Searching and Matching Software Components with Multiple Interfaces

Luis Iribarne\textsuperscript{1} and Antonio Vallecillo\textsuperscript{2}

\textsuperscript{1}University of Almería, Spain.
\textsuperscript{2}University of Málaga, Spain.

CBD Workshop @ TOOLS Europe 2000

June 5, 2000
• CBSE is moving organizations from applications development to applications assembly.

• The development effort now becomes one of gradual discovery about the components, their capabilities, their internal assumptions, and their incompatibilities when used in concert.

• Software component search and retrieval have now become crucial activities for building applications.

• Abstract specifications vs. Concrete specifications.
Traditional Approaches

- Components only offer 1 interface with their ‘supported’ methods.
- No ‘required’ services are taken into account.

Our approach tries to address both issues
Component Interfaces

- $C = \{ R, \overline{R} \}$
  - $\mathcal{R} = \{ R_1, ..., R_n \}$ the supported operations
  - $\overline{\mathcal{R}} = \{ \overline{R}_1, ..., \overline{R}_m \}$ the required operations

Component Composition

- $C_3 = C_1 \mid C_2$ is a new component, given by:
  \[
  \mathcal{R}_3 = \begin{cases} 
  \mathcal{R}_1 \cup \mathcal{R}_2 & \text{iff } \mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset \\
  \text{undefined} & \text{iff } \mathcal{R}_1 \cap \mathcal{R}_2 \neq \emptyset
  \end{cases}
  \]
  \[
  \mathcal{R}_3 = \mathcal{R}_1 \cup \mathcal{R}_2 - \{ \mathcal{R}_1 \cup \mathcal{R}_2 \} 
  \]
Interface Operations

- **Inclusion**: \( \{ R_1^1, ..., R_s^i \} \subseteq \{ R_2^1, ..., R_t^i \} \)

- **Intersection**: \( \{ R_1^1, ..., R_s^i \} \cap \{ R_2^1, ..., R_t^i \} \)

- **Hiding**: \( C_1 - \{ R \} = \{ R_1 - \mathcal{R}, \overline{\mathcal{R}}_1 \} \)

- **Replaceability**: \( C_1 \sqsubseteq C_2 \) iff \( (C_1.\mathcal{R}_1 \subseteq C_2.\mathcal{R}_2) \land (C_1.\overline{\mathcal{R}}_1 \supseteq C_2.\overline{\mathcal{R}}_2) \)

- **Equivalence**: \( C_1 \equiv C_2 \) iff \( C_1 \sqsubseteq C_2 \land C_2 \sqsubseteq C_1 \)

All operations defined on top of the traditional “\( \sqsubseteq \)” operator (syntactic, protocol, or semantic substitutability)
Building Apps with COTS components

0) \( A = A_1 \mid \ldots \mid A_n \) and then \( A = \{A.R, A.\overline{R}\} \)

1) **Candidates** selection

2) **Configurations** generation

3) **Closing** of configurations
Concepts

- **Candidates** (from a repository \( \mathcal{B} \)):
  \[
  C_{\mathcal{B}}(A) = \{ B \in \mathcal{B} \mid A.\mathcal{R} \cap B.\mathcal{R} \neq \emptyset \}
  \]

- **Configurations**: 
  - A set of candidate components \( C_1, \ldots, C_n \) with no service gaps or service overlaps.

- **Closure**: 
  - A transitive closure of a configuration \( C \) wrt \( C_{\mathcal{B}}(A) \)
function configs(i, Sol, S)
    /* 1 ≤ i ≤ k is the level, Sol the configuration being built */
    if i ≤ k then
        // case 1: try to include $C_i$ or part of it in $Sol$
        if $C_i \cap Sol \neq \emptyset$ then // $C_i$ or part of it can be included
            Sol := Sol ∪ {$C_i - (Sol \cap C_i)$};
        endif;
        if $A \subseteq Sol$ then // if Sol is a valid configuration...
            S := S ∪ {Sol} // ...we include it in $S$
        else // but if we still have service gaps...
            configs(i + 1, Sol, S) // ...we keep on searching
            endif;
        Sol := Sol - {$C_i - (Sol \cap C_i)$}
        endif;
        // case 2: try a configuration without $C_i$
        configs(i + 1, Sol)
    endif
endfunction
A simple example

\[ E = \{\text{Calculator (CAL), Calendar (CIO), Agenda (AG), Meeting Scheduler (MS)}\} \]
Abstract component specs vs. concrete candidate component specs

