### Call for Workshop Participation

**Service-Oriented Architectures and Software Product Lines - Putting Both Together**  
*(SOAPL 2008)*  
**Monday, 8 September 2008**

<table>
<thead>
<tr>
<th>Description</th>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Presentations</strong></td>
<td><strong>Attendees</strong></td>
</tr>
<tr>
<td>Krut, Robert &amp; Cohen, Sholom. <em>Workshop on Service-Oriented Architectures and Software Product Lines - Putting Both Together (SOAPL 2008)</em></td>
<td></td>
</tr>
</tbody>
</table>
**Javier Baro**, UPM, javierbaro@gmail.com  
**Jörg Bartholdt**, Siemens AG, joerg.bartholdt@siemens.com  
**Thorsten Berger**, University of Leipzig, berger@informatik.uni-leipzig.de  
**Nicola Boffoli**, University of Bari, boffoli@di.uniba.it  
**Sholom Cohen**, SEI, sgc@sei.cmu.edu  
**Hyunsik Choi**, Postech, nllbut@postech.ac.kr  
**Peter Dolog**, Aalborg University, dolog@cs.aau.dk  
**Marius Dragouinoiu**, University of Limerick, marius.dragouinoiu@il.ie  
**Sebastian Guenther**, Universität Magdeburg, sebastian.guenther@iti.cs.uni-magdeburg.de  
**Paul Jensen**, Overwatch Textron, paul.jensen@overwatch.com |
| Dolog, Peter & Schafer, Michael. *Feature Based Design of Web Service Transaction Compensations* |  
**Jörg Bartholdt**, Jörg; Franke, Bernd; Schwanninger, Christa; & Stal, Michael. *Combining Product Line Engineering and Service Oriented Architecture in Health Care Infrastructure Systems: Experience Report*  
**Rusk, J. Jeffrey & Gasevic, Dragan. *Semantic Web Services-based Reasoning in the Design of Software Product Lines*  
**Acher, Mathieu; Collet, Philippe; Lahire, Philippe; & Montagnat, Johan. *Imaging Services on the Grid as a Product Line: Requirements and Architecture*  
**Boffoli, Nicola; Caivano, Danilo; Castelluccia, Daniela; Maria Maggi, Fabrizio; & Visaggio, Giuseppe. *Business Process Lines for SOA Development through the Software Product Lines Paradigm*** |

---

**SOAPL 2007**  
**SPLC 2008 Workshops**

**Contact Information**  
Robert Krut  
rk@sei.cmu.edu  
Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213  
Phone: +1-412-268-8505  
Fax: +1-412-268-5758

---

Description

Service-Oriented Architecture (SOA) and software product line (SPL) approaches to software development share a common goal. They both encourage an organization to reuse existing assets and capabilities rather than repeatedly redeveloping them for new systems. The intent is that organizations can capitalize on reuse to achieve desired benefits such as productivity gains, decreased development costs, improved time to market, higher reliability, and competitive advantage. Their distinct goals may be stated as:

- **SOA**: "enable assembly, orchestration and maintenance of enterprise solutions to quickly react to changing business requirements" [Wienands]
- **SPL**: systematically capture and exploit commonality among a set of related systems while managing variations for specific customers or market segments

This workshop will build on results of the SOAPL 2007 workshop: Service-Oriented Architectures and Product Lines - What is the Connection? and the workshop report [Cohen & Krut]. This year's workshop, SOAPL 2008, will explore experiences in integrating SOA and SPL, specifically:

1. How web services have been used to support product lines using a service-oriented architecture?
2. How product line practices have been used to support web services and service-oriented architectures?

