Model Merging Tool
Shortcomings

MoDSE 2008
Athens

Stephen Barrett
Concordia University, Montreal
Goals of Research

- Investigate how well today’s MDE tools support collaborative engineering:
  - Comparison and merging of changed artifacts
  - Premise: Model merging should work at least as well as source code merging

- Gather “anti-patterns” of merge behavior:
  - Validate intuition
  - Set a standard of expectation
  - Identify roots of incorrect, surprising, and counter-intuitive behavior
## Tools for Survey

- Readily available
- Currently used in industry
- Differing approaches to merging
- IBM Rational Rose 7.0
- IBM Rational Software Architect 7.0
- Sparx Enterprise Architect 6.5
- CVS & Subversion, “controls”
Methodology: Merge Evaluation Cases

- Consist of one or more merge scripts
  - Description of simple ancestor model
  - Changes to derived left and right models
  - Merge action for each change
    - Change-action pairs
    - Specialized for tool under study
  - Expected outcome model
    - Observed outcome for tool under study
## Merge Evaluation Case: Sample

<table>
<thead>
<tr>
<th>Empty class A</th>
<th>Ancestor model (#1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left contributions (#2)</td>
</tr>
<tr>
<td></td>
<td>Add empty class B - reject_m</td>
</tr>
<tr>
<td>Empty class A</td>
<td>Expected outcome (1)</td>
</tr>
<tr>
<td></td>
<td>Left contributions (#2)</td>
</tr>
<tr>
<td></td>
<td>Add empty class B - reject_m</td>
</tr>
<tr>
<td>Empty class A</td>
<td>Expected outcome (2)</td>
</tr>
<tr>
<td></td>
<td>Left contributions (#2)</td>
</tr>
<tr>
<td></td>
<td>Add empty class B - accept_m, accept_v</td>
</tr>
<tr>
<td>Empty class A</td>
<td>Expected outcome (3)</td>
</tr>
</tbody>
</table>
# Methodology: Change Merge Actions

<table>
<thead>
<tr>
<th>Operation</th>
<th>Arguments</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>model, view</td>
<td>Accept specified component(s) of change.</td>
<td>Not in EA</td>
</tr>
<tr>
<td>reject</td>
<td>model, view</td>
<td>Reject specified component(s) of change.</td>
<td>Not in EA, Accept #1 in Rose</td>
</tr>
<tr>
<td>ignore</td>
<td></td>
<td>Do nothing for change.</td>
<td>Not in EA</td>
</tr>
<tr>
<td>sync</td>
<td>model</td>
<td>Synchronize this model with specified model.</td>
<td>EA only</td>
</tr>
</tbody>
</table>

*1-April-2008  
*Model Merging Tool Shortcomings*
Input: Add Same Class (RSA)

What's the outcome model? How about the view?
Actions: Add Same Class (RSA - left, model & view)

add class B: accept (m, v)
Actions: Add Same Class (RSA - right, model & view)

add class B: $\text{accept}(m, v)$
Outcome: Add Same Class (RSA)

Tool ID over class name?
add class B: $\text{sync} (A)$

Actions: Add Same Class
Outcome: Add Same Class
Input: Add Same Class
Actions: Add Same Class (Rose - left, model)

add class B: `accept(m)`
Actions: Add Same Class (Rose -right, view)

add class B: accept (v)
Outcome: Add Same Class (Rose)

But which B is it?
Summary: Add Same Class

- Model element identity problem
  - Element duplication (RSA, EA)
  - Element elimination (Rose)
  - Inability to match model elements
Input: Add Different Classes (RSA)

ancestor (#1)

left (#2)

right (#3)

Outcome model?
Add class L: $\text{accept}(m, v)$
Actions: Add Different Classes
(RSA - right, model & view)

add class R: accept (m)
ignore (v)
Outcome: Add Different Classes (RSA)
Summary: Add Different Classes

- Model/view obfuscation
  - Element hiding (RSA)
  - View information confounds understanding
  - View clutter increases merge errors
rename AL: \textit{sync} (A)

rename AR: \textit{sync} (A)
Actions: Rename Class (RSA - left & right, model)

rename AL: accept(m)  rename AR: ignore(m)  “rejection by acceptance”  ‘acceptance by rejection’
rename AL: accept(m)  rename AR: accept(v)
Summary: Rename Class

- **Raw merging**
  - Last change wins (EA)

- **Model/view obfuscation**
  - Acceptance by rejection (RSA)
  - Element masking (Rose)
  - Database view-update similarities
Possible Applications of Catalog

- Leverage in order to:
  - Establish model relationships (i.e., match)
  - Set expectation standard
  - Benchmark tools

- Apply in model matching plug-in
  - Strategy for matching engine
    - EMF Comparison Framework (Brun, et al)
    - Epsilon Merging Language (Kolovos, et al)
Thank You

Questions?