Enterprise Interoperability with SOA: a Survey of Service Composition Approaches

Rodrigo Mantovaneli Pessoa
Eduardo Gonçalves Silva
Marten J. van Sinderen
Dick A. C. Quartel
Luís Ferreira Pires

September 2008
Outline

• Motivation
• Comparison framework
• Composition approaches overview:
  – METEOR
  – SODIUM
  – WSMF
  – MoSCoE
  – SeCSE
• Conclusions
• Questions and comments
No Enterprise Is an Island
The World is Always Changing…

- Law & Regulation
- Technology
- Costumer Needs
- Partner Requirements
- Sourcing
The need for integration and flexibility

• Service Composition
  – Claims to facilitate the construction of flexible and loosely coupled applications, and therefore is seen as an enabling factor for enterprise interoperability.
  – A landscape of languages and techniques for services composition has emerged and is continuously being enriched.
  – However, little effort has been dedicated to systematically evaluate the capabilities and limitations of these languages and techniques.
Comparison Framework

• Based on detailed analysis of the different phases of the service composition life-cycle.

• Allows studying how different approaches deal with the service composition life-cycle and provides basic guidelines for their analysis, evaluation and comparison.
## Comparison Framework

<table>
<thead>
<tr>
<th>Life-cycle Phase</th>
<th>Evaluation Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Discovery</td>
<td>Service description</td>
</tr>
<tr>
<td></td>
<td>Service matching and selection</td>
</tr>
<tr>
<td>Process model creation</td>
<td>Behaviour specification</td>
</tr>
<tr>
<td></td>
<td>Information specification</td>
</tr>
<tr>
<td></td>
<td>Level of automation</td>
</tr>
<tr>
<td></td>
<td>Composition time</td>
</tr>
<tr>
<td></td>
<td>Coordination distribution</td>
</tr>
<tr>
<td>Process model verification</td>
<td>Composition correctness</td>
</tr>
<tr>
<td>Execution</td>
<td>Service binding</td>
</tr>
</tbody>
</table>
Service Composition Approaches

• The compared approaches:
  – METEOR-S
  – SODIUM
  – WSMF
  – MoSCoE
  – SeCSE
METEOR

- Apply Semantic Web techniques to enable service composition
- The focus is not in developing ontologies for representing functionality/preconditions/effects but to use such ontologies for semantic annotations.
- Template based process generation
- The process templates act as configurable modules that capture the semantic requirements of the participating activities, control flow, data flow and intermediate transformations
WSDL with Semantic Annotations

- WSDL constructs are linked to ontological concepts to explicate the semantics of:
  - Input and Output Message Parts
  - Operations
- Additional tags to represent:
  - Pre- and post-conditions
  - Effects
Template Construction

• Activities are linked by control flow constructs

• Process designer can either specify a semantic description of the service or a binding to a known service

• Semantic descriptions allow to automatically discover and bind services to the process
Summary of Steps in Discovery

1. Services selection based on the functional requirements
   - Using operation → ontology mapping

2. Ranking based on semantic similarity based on input/output semantics of candidate services and requirement template
   - Using message part → ontology mapping

3. Optional step includes semantic similarity based on semantics of preconditions/effects of the candidate services and requirement template
   - Using precondition and effect tags
METEOR Perspective

- Process Template captures the semantics of activities in a process and can be generated into an executable process

- Utilizes Data, Functional, QoS Semantics during template construction and service discovery

- Semantic discovery is better in finding appropriate services in comparison with keyword/taxonomy
SODIUM

• The project addresses the need for standards-based, integration of heterogeneous services
  – Web services
  – P2p services
  – Grid services
• Supports the visual service composition, analysis, execution, management and monitoring of services, in an unified way
SODIUM Generic Service Model (GeSMO)

- **Motivation:** to form a common conceptual basis for the specifications of the SODIUM languages and tools

- **Main Characteristics:**
  - Based on the conceptual analysis of the state-of-the-art in the areas of Web, Grid, and P2P services
  - Conceptualization of services from different points of view
  - Core layer describing common service concepts
  - Extension layers describing distinct service concepts
Unified Service Composition Language (USCL)

- **Motivation:** To provide an executable language for the service compositions

- **Main Characteristics:**
  - Defined with the use of XML Schema
  - Support of most common workflow patterns
  - Support for web, p2p, and grid service types
Visual Service Composition Language (VSCL)

• **Motivation:** to establish a visual language that will facilitate users in developing heterogeneous compositions

• **Main Characteristics:**
  – Standards-based (extends UML 2.0)
  – Support for web, p2p, and grid service types
  – Explicit support of control and data flows
Visual Composition Language
Example

Diagram showing a process flow with nodes labeled 'Transformation\textsubscript{mapquest}', 'Transformation\textsubscript{map24}', and 'local authority directions'. The process starts with 'start street' and 'start city', and ends with 'end street' and 'end city'. The 'mapQuest' and 'map24' interfaces are involved in the transformation process, with 'rsrc0' and 'rsrc1' as endpoints. The 'determineBestRoute' function is also depicted, influencing the flow of the process.
SODIUM Perspective

• The SODIUM platform is especially useful for the grid community
  – Visual Composition Suite makes it possible to compose grid services in a graphical way
  – Monitor tool enables the visualization of the progress of execution in a state-full way.
WSMF

- Features:
  - A web service modeling ontology (WSMO)
  - An execution environment (WSMX)
  - A description language (WSML):
    - A language for the specification of ontologies and different aspects of services;
WSMF Top Notions

