Context-Aware Autonomous Web Services in Software Product Lines

Germán H. Alférez¹ and Vicente Pelechano²

¹ Facultad de Ingeniería y Tecnología, Universidad de Montemorelos, Mexico
² Centro de Investigación en Métodos de Producción de Software (ProS), Universitat Politècnica de València, Spain

Presented at the 15th International Software Product Line Conference (SPLC).
Munich, Germany. August, 2011
Service-Oriented Architecture:

Improves the **agility** and **cost-effectiveness** of a company.

Web services are the most common realization of SOA:

- Run in heterogeneous and complex environments.
  - **Adaptation mechanisms:**
    - Impractical to assign manual reconfiguration tasks.
      - *Burden to IT staff & reaction to contextual events.*
    - Autonomic Computing: self-* mechanisms.
      - *Dynamic binding & adaptation policies.*

- In SOA, **reusability logic is divided into services.**
  - SOA does not promote prescribed reuse of Web services.
  - Variants among systems are difficult to capture explicitly using the notion of Web services.
Problem

Need for Autonomic Adaptation of Web services

Need for Systematic Reuse of Web Services
Our Approach

Need for Autonomic Adaptation of Web services

Need for Systematic Reuse of Web Services

Supporting method for: Designing & implementing context-aware autonomous Web services in systems families.

Supporting tool
Our Approach

Need for Autonomic Adaptation of Web services

Need for Systematic Reuse of Web Services

Supporting method for: Designing & implementing context-aware autonomous Web services in systems families.

Supporting tool

Autonomic Computing + SPL engineering + Models at runtime + Dynamic SPL (DSPL) engineering
Our Approach

Our method's basis:

- **Autonomic Computing**: Automate tasks for self-adapting Web service operations.

- **SPL Engineering**: Activation/deactivation of SPL features at runtime.

Web service operations characterized by **SPL features**

**Systematic Reuse**

Activation/deactivation of SPL features at runtime

**Autonomic reconfiguration of service compositions** depending on contextual changes
Our approach

Our method's basis (Cont.):

- **DSPL Engineering**: The architecture of a DSPL allows a flexible service recomposition.
  - When features are activated/deactivated
    - A DSPL architecture binds variation points at runtime

- **Models at Runtime**: The production capability is based on reusable models (core assets).
  - **Variability models**: Easy-to-understand and semantically rich *adaptation policies* for decision making.
Our Approach

Requirements:

1. Context: Any environmental information that can be used by a Web service at runtime.

2. Measure Instruments:

   • Monitor the context and get the measures for basic metrics of specific quality attributes.
   • Availability and time.
Our Approach

Requirements (Cont.):

3. Context Conditions:

• New context event → **Does it violate any context condition** (Service Level Agreement or contract)?
  – Contract is violated → **Reconfiguration** of the service composition.

4. Resolutions:

• If a **context condition** has been accomplished: **What are we going to do?**
• Express **adaptation policies** or **transitions** between different configurations of service compositions.
• \( R_c = \{(F, S)\} \mid F \in [FM] \land S \in \{\text{Active, Inactive}\} \)
Our Approach

Our method's SPL activities:

1. Domain Engineering Activity.
   - **Reusable models**: Production capability for service compositions.

2. Application Engineering Activity.
   - Supports the *derivation of specific service compositions from a product family*.
   - Autonomic recomposing Web services: **Model-based Reconfiguration Engine for Web services (MoRE-WS)**.
Our Approach

MoRE-WS:

→ MoRE-WS translates context changes into changes in the activation/deactivation of features.

