A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment

Quan Z. Sheng, Boualem Benatallah, Eileen Oi-Yan Mak
University of New South Wales, Sydney, Australia

Marlon Dumas
Queensland University of Technology, Brisbane, Australia
SELF-SERV provides a platform for rapid composition of Web services, where Web services are declaratively composed, and the resulting composite services are executed in a peer-to-peer and dynamic environment.

SELF-SERV distinguishes three types of services: elementary services, composite services, and service communities. Composite services are described through statecharts. Service communities perform dynamic provider selection through selection policies.
2. Service Creation

The Service Builder provides a statechart editor for describing composite service operations. It also offers search and browse facilities to locate operations from existing services and import them into composite services. It provides an editor for describing service operations, loading and configuring SELF-SERV built-in classes (e.g., linking an operation to a scoring service of a community).
3. Service Deployment

The deployment of a composite service requires the generation and deployment of routing tables (i.e., precondition/postprocessing tables of each state of a composite service statechart). The routing tables (XML documents) are then uploaded into the hosts of the corresponding component services, together with the wrapper and coordinator classes.
4. Service Discovery Engine

Services are published in a UDDI registry and located through the Service Discovery Engine.
5. Service Execution

SELF-SERV adopts an orchestration model based on peer-to-peer interactions between software components hosted by the providers participating in the composition. The orchestration of a composition service execution involves two types of messages: control flow notifications and service invocations/completions.
6. Performance Evaluation

We conducted several experiments to study the performance of SELF-SERV using a cluster of PCs.

The results show that the deployment speed is acceptable, and that the peer-to-peer approach to composite service execution leads to increased throughput through better workload distribution.

<table>
<thead>
<tr>
<th>No. of component service</th>
<th>15</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment cost (second)</td>
<td>8.9</td>
<td>10.1</td>
<td>13.1</td>
<td>15.9</td>
<td>19.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of component service</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment cost (second)</td>
<td>21.8</td>
<td>25.9</td>
<td>30.2</td>
<td>32.9</td>
<td>34.1</td>
</tr>
</tbody>
</table>

Deployment cost of composite services.

![Number of received messages](chart1.png)

Workload allocation.

![Composite service execution time](chart2.png)

Execution time with different message sizes.