<table>
<thead>
<tr>
<th>$E$ Architecture</th>
<th>$C_B(E)$: Set of candidate components</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Abstract specifications)</td>
<td>(Concrete specifications)</td>
</tr>
<tr>
<td>$CIO = {R_{CIO}}$</td>
<td>$C_1 = {R_{CIO}}$</td>
</tr>
<tr>
<td>$CAL = {R_{CAL}}$</td>
<td>$C_2 = {R_{CAL}}$</td>
</tr>
<tr>
<td>$AG = {R_{AG}, R_{LIS}, \overline{R}<em>{CAL}, \overline{R}</em>{CIO}}$</td>
<td>$C_3 = {R_{AG}, R_{CIO}, \overline{R}_{CAL}}$</td>
</tr>
<tr>
<td>$MS = {R_{MS}, \overline{R}<em>{AG}, \overline{R}</em>{CAL}, \overline{R}_{CIO}}$</td>
<td>$C_4 = {R_{LIS}}$</td>
</tr>
<tr>
<td>$MS = {R_{MS}, \overline{R}<em>{AG}, \overline{R}</em>{CAL}, \overline{R}_{CIO}}$</td>
<td>$C_5 = {R_{MS}, R_{AG}, \overline{R}_{CIO}}$</td>
</tr>
<tr>
<td>$C_6 = {R_{CAL}, R_{LIS}, \overline{R}_{P}}$</td>
<td>$C_6 = {R_{CAL}, R_{LIS}, \overline{R}_{P}}$</td>
</tr>
</tbody>
</table>
## Results of the configs() algorithm

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td>R_AG</td>
<td>R_LIS</td>
<td>R_MS</td>
<td></td>
<td>C_1, C_2, C_3−{R_CIO}, C_4, C_5−{R_AG}</td>
</tr>
<tr>
<td>2</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td>R_AG</td>
<td>R_LIS</td>
<td>R_MS</td>
<td></td>
<td>Same as above</td>
</tr>
<tr>
<td>3</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td>R_AG</td>
<td>R_LIS</td>
<td></td>
<td></td>
<td>∅− R_MS missing (gap)</td>
</tr>
<tr>
<td>4</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td>R_AG</td>
<td>R_LIS</td>
<td></td>
<td></td>
<td>∅− R_MS missing (gap)</td>
</tr>
<tr>
<td>5</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td>R_AG</td>
<td></td>
<td>R_MS</td>
<td>R_CAL</td>
<td>C_1, C_2, C_3−{R_CIO}, C_5−{R_AG}, C_6−{R_CAL}</td>
</tr>
<tr>
<td>6</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td>R_AG</td>
<td></td>
<td></td>
<td></td>
<td>∅− R_LIS missing (gap)</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td></td>
<td>R_LIS</td>
<td>R_MS</td>
<td>R_AG</td>
<td>C_1, C_2, C_4, C_5</td>
</tr>
<tr>
<td>13</td>
<td>R_CIO</td>
<td>R_CAL</td>
<td></td>
<td></td>
<td>R_MS</td>
<td>R_AG</td>
<td>C_1, C_2, C_5, C_6−{R_CAL}</td>
</tr>
<tr>
<td>17</td>
<td>R_CIO</td>
<td></td>
<td>R_AG</td>
<td>R_LIS</td>
<td>R_MS</td>
<td></td>
<td>C_1, C_3−{R_CIO}, C_5−{R_AG}, C_6</td>
</tr>
<tr>
<td>21</td>
<td>R_CIO</td>
<td></td>
<td>R_AG</td>
<td></td>
<td>R_MS</td>
<td></td>
<td>C_1, C_3−{R_CIO}, C_4, C_5−{R_AG}, C_6−{R_LIS}</td>
</tr>
<tr>
<td>25</td>
<td>R_CIO</td>
<td></td>
<td></td>
<td>R_LIS</td>
<td>R_MS</td>
<td>R_AG</td>
<td>C_1, C_4, C_5, C_6−{R_LIS}</td>
</tr>
<tr>
<td>29</td>
<td>R_CIO</td>
<td></td>
<td></td>
<td></td>
<td>R_MS</td>
<td>R_AG</td>
<td>C_1, C_5, C_6</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>R_CAL</td>
<td>R_CIO, R_AG</td>
<td>R_LIS</td>
<td>R_MS</td>
<td></td>
<td>C_2, C_3, C_4, C_5−{R_AG}</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>R_CAL</td>
<td>R_CIO, R_AG</td>
<td></td>
<td>R_MS</td>
<td>R_LIS</td>
<td>C_2, C_3, C_5−{R_AG}, C_6−{R_CAL}</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td>R_CIO, R_AG</td>
<td>R_LIS</td>
<td>R_MS</td>
<td>R_CAL</td>
<td>C_3, C_4, C_5−{R_AG}, C_6−{R_LIS}</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td>R_CIO, R_AG</td>
<td></td>
<td>R_MS</td>
<td>R_CAL</td>
<td>C_3, C_5−{R_AG}, C_6</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>∅− R_CIO, R_AG, R_MS missing (gap)</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>∅− All services missing (gap)</td>
</tr>
</tbody>
</table>
Future Issues

- Metrics and heuristics for configurations
- Respecting the Application’s internal structure
- Using this approach with existing ADLs