**Goals:** Objectives that a client wants to achieve by using Web Services

**Ontologies:** Provide the formally specified terminology of the information used by all other components

**Web Services:**
- *Semantic description of Web Services:*
- Capability (functional)
- Interfaces (usage)

**Mediators:** Connectors between components with mediation facilities for handling heterogeneities
WSMF Goals

• **Decoupling of request and service**
  - Requester formulates objective independent / without regard to services for resolution
  - ‘Intelligent’ mechanisms detect suitable services for solving the Goal
  - Allows re-use of Goals

• **Usage of Goals within Semantic Web Services**
  – A Requester, that is an agent (human or machine), defines a Goal to be resolved
  – Web Service Discovery detects suitable Web Services for solving the Goal automatically
  – Goal Resolution Management is realized in implementations
WSMF Mediators

• **Declarative Approach:**
  – Semantic description of resources
  – ‘Intelligent’ mechanisms that resolve mismatches

• **Four mediator types:**
  – *ooMediators* - mediators that resolve possible representation mismatches between ontologies.
  – *ggMediators* - mediators that link two goals.
  – *wgMediators* - mediators that link a goal to services.
  – *wwMediators* - mediators that link two services.
WSMF Capability Specification

• Non functional properties

• Pre- and Post-conditions
  – They define conditions over the input/output

• Assumptions
  – Conditions on the state of the world that has to hold before the Web Service can be executed and work correctly

• Effects
  – Conditions on the state of the world that hold after execution
Choreography: based on ASMs

Abstract State Machine Rules

- stateSignature
  - in a → wsdl.interfaceMessageReference ...
  - out b → wsdl.interfaceMessageReference ...

- transitionRules
  - If a then add(b)
  - ...

If message A is in the memory, then add message B to the memory from invocation of related operation.
WSMF Perspective

- Provides a conceptual model for Web Services modelling and related aspects
- Stress on solving the integration problem
  - Mediation as a key element
- All the way from conceptual modelling to usable implementation (WSML, WSMO, WSMX)
## Approaches Comparison

<table>
<thead>
<tr>
<th></th>
<th>METEOR-S</th>
<th>SODIUM</th>
<th>MoSCoE</th>
<th>SeCSE</th>
<th>WSMF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service description</strong></td>
<td>Semantically augmented with SAWSDL</td>
<td>UML models enriched with constraints and OMG’s QoS profile</td>
<td>Services are represented in OWL-S and WSDL</td>
<td>Services are represented in Faceted service specification</td>
<td>Capability described in terms of pre- and post-conditions, assumptions, and effects</td>
</tr>
<tr>
<td><strong>Service matching and selection</strong></td>
<td>three-phase matching algorithm based on semantic similarity</td>
<td>Based on USQL queries with behavioural constraints</td>
<td>Semantic reasoning optimized with non-functional properties</td>
<td>Two-phase matching algorithm based on text-searching functions</td>
<td>Keyword matching, lightweight semantic matching and QoS based</td>
</tr>
<tr>
<td><strong>Behaviour specification</strong></td>
<td>Based on process templates with BPEL-like syntax</td>
<td>A graphical composition language is used to define data and control flow</td>
<td>Based on UML state machine diagram and Symbolic Transition Systems</td>
<td>BPEL-like service composition creation based on abstract workflow definition</td>
<td>Based on abstract state machines, consisting of states and guarded transitions</td>
</tr>
<tr>
<td><strong>Information specification</strong></td>
<td>Model reference annotations and schema mapping annotations</td>
<td>Data objects used as internal data representation format and data transformations expressed in QVT</td>
<td>Inter-ontology mappings</td>
<td>Faceted service specification</td>
<td>Ontologies are used as internal data representation format and mediators are defined in case of data mismatch</td>
</tr>
<tr>
<td><strong>Level of automation</strong></td>
<td>Support for manual and semi-automatic</td>
<td>Semi-automatic support</td>
<td>Semi-automatic support</td>
<td>Semi-automatic support</td>
<td>Automatic</td>
</tr>
<tr>
<td><strong>Composition time</strong></td>
<td>Design-time and runtime</td>
<td>Design-time</td>
<td>Design-time</td>
<td>Design-time with binding rules for dynamic adaptation</td>
<td>Runtime</td>
</tr>
<tr>
<td><strong>coordination distribution</strong></td>
<td>Centralized</td>
<td>Centralized</td>
<td>Centralized</td>
<td>Centralized</td>
<td>Peer-2-peer</td>
</tr>
<tr>
<td><strong>Composition correctness</strong></td>
<td>State machine based verification of BPEL</td>
<td>No support for formal proof of correctness</td>
<td>Symbolic transitions system based</td>
<td>No explicit support</td>
<td>No explicit support</td>
</tr>
<tr>
<td><strong>Service binding</strong></td>
<td>Static binding, deployment-time binding and dynamic binding</td>
<td>Design-time, compilation-time, deployment-time and Runtime</td>
<td>Static binding</td>
<td>Static and dynamic binding</td>
<td>Static and dynamic binding</td>
</tr>
</tbody>
</table>
Conclusions

• Our framework:
  – Developed following the phases of composite service life-cycle
  – Establishes a common set of criteria that provide basic guidelines for the evaluation process;
  – Enables a more comprehensive understanding of existing approaches by exposing their advantages and drawbacks.

• The compared approaches:
  – Focus on specific phases of the life-cycle, while neglecting others
  – widely differ in how they address the mentioned requirements

• Opportunities:
  – Combining the benefits of the different approaches
Questions and Comments