An Intelligent Broker Approach to Semantics-based Service Composition

Yufeng Zhang
National Lab. for Parallel and Distributed Processing
Department of Computing Science
National Univ. of Defense Technology, Changsha, China
Email: yufengzhang@nudt.edu.cn

Hong Zhu
Department of Computing and Communication Technology
Oxford Brookes University
Oxford OX33 1HX, UK
Email: hzhu@brookes.ac.uk
Approaches to service composition

• Workflow:
  – Workflow definition + workflow engines:
    • e.g. orchestration and choreography; BPEL4WS; OWL-S

• Planning:
  – E.g. OWLS-Xplan, SHOP2, etc.

• Broker:
  – Sycara et al. [2003, 2004] :
    • high flexibility
    • wide applicability
Current state of service brokers

The functions of service brokers:

– **Interpreting** the semantics of service queries and the registered capabilities of service providers;

– **Searching** for the service providers that matches a requester's query and sometime selecting the one with best track record of quality of services;

– **Invoking** the selected service provider on the requester's behalf and interacting with the provider if necessary to fulfil the query;

– **Returning** query results to the requester.
The goal

To enhance the power of service brokers with the capability of:

- **decomposing** requested services into a number of subtasks,

- **searching** for the best fit services for each subtask,

- **composing** and coordinating these services in execution.
I-Broker: Architecture

Service Broker

Knowledge base

Controller

Communicator

Planner

Search Engine

Coordinator

Domain Ontology

Matchmaker

UDDI Registry

Service S1

Service S2

Service Sn
I-Broker: Control Process

1. User: Request a service
2. Generate service composition plan
   - If Plan exists
     - Abstract Plan
       - Subtask 1
       - ... Subtask n
     - Convert each subtask into required service capability
     - Search for services
   - If No plan
     - No candidate service
     - Matchmaker
     - Candidate service exists
3. Service result
   - Service succeeds & no more subtask
   - Service fails
   - Service successes & more subtask to do
4. Report success & return results
5. Invoke service
6. Invoking message
7. Construct service invocation message
8. Concrete service composition plan
9. Select a service for each subtask
10. A list of candidate services
Knowledge-base

• It contains codified knowledge on how a task can be fulfilled by a number of subtasks.
• Each type of tasks is defined by a set of parameters
  – descriptive parameters
    • describes the functionality of the task, such as the activity of the task, the execution environment of the task, and so on.
  – functional parameters
    • gives the data to be transformed by the task, including input and output data.
• The values of parameters are concepts defined in the ontology of the application domain.
• The knowledge is represented in the form of rules:
  \[ T(p_1, \ldots, p_n) \Rightarrow T'_1(p_{1,1}, \ldots, p_{1,n_1}); \ldots; T'_k(p_{k,1}, \ldots, p_{k,n_k}) \]
Prototype Implementation

I-Broker

Knowledge-Base Manager

Ontology Manager

Manages

Knowledge-base Frame

Broker Skeleton

Manages

Manages

Manages

Invokes

Uses

Domain Ontology in OWL

Service

Service

Refers to

Refers to

Written in OWL-S Profile

Refer to

Uses

Refer to

Registered as a semantic web service

Invokes to search services

Invokes to search services

Inferred by

OWL-S/UDDI Matchmaker

OWL-S Language

Semantics Web Service Infrastructure

7/2011

COMPSAC 2011
Experimental Evaluation

• Goal: The services that test broker searches for
  – to evaluate the scalability of the proposed approach

• Design of the experiments
  – Three types of objects in the experiments:
    • Services:
      – registered to the semantic WS registry.
      – capabilities represented in the form of service profiles.
    • Rules in knowledge-base:
      – represented in the form of XML files
      – stored locally within the broker as the knowledge-bases.
    • Service requests:
      – represented in the format of XML using the ontology.

