td>
<td>0.53</td>
</tr>
<tr>
<td>Figure 13</td>
<td>11</td>
<td>11</td>
<td>0.50</td>
</tr>
<tr>
<td>Figure 14</td>
<td>7</td>
<td>8</td>
<td>0.46</td>
</tr>
</tbody>
</table>

**Case 1** – Single Inheritance only  
**Figure X** – Focus+Context Views  
**Case 2** – Including Multiple Inheritance  
**Edge Density** – $|\text{edges}| / (|\text{edges}| + |\text{nodes}|)$
Related Work [Musial’03]

- Focus+context technique
- Goal is visual-space efficiency (24 vs 75)
- Only a single class considered in focus
- Aggregated elements depends on physical (i.e., graphical) distance
- Symbols representing selective aggregations doesn’t give any information about the abstracted elements (i.e., lossy context)
- Does not preserve original layout
Conclusions

• Onion Notations
  – preserve UML symbols and semantics

• Transformation technique
  – Reduces cognitive load (reduces edges)
  – preserves layout

• Conjecture
  – onion graph representation of UML class diagram produces focus+context views that provide high degree of support for exploration and explanation tasks
Future Work

• User studies for validating our conjecture
• Other transformation techniques
• Integration with existing UML tools
Questions???
Possible Solutions

• Dynamic queries to filter the information space [Shneiderman’94]
  – lossy (risk of filtering out the most relevant data)
• Everything in detail or detail on demand [card ‘99]
  – e.g., scroll, zoom, and pan
  – gives detail but loses context or vice versa
• Multiple co-ordinated views [card ‘99]
  – requires user to synthesis views
• Focus+context [card ‘99]
  – both detail and overview in a single view
UML Class model as a Graph

• Multi-graph
  – Multiple edges (relations) between a pair of nodes (classes)

• Mixed-graph
  – Hierarchical (e.g., generalizations and aggregations)
  – Non-hierarchical (e.g., associations and dependencies)
  – Reflection (self-associations)

• Domain-specific
  – Object-oriented analysis and design
Preserve UML Syntax and Semantics

- Similarity with the UML notations for elements and relationships
- Structure and semantics of the represented class model (diagram)