Towards Anomaly Comprehension
Using Structural Compression
to Navigate Profiling Call-Trees

Shen Lin, François Taïani,
Thomas C. Ormerod, Linden J. Ball

Lancaster University (UK)
Why visualise anomalies?

- Software easier to construct than to **understand**
  - as climbing up vs. climbing down

- Particularly true for **emergent dynamic behaviours**
  - e.g. dependability and **performance**
  - arise from interactions within/without target system
  - phenomena might span large set of components
  - at different levels of abstraction

- **Dynamic** observation data difficult to make sense of
  - **large** size (up to 1,000,000 invocations for a few sec)
  - likely to involves **poorly understood** parts
Example: Performance

- Piece of grid middleware (Globus ws-core)

impossible to vary level of abstraction (methods)

difficult to relate different parts of the data

impossible to abstract locally (methods)
Visualising Dynamic Data

- Strategies: extract **salient features**
  - collapse recurring patterns (e.g. Jinsight)
  - remove and focus (e.g. CosmOpen)
  - data-mining (e.g. PCA in Xu et. al, SOSP’09)

- Our take in this work
  - use **structural information** in dynamic data
    - e.g. `org.apache.axis.utils.ClassUtils.forName()`
  - vary ‘**local abstraction**’ level at which data is shown

- Contributions
  - application of structural compaction to **dynamic data**
  - algorithm for **localised structural compaction**
  - prototype and **explorative user study**
Dynamic Profiling Data

- Toy example: weighted call tree (sample based profiling)
Full compaction

- Only highest level packages visible
Progressive exploration

- Different levels of compaction in different parts of graph
  → including for the same package (here lib3)
Realisation

- **Uniform compaction reasonably well-understood**
  - e.g. merge all ‘org.globus.xx’ nodes together
  - used by other tools (e.g. Creole for static analysis)

- **Challenges** here:
  - same entity → different compaction in different areas
  - but uniform compaction locally (otherwise confusing)

- **Question**
  - How do we encode these different ‘localised’ levels?
  - Which mechanisms to drive ‘local’ compaction?
Compaction Algorithm

- Local compaction level
- ‘Take over mechanisms’
  - use ‘vicinity’ relation
  - cascading compaction

More details in paper
Demo
Evaluation

Design

- small scale and exploratory (4 users)
- 2 categories of programs (‘small’ and ‘large’)
- two tools: ProfVis and Treetable
- each user applied both tools to both categories
- task: identify performance issues

Two quantitative measures

- perceived understanding (from test user)
- assessed understanding (from us)

<table>
<thead>
<tr>
<th></th>
<th>LoC</th>
<th>classes</th>
<th>methods</th>
<th>prof.</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>BubbleSort</td>
<td>59</td>
<td>2</td>
<td>7</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Simulation</td>
<td>140</td>
<td>5</td>
<td>16</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>OPSBrowser</td>
<td>13624</td>
<td>172</td>
<td>1002</td>
<td>1059</td>
<td></td>
</tr>
<tr>
<td>ws-core-3.9.4</td>
<td>42477</td>
<td>432</td>
<td>2550</td>
<td>1341</td>
<td></td>
</tr>
</tbody>
</table>
Results

![Graph showing perceived understanding vs. assessed understanding with different symbols for treetable and profvis with small and large datasets.](image-url)
Results (2)

- Users seem
  - to **overestimate** their understanding on large programs with TreeTable
  - to **underestimate** their understanding on large program with ProfVis

- Possible explanations
  - TreeTable hides full scope while ProfVis does not
  - As a result possible ‘false sense of mastery’
Results (3)

- Usage patterns seem to support this interpretation
  - users go deep w/ TreeTable, tend to hover w/ ProfVis

![Graphs showing depth of interaction over time for TreeTable and ProfVis]
Results (4)

- Understanding
  - slight advantage for Provis
- But
  - very small study!
  - informally, users preferred TreeTable
- Questions raised
  - acceptability
  - ergonomics
  - design combination
Conclusion

- A novel **localised structural compaction** algorithm
  - applied to performance profiling traces
- Preliminary **evaluation**
  - potential gap btw. user perception and their actual perf.
  - and btw. subjective preference and actual perf.
  - impact of presentation on search strategy
- **Future** work
  - larger study!
  - combine design dimensions (e.g. text + compaction)
  - explore approaches to avoid ‘false sense of mastery’