The service requests submitted to the brokers
About the workflow in the application domain and the usages of specific services
Experiment method

• Data mutation technique: [Shan&Zhu 2009]
  - Seeds of test data
  - Mutation operators

Seeds of Services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASCAT Test Translator</td>
<td>Generate test cases from algebraic specifications</td>
</tr>
<tr>
<td>Test Executor</td>
<td>Translate test cases from CASCAT format into Test Executor format</td>
</tr>
<tr>
<td>Klee</td>
<td>Execute tests for a numeric calculator WS</td>
</tr>
<tr>
<td>Magic</td>
<td>Symbolic execution of C code</td>
</tr>
<tr>
<td>XML Comparator</td>
<td>Check component’s conformance to specification</td>
</tr>
<tr>
<td>Java NCSS</td>
<td>Compare XML files</td>
</tr>
<tr>
<td>Findbugs</td>
<td>Metrics for Java program</td>
</tr>
<tr>
<td>PMD</td>
<td>Find bugs in Java program by static analysis</td>
</tr>
<tr>
<td>WSDL Test Gen</td>
<td>Find potential bugs in Java by static analysis</td>
</tr>
<tr>
<td>WS Test Executor</td>
<td>Generate test cases from WSDL</td>
</tr>
<tr>
<td></td>
<td>Execute tests generated by WSDL Test Gen</td>
</tr>
</tbody>
</table>
Mutation operators

Given an ontology.
Let $x$ be any of the parameters in service profiles.

- **RxF:**
  - Replace the $x$ parameter in the profile, which is a class in the ontology, by its father class in the ontology;

- **RxS:**
  - Replace the $x$ parameter in the profile by one of its subclasses in the ontology;

- **RxB:**
  - Replace the $x$ parameter in the profile by one of its brother classes in the ontology;

- **RxN:**
  - Replace the $x$ parameter in the profile by a class in the ontology that has no relation to the parameter.

<table>
<thead>
<tr>
<th></th>
<th>seeds</th>
<th>Mutants</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>11</td>
<td>460</td>
<td>471</td>
</tr>
<tr>
<td>Rule</td>
<td>40</td>
<td>2049</td>
<td>2089</td>
</tr>
</tbody>
</table>
Scalability w.r.t. the number of services

• Vary the registry size from 20 to 471
• Execute the broker repeatedly for 30 times

\[ y = 0.0004x^2 + 6.1862x + 207.16 \]
\[ R^2 = 0.9948 \]

almost a linear function
Scalability w.r.t. the size of KB

- Vary the knowledge size from 100 to 2089
- execute the broker on each service request for 20 times

\[ y = 4E-05x^2 + 0.0609x + 20.044 \]
\[ R^2 = 0.9873 \]

A quadratic polynomial function

The number of task plan templates
Scalability w.r.t. to task complexity

• vary the service requests (5 classes of requests)
• execute the broker on each service request for 30 times

\[ y = 10.601x^2 + 3168.2x + 119.49 \]
\[ R^2 = 0.9999 \]

A quadratic polynomial function
## Comparison with Related Works

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Planning</th>
<th>I-Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow encoded in executable form</td>
<td>Workflow knowledge is represented in the form of task decomposition rules</td>
<td>Workflow knowledge is represented in the form of task decomposition rules</td>
</tr>
<tr>
<td></td>
<td>The knowledge of workflow is simply converted from OWL-S</td>
<td>Workflow knowledge is treated as domain knowledge</td>
</tr>
<tr>
<td>Workflow knowledge is used statically</td>
<td>Plans are mostly generated by a brutal force of inference</td>
<td>Use workflow to generate service plan dynamically</td>
</tr>
<tr>
<td></td>
<td>• Use the whole service registry as the search space</td>
<td>Use rules as a means of reduce search space:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• scalable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• efficient</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knowledge-base and ontology are open, and composable</td>
</tr>
</tbody>
</table>

*COMPSAC 2011*
Future work

• Develop knowledge-based manager
  – Enable user to write rules in a language of high level of abstraction rather than directly in XML
  – Support populating, updating, and testing the knowledge-base

• Embed service monitoring functionality in the service broker
  – QoS directed service selection

• Replace Semantic Web Service with a better semantic inference mechanism