Our Approach

Case Study:

A SPL for **mobile tourist planners** based on Web services:

- Lists the tourist attractions of a city.
- Recommends trips to those places depending on the **weather** and **current location**.
Our Approach

Domain Engineering Activity:

- Feature Model
- Composition Model
- Context Model
- Weaving Model

Systems Engineer Creates 1 Waves 1 Waves 1 Feature Model for Measure Instrum.
Our Approach

Domain Engineering Activity / Feature Model:

- Describes the dynamic system configurations and the variants of the system.
- Some features denote the initial system configuration, while other features represent potential variants.
Our Approach

Domain Engineering Activity / Composition Model:

- **Web services** and the **sequence flows** among them.
- **UML Activity diagram.**
Our Approach

Domain Engineering Activity / Composition Model:

- **Web services** and the **sequence flows** among them.
- **UML Activity diagram.**
Our Approach

Domain Engineering Activity / Composition Model:

- **Web services** and the **sequence flows** among them.
- UML Activity diagram.
Domain Engineering Activity / **Composition Model:**

- **Web services** and the **sequence flows** among them.
- **UML Activity diagram.**
### Domain Engineering Activity / Composition Model:

Mapping rules.

<table>
<thead>
<tr>
<th>Feature Model</th>
<th>Composition Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Elements</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>Root</td>
<td>Mobile Tourist Planner feature</td>
</tr>
<tr>
<td>Compound features (interior nodes)</td>
<td>Weather feature</td>
</tr>
<tr>
<td>Leaves (primitive features)</td>
<td>Global Weather feature</td>
</tr>
<tr>
<td>And (all subfeatures must be selected)</td>
<td>N/A&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>Alternative (only one subfeature can be selected)</td>
<td>Weather single choice</td>
</tr>
<tr>
<td>Or (one or more features can be selected)</td>
<td>Transportation multiple choice</td>
</tr>
<tr>
<td>Mandatory (features that are required)</td>
<td>Location Feature</td>
</tr>
<tr>
<td>Optional (features that are optional)</td>
<td>Billing feature</td>
</tr>
</tbody>
</table>

<sup>a</sup> Not applicable in the case study.
Our Approach

Domain Engineering Activity / Weaving Model:

- Define and capture relationships between features in the Feature Model and model elements of the Composition Model.
- One-to-many relationship.
Our Approach

Domain Engineering Activity / **Weaving Model**:

- **Define** and **capture relationships** between **features** in the **Feature Model** and **model elements** of the **Composition Model**.
- One-to-many relationship.
Domain Engineering Activity / Feature Model for Measure Instruments:

- Measure instruments in terms of features: e.g. Response time and execution time.
  - They can be systematically reused.
Domain Engineering Activity / Context Model:

- Ontology-based.
  - Formal analysis of the domain knowledge. Context reasoning using first-order logic.
Our Approach

Application Engineering Activity:

Domain Engineering Activity

- Feature Model
- Composition Model
- Context Model
- Weaving Model

Systems Engineer Creates

Initial Configuration

Run-time Configuration
Our Approach

Domain Engineering Activity
- System Engineer creates Feature Model
- Feature Model is woven to Composition Model and Context Model

Application Engineering Activity
- Selects Features to Initial Configuration
- Initial Configuration leads to Run-time Configuration
Our Approach

Current Configuration = 
{Mobile Tourist Planner, Location, Weather, Global Weather, Transportation, Bicycle, Bus}
Our Approach

Domain Engineering Activity

- Feature Model
- Composition Model
- Context Model
- Weaving Model

Systems Engineer

Application Engineering Activity

- Initial Configuration
  - Feature Model
  - Feature Model for Measure Instrum.

- Run-time Configuration

Selects Features 2
Selects Metrics 3
Creates 1
Our Approach

Current Configuration (Measure Instruments) = \{Availability, Response Time, Execution Time\}
Our Approach

Domain Engineering Activity

- Systems Engineer
  - Creates Feature Model
  -唐山 Weaving Model
  -唐山 Composition Model
  -唐山 Context Model

Application Engineering Activity

- Initial Configuration
  - Selects Features
  - Selects Metrics
  - Adjusts Context Conditions
  -唐山 Feature Model
  -唐山 Context Model

- Run-time Configuration
  -唐山 Feature Model for Measure Instrum.
  -唐山 Context Condition
Our Approach

GlobalWeather_Unavailable = (Global Weather, isAvailable, false)

Location_HiRespTime = (Location, responseTime, > 2,000 ms)
Our Approach

Domain Engineering Activity

- Feature Model
- Composition Model
- Context Model
- Weaving Model for Measure Instrum.