Topics of interest for the workshop include, but are not limited to:

- Practice areas that span both SOA and product lines (e.g., domain analysis, legacy mining, operations/governance, etc.)
- Handling variability through services
- Cost models to justify investment in SOA for product lines
- Use of support technology such as: domain specific languages, tools, other
- Differences between service-oriented and more conventional product line development approaches
- Architectural approaches: static vs. dynamic

Audience

Participants in the SOAPL 2008 will include product line and service-oriented practitioners who have experience in integrating service-oriented architectures and software product lines approaches. These include practitioners in product line engineering, product line management, and architects/developers of SOA-based systems.
Schedule

The workshop will be highly interactive and focus on making tangible progress towards answering the two questions relating to results in integrating SOA and product line practices. The morning session will feature invited speakers and selected presentations based on position papers. Participants will be assigned to groups that reflect specific topics. After the workshop, the leader of each working group will be asked to write a summary of the working group's discussion and (especially) its conclusions.

Submission Instructions

Prospective participants are required to submit a 3-6 page position paper or experience report pertaining to the workshop topics listed above or describing the software architecture or other artifacts of a SOA-based product line.

All submissions will be reviewed by members of the program committee for quality and relevance. Accepted papers will become part of the workshop proceedings. Three or four papers will be chosen to be presented during the workshop to foment discussion. Submit your paper in PDF form to soa-workshop@sei.cmu.edu or by July 1, 2008. Notifications of paper or experience report acceptance will be sent by July 15, 2008. The camera-ready version of accepted papers is due July 31, 2008.

Workshop Organizers

- Sholom Cohen, Software Engineering Institute, USA
- Dragan Gasevic, Athabasca University, Canada
- Andreas Helferich, Universität Stuttgart, Germany
- Robert Krut, Software Engineering Institute, USA
- Jaejoon Lee, Lancaster University, UK
- Grace Lewis, Software Engineering Institute, USA
- Tomi Männistö, Helsinki University of Technology, Finland
- Curt Pederson, American Family Insurance, USA
- Dennis Smith, Software Engineering Institute, USA
- Christoph Wienands, Siemens Corporate Research, USA


Workshop on Service-Oriented Architectures and Software Product Lines - Putting Both Together (SOAPL 2008)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Bob Krut & Sholom Cohen
8 September 2008
SPLC 2008, Limerick, Ireland
# Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-09:15</td>
<td>Introductions and Goals</td>
</tr>
<tr>
<td>09:15-10:15</td>
<td>Invited Speaker: Dr Peter Dolog, Aalborg University</td>
</tr>
<tr>
<td>10:15-10:30</td>
<td>Break</td>
</tr>
<tr>
<td>10:30-12:30</td>
<td>Presentations and Discussions (20 minute time limit plus</td>
</tr>
<tr>
<td></td>
<td>questions and discussion on each presentation)</td>
</tr>
<tr>
<td>12:30-14:00</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00-14:30</td>
<td>Final Presentation and Discussion (if necessary)</td>
</tr>
<tr>
<td>14:30-15:30</td>
<td>General Discussion</td>
</tr>
<tr>
<td>15:30-15:45</td>
<td>Break</td>
</tr>
<tr>
<td>15:45-17:00</td>
<td>General Discussion Continued</td>
</tr>
<tr>
<td>17:00-17:30</td>
<td>Conclusion: Goals Addressed, Topics for San Francisco, Future Work</td>
</tr>
</tbody>
</table>
Workshop Organizers

Sholom Cohen, Software Engineering Institute, USA
Dragan Gasevic, Athabasca University, Canada
Andreas Helferich, Universität Stuttgart, Germany
Robert Krut, Software Engineering Institute, USA
Jaejoon Lee, Lancaster University, UK
Grace Lewis, Software Engineering Institute, USA
Tomi Männistö, Helsinki University of Technology, Finland
Curt Pederson, American Family Insurance, USA
Dennis Smith, Software Engineering Institute, USA
Christoph Wienands, Siemens Corporate Research, USA
The First Workshop on Service-Oriented Architectures and Product Lines (SOAPL 2007)


Service Oriented Architectures and Product Lines - What is the Connection?

