Predicting Change Impact from Logical Models

**Problems**
- Prevailing models (e.g. UML) may be insufficient for predicting change impact because they:
  - Only capture syntactic dependencies and not semantic dependencies
  - Do not include concept of design rules [Baldwin et al. 00] (stable design decisions)
  - Data mining of co-change patterns in source control repository (e.g. [Ying et al. 04], [Zimmermann et al. 04]) useful to:
    - Identify semantic/logical dependencies between components
    - Predict change impact scope
  - However, data-mining results may be inaccurate when:
    - Version history is limited
    - Architecture is refactored

**ACN-Based Approach**
- Transform prevailing design model (e.g. UML class diagram) into Augmented Constraint Network (ACN) [Cai et al. 05]
- ACN assumption relation more than syntactic dependencies but less than transitive closure [Huynh et al. 08]
- Predict change impact from ACN-based constraint graph
- Assign weights to elements based on graph structure
- Recommend highest weight elements for change impact

**Hybrid Approach**
- ACN-based approach alone may not be enough
- Use ACN-based approach to get initial candidates
- Adjust candidates’ weights based on confidence [Zimmermann et al. 04]
- Frequency that an element and change source are changed together in revision history

**Evaluation**
- Hadoop Core – Java framework for map-reduce distributed computing
- Young system – 3 years of development
- 15 minor releases, no major releases
- Looked at 300+ modification tasks
- Identify change source(s) for each modification task
- Ran both ACN-based and hybrid approaches
- Compared against actual solution recorded by bug tracking system
- Ran existing data-mining approach [Zimmermann et al. 04]
- Compare quality of predictions versus data-mining predictions
  - Precision = 
    - # recommended
    - # correct
  - Recall = 
    - # correct
    - solution size – change source size
  - \( F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \)

**Results – F1 Comparison**

<table>
<thead>
<tr>
<th></th>
<th>ACN</th>
<th>Mining</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.17</td>
<td>21.8</td>
<td>6.48</td>
<td>8.78</td>
</tr>
<tr>
<td>0.70</td>
<td>22.2</td>
<td>-0.70</td>
<td>6.48</td>
</tr>
<tr>
<td>0.02</td>
<td>27.8</td>
<td>1.53</td>
<td>6.48</td>
</tr>
<tr>
<td>0.64</td>
<td>22.4</td>
<td>8.55</td>
<td>6.48</td>
</tr>
<tr>
<td>3.44</td>
<td>25.0</td>
<td>3.57</td>
<td>6.48</td>
</tr>
<tr>
<td>-2.26</td>
<td>22.6</td>
<td>-1.24</td>
<td>6.48</td>
</tr>
<tr>
<td>6.67</td>
<td>15.0</td>
<td>8.33</td>
<td>6.48</td>
</tr>
<tr>
<td>2.74</td>
<td>26.3</td>
<td>2.34</td>
<td>6.48</td>
</tr>
<tr>
<td>-3.90</td>
<td>34.8</td>
<td>-1.99</td>
<td>6.48</td>
</tr>
<tr>
<td>-2.06</td>
<td>35.6</td>
<td>3.87</td>
<td>6.48</td>
</tr>
<tr>
<td>-0.87</td>
<td>17.7</td>
<td>0.14</td>
<td>6.48</td>
</tr>
<tr>
<td>-1.05</td>
<td>21.2</td>
<td>-0.03</td>
<td>6.48</td>
</tr>
<tr>
<td>-3.65</td>
<td>28.0</td>
<td>-1.26</td>
<td>6.48</td>
</tr>
</tbody>
</table>

**Conclusion**
- When system is young, with limited history
  - Both ACN-based and Hybrid approaches outperform data-mining approach
  - As system matures:
    - Purely ACN-based approach degrades
    - Hybrid approach still comparable to data-mining approach

**Future Work**
- Evaluate on addition software systems
- Compare against other impact scope prediction approaches
- Investigate methods for selecting top candidates in ACN-based approach to improve recall

**Diagram**

1. UML Class Diagram
2. Augmented Constraint Network
3. Constraint Graph

- MapSite_interface: (orig, other);
  - Room_interface = orig => Room_interface
  - EnchantedRoom_interface: (orig, other);
    - EnchantedRoom_impl: (orig, other);
  - Room_interface = orig => MapSite_interface = orig;
  - EnchantedRoom_interface = orig => Room_interface = orig;
  - EnchantedRoom_impl = orig = orig;
  - Room_interface, MapSite_interface;
  - Wall_interface, MapSite_interface;
  - (EnchantedRoom_interface, Room_interface);

- Formulate UML into ACN
- Compute Impact Scope
- From Constraint Graph
- Report elements with highest weights to be in impact scope.
- Assign weights based on:
  1. Number of modules containing both element and change source. Modules identified by algorithm from [Cai et al. 06].
  2. Distance of element from change source.
  3. Whether element is a design rule [Baldwin et al. 00] to change source.
  4. Whether element is in same module/package/namespace as change source.