Application Engineering Activity

Initial Configuration

- Feature Model
- Context Model
- Feature Model for Measure Instrum.

Run-time Configuration

- Context Condition
- Resolution

Systems Engineer

Creates Feature Model

Selects Features

Selects Metrics

Adjusts Context Conditions

Adjusts Resolutions
Our Approach

\[ R_{\text{GlobalWeather_Unavailable}} = \{(\text{Mobile Tourist Planner, Active}), (\text{Location, Active}), (\text{Weather, Active}), (\text{Global Weather, Inactive}), (\text{Weather Forecast, Active}), (\text{Transportation, Active}), (\text{Bicycle, Active}), (\text{Bus, Active})\} \]
Our Approach

Domain Engineering Activity

- Systems Engineer
- Creates
- Feature Model
- Composition Model
- Context Model
- Weaving Model

Application Engineering Activity

- Initial Configuration
  - Selects Features
  - Selects Metrics
  - Adjusts Context Conditions
  - Adjusts Resolutions
  - Feature Model
  - Feature Model for Measure Instrum.
  - Context Model
  - Context Condition
  - Resolution
  - Monitors

- Run-time Configuration
  - Depends on
  - For
Our Approach

Application Engineering Activity / **Runtime Configuration**:

IBM's reference model for autonomic control loops (MAPE-K loop)
Our Approach

Application Engineering Activity / Runtime Configuration:

a. Monitor:

- Captures **basic metrics** of specific **quality attributes** from the context.
- **Monitor component** of **SALMon** (Ameller and Franch @ ICCBSS 2008).

![Diagram of Autonomic Manager with Monitor, Analyze, Plan, Execute, Knowledge, Managed System, Sensors, and Actuators]
Our Approach
Our Approach

Application Engineering Activity / Runtime Configuration:

b. Analyze:

(Global Weather, Inactive), (Weather Forecast, Active)
Our Approach

Application Engineering Activity / Runtime Configuration:

c. Plan:

Reconfiguration actions stated as $A\nabla$ and $A\Delta$.

Given $R_{context\ condition} \rightarrow \text{Reconfiguration Plan.}$ $R_{\text{globalWeather\_Unavailable}}$:

$A\nabla_{\text{GlobalWeather\_Unavailable}} = \{\text{Global Weather, WeatherDecisionTOGlobalWeather, GlobalWeatherTOTransportDecision}\}$

$A\Delta_{\text{GlobalWeather\_Unavailable}} = \{\text{Weather Forecast, WeatherDecisionTOWeatherForecast, WeatherForecastTOTransportDecision}\}$
d. Execute:

- Execution of the **Reconfiguration Plan**.
- Web services are created using the **Java API for XML Web Services (JAX-WS)** and deployed as **OSGi bundles in Swordfish**.
Our Approach

Domain Engineering Activity

- Feature Model
- Composition Model
- Context Model
- Weaving Model

Application Engineering Activity

- Initial Configuration
  - Feature Model
  - Feature Model for Measure Instrum.
  - Context Model
- Run-time Configuration
  - SALMon
  - MoRE-WS
  - Reconfiguration Plan
  - Swordfish

Systems Engineer

- Creates Feature Model
- Waves Feature Model
- Updates Context Model
- Updates Feature Model for Measure Instrum.
Our Approach

Application Engineering Activity / Runtime Configuration:

e. Knowledge:

- The SPARQL Protocol and RDF Query Language (SPARQL):
  - Data source to be queried: Ontology.
  - INSERT and ASK.

- The EMF Model Query framework (EMFMQ): To query the Feature Model and the Weaving Model.
Our Approach
Presented a method to design and implement context-aware autonomous Web services in system families.

- Autonomic Computing, SPL engineering, DSPL architecture, and models at runtime.
- Small case study using MoRE-WS prototype.
Evaluate our approach with respect to:

- Autonomic-level achievement.
- Scalability of model-handling technologies at runtime.

**Tool to validate reconfigurations** of service compositions at design time to prevent negative effects during execution.
Thanks!

Questions?

harveyalferez@um.edu.mx