SOAPL 2007 explored the connections from two perspectives:

1. Can services support product lines using a service-oriented architecture?

2. How can use of product line practices support services and service-oriented architectures?


http://www.sei.cmu.edu/publications/documents/08.reports/08sr006.html
The Second Workshop on Service-Oriented Architectures and Product Lines (SOAPL 2008)

Service Oriented Architectures and Product Lines - Putting Both Together

SOAPL 2008 explores experiences in integrating SOA and SPL:

1. How web services have been used to support product lines using a service-oriented architecture?

2. How product line practices have been used to support web services and service-oriented architectures?

Participants in the workshop hopefully includes product line and service-oriented practitioners who have experience in integrating service-oriented architectures and software product lines approaches.

Five position papers were accepted.
Accepted Papers

Combining Product Line Engineering and Service Oriented Architecture in Health Care Infrastructure Systems: Experience Report
Jörg Bartholdt, Bernd Franke, Christa Schwanninger, and Michael Stal, Siemens AG

Semantic Web Services-based Reasoning in the Design of Software Product Lines
J. Jeffrey Rusk and Dragan Gasevic, Athabasca University

Service-Oriented Product Lines: A Development Process and Feature Management Model for Web Services
Sebastian Gunther, Otto-von-Guericke-Universitat Magdeburg, and Thorsten Berger, Universitat Leipzig

Imaging Services on the Grid as a Product Line: Requirements and Architecture
Mathieu Acher, Philippe Collet, Philippe Lahire, and Johan Montagnat, Universite de Nice

Business Process Lines for SOA Development through SPL Paradigm
Nicola Boffoli, Danilo Caivano, Daniela Castelluccia, Fabrizio Maria Maggi, and Giuseppe Visaggio, University of Bari - Via E.
Workshop Theme

Two major themes for this year’s workshop:

1. variability and variability mechanisms
2. product composition

both within the context of SOA and product lines.

All of the papers touched on one or both of those themes. These topics will provide a starting point for the workshop.

Other suggested topics:

• in advance
• as papers are presented and discussed
Invited Speaker

Dr. Peter Dolog
Associate Professor
Computer Science Department
Aalborg University

Leads the Intelligent Web and Information Systems (IWIS) group

includes adaptive hypertext and hypermedia, user modelling, personalization, web based systems, web services, software product lines and technology enhanced learning.

Presentation Title:
Feature Based Design of Web Service Transaction Compensations
Workshop Topics

How have web services been used to support product lines using a service-oriented architecture?

How have product line practices been used to support web services and service-oriented architectures?

Additional topics:

- variability and variability mechanisms
- product composition
Conclusion

Were the goals of this workshop addressed? Comments?
What would you recommend as the topic for San Francisco?
What future work in this area will you be doing?
Feature Based Design of Web Service Transaction Compensations

Peter Dolog with Michael Schäfer
dolog@cs.aau.dk
CS Department
Intelligent Web Information Systems

SOAPL 2008 @ SPLC 2008, September 2008, Limerick, Ireland
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Intelligent Web and Information Systems
http://iwis.cs.aau.dk

Adaptation Techniques and Algorithms
Different Application Areas
Adaptive Infrastructures/Middleware

Engineering Adaptation
Adaptation/Customization

- Customization by humans (designers)
- Dynamic adaptation by a system itself
- Adaptation is about decision on which information resource or function variant to provide or recommend access to,
- We need a knowledge to decide about appropriate information or service configuration in a certain processing step (user or other):
  - Resource and information access environment
  - Application domain
  - User/Context
  - And their configuration – variants and their meaningful combinations for certain purposes
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Open Web Service Environment

Service Providers
- A number of autonomous service providers exist
- They can provide similar functionality
- They can dis-/appear any time
- Each wants to maximize its profit for executing provided services by external consumers

Service Consumers
- Number of consumers with similar requirements exist
- They want to achieve high value for their expense
- To maximize their service
- By composing matched available services from different providers
Software Product Lines

Software Providers
- Number of reusable software assets exist
- They may vary in its functionality
- They want to maximize its profit by providing the assets in an application in a family mostly from one company

Software Consumers
- Number of consumers with similar requirements
- They want to achieve high value for their expense
- To maximize their service
- By composing a final application from the reusable assets
Difference

