WS-Binder: a Framework to enable Dynamic Binding of Composite Web Services

Authors: M. Di Penta, R. Esposito, M.L. Villani, RCOST – University of Sannio

R. Codato, M. Colombo and E. Di Nitto CEFRIEL

Context

- Allow for **dynamic binding in web service compositions**
  - Services can be discovered
    - before execution and stored in lists
    - at runtime
  - Bindings may vary
    - for different users
      - selection based on user preferences
      - customized global QoS guarantees (SLA)
    - over time
      - service availability may change
      - new services published
Features: Binding Types

- Two types of dynamic binding provided
  - Run-time local binding
    - What: determine each binding when needed
    - Why: to satisfy pre-defined local preferences
    - When: performed during process execution
    - How: by using context information
  - Pre-execution time global binding & re-binding
    - What: determine all bindings at once
    - Why: to satisfy pre-defined global QoS preferences
    - When: performed right before process execution (i.e., as latest as possible)
    - How: by using statistics
    - Bindings may change during execution if needed

Example

- Cost < 5$
- Resp Time < 2 min
- Maximize availability
Service Interface Mapping

- Different operation signatures

- Use of semantic facets
  - Input/Output parameters refer to ontological concepts
  - Mapping performed by adapters
    - A basic adapter looks for concepts match
    - Ad-hoc adapters may be introduced to enhance mapping by using concept relationships of the underlying ontology

Architecture

- Design time
  - Abstract services (WSDL)
  - BPEL process design
  - Preferences

- Deployment time
  - Process wrapper
  - Proxy services
  - Service adapters

- Execution time
  - Wrapper
    - Calls Binder
    - Calls engine
Pre-Execution Time binding: set up

Service Discovery
Continue the execution

- Core
- Configurator
- Monitoring
- Selector
- Monitoring Data

BPEL Execution Engine

- AS1 Proxy
- S1A
- S1B
- S1C
- AS2 Proxy
- S2A
- S2B

Concrete Services

Selection algorithms

- Local dynamic binding approach
  - Selection according to algorithms referred to in the preferences
    ```xml
    <selectionPreferences name="" description="">
      <preference>
        <key>implementation</key>
        <value>LocalSelection</value>
      </preference>
      <preference>
        <key>algorithm</key>
        <value>singleQualityRanking</value>
      </preference>
      <preference>
        <key>quality</key>
        <value>reliability</value>
      </preference>
    </selectionPreferences>
  ```
- Pre-execution global binding
  - Estimate the QoS of concretizations
  - Use genetic algorithms (GA) to find the optimal solution
- Re-binding
  - Determine the re-binding slice
  - Apply the global binding approach to the slice
Workflow QoS estimation

- QoS Aggregation Functions
  - QoS attributes currently considered:
    - Cost
    - Response time
    - Availability
  - Formulae

<table>
<thead>
<tr>
<th>QoS Attr.</th>
<th>Sequence</th>
<th>Switch</th>
<th>Fork</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (T)</td>
<td>$\sum_{t_i} T(t_i)$</td>
<td>$\sum_{p_{i,t}} T(t_i)$</td>
<td>$\max{T(t_i)_{i \in {1, \ldots, p}}}$</td>
<td>$k \cdot T(t)$</td>
</tr>
<tr>
<td>Cost (C)</td>
<td>$\sum_{t_i} C(t_i)$</td>
<td>$\sum_{p_{i,t}} C(t_i)$</td>
<td>$\sum_{t_i} C(t_i)$</td>
<td>$k \cdot C(t)$</td>
</tr>
<tr>
<td>Availability (A)</td>
<td>$\prod_{t_i} A(t_i)$</td>
<td>$\prod_{p_{i,t}} A(t_i)$</td>
<td>$\prod_{t_i} A(t_i)$</td>
<td>$A(t)^b$</td>
</tr>
<tr>
<td>Reliability (R)</td>
<td>$\prod_{t_i} R(t_i)$</td>
<td>$\prod_{p_{i,t}} R(t_i)$</td>
<td>$\prod_{t_i} R(t_i)$</td>
<td>$R(t)^b$</td>
</tr>
<tr>
<td>Custom Attr. (F)</td>
<td>$f_{\text{sequence}}(T(t_i), A(t_i), R(t_i))$</td>
<td>$f_{\text{switch}}(p_{i,t}, T(t_i))$, $f_{\text{fork}}(p_{i,t}, T(t_i))$</td>
<td>$f_{\text{fork}}(p_{i,t}, T(t_i))$</td>
<td>$I_{\text{loop}}(k, F(t))$</td>
</tr>
</tbody>
</table>

Finding a solution using Genetic Algorithm

- Representation:
  - Array chromosome
  - A slot for each abstract service invoked in the process
  - Value indicates the binding
  - $S_1, \ldots, S_N$ abstract services
  - $CS_{1}, \ldots, CS_k$ concrete services for $S_i$
**GA Operators**

**Crossover:**

- CS11, CS22, CS33, CS44
- CS11, CS24, CS33, CS44
- CS11, CS22, CS34, CS44

**Mutation:**

- CS11, CS22, CS33, CS44
- CS11, CS22, CS31, CS44

**Configuration example**

- Selection: Roulette wheel
- $p_{cross} = 0.7$
- $p_{mut} = 0.1$
- GA type: Simple with elitism of 2 (best) individuals
- Population size: 100

---

**Re-binding**

- **When:**
  - $QoS_{actual} - QoS_{estimated} > \text{threshold}$
  
- **Due to one of the following events**
  1. Difference between estimated and actual number of loop iterations
  2. Choice of the path to be followed in a switch
  3. Difference between the estimated and actual QoS of some executed services
  4. A service is not available

- Triggered at specified breakpoints, e.g., while, switch and invoke nodes
Determining the re-binding slice

Evaluation and future work

- Being evaluated from the industrial partners of the SeCSE project
  - Telecommunications and Automotive domains
  - We are collecting feedbacks
- Framework being enhanced with:
  - User-defined QoS attributes
    - Language to specify aggregation formulae
- Focus on
  - Extension of BPEL to support the definition of rules constraining autonomic behavior of the composition
  - Negotiation of SLAs
  - Support for transactions
Related Work

• Various languages and environments for dynamic composition
  • Examples: SELFSERV, E-FLOW, MAIS, METEOR-S...

• Advantages of our approach
  • Uses/Extends the BPEL de-facto standard (as MAIS and METEOR-S)
  • Supports various kinds of bindings (pre-execution and runtime (re)binding) ... and binding policies (QoS, context-dependent, ...