Client is composing in web service world
Client is composing from different providers in web service world
Services used in the composition may be exchanged
Question:
  • What can be achieved by current state of the art software product lines techniques?
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Payroll Scenario

Company

Print and send payslip
Transfer salary
Transfer tax

Bank

Print and mail payslip
Transfer tax

Employee

Wait for payment
Transfer monthly instalment for the new car

Transfer car instalment

Employee

Transfer salary

Service Oriented Payroll Scenario

Company

- Transfer salary
- Transfer tax
- Print and mail payslip

Bank

- Transfer car instalment
- Transfer salary
- Transfer tax

Employee

Print and mail payslip

To reach mutually-agreed outcome (commit/cancel)
In environment with concurrent access
Transactions

Control the execution of the required operations on the external services.
Consist of a set of operations (e.g. database operations) that are performed by multiple participants.
Control the collective outcome of the operations.

*Distributed transactions* control the execution of operations on multiple providers.
• Participant
• Coordinator
Error Compensation
Different transaction specifications exist for different purposes

Backward recovery
Normally, predefined rollback operations are executed in order to restore the state before the transaction.
- Time and money is lost
- Dependent transactions also have to roll back (domino effect)

Forward recovery
Aims at changing pro-actively the state of the participant or transaction to enable a successful execution after a failure.
- Complex
- Can normally only be performed semi-automatically
Traditional WS-Transaction Coordin. Structure

1. Create new transaction
2. Return coordination context
3. Invoke service, send coordination context
4. Register with coordination context
5. Confirm registration
6. Process request
7. Send request result
8. Abort transaction

Request failure handling

Web Service 1
Transfer salary

Company: Business Process

Transaction Coordinator

→ Failure
WS – Tx / Business Activity Coordination Type

abstract state diagram
Payroll Processing

1. Transfer of the salary to the employee's account
2. Transfer of the tax to the tax authority's account
3. Specify the salary details, print and send the payslip

Web Service 1
- Transfer salary

Web Service 2
- Transfer tax

Web Service 3
- Print and send payslip

Accounts
- Company
- Employee
- Tax
- Car Dealer
Motivating Scenario – Problem

A service fails due to an internal error. The error can only be compensated by aborting the complete transaction. Why should the transaction be aborted, if a different service exists that can perform the same operations?
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Extended Transaction Coordination Structure

1. Create new transaction
2. Return coordination context
3. Invoke abstract service, send coordination context
4. Request adapter context
5. Register with coordination context
6. Confirm registration
7. Return adapter context
8. Invoke concrete service, send adapter context
9. Register with adapter context
10. Confirm registration
11. Process request
12. Send request result
13. Send request result
New Components - Abstract Service

Does not directly implement functionalities. Manages a list of *concrete services*. Is a mediator between the client and the concrete service. Manages and performs compensation actions. Interfaces:

- Service
- Event (internal compensation handling)
- Compensation (external compensation handling)
- Contract exchange

Abstract Service

- Management
  - Concrete service list
  - Concrete service wrappers
  - Compensation rules repository
  - Contract repository

Contract exchange

Event
### Compensation Activities and Types

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Compensation Type</th>
<th>ServiceReplacement</th>
<th>LastRequestRepetition</th>
<th>PartialRequestRepetition</th>
<th>AllRequestRepetition</th>
<th>CompensationForwarding</th>
<th>AdditionalServiceInvocation</th>
<th>AdditionalRequestGeneration</th>
<th>ServiceAbortInitiation</th>
<th>RequestSequenceChange</th>
<th>ResultResending</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>NoCompensation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>Repetition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>Replacement</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>Replacement</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>Forwarding</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
<td>(X)</td>
</tr>
<tr>
<td>08</td>
<td>AdditionalService</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>AdditionalRequest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SessionRestart</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X Included compensation activity  (X) Possibly included compensation activity
Example: Internal Compensation Rule

```xml
<cmp:InternalCompensationRule identifier="internalFailureLastRequestResending">
  <cmp:CompensationCondition>
    <cmp:ParticipantState stateType='http://schemas.xmlsoap.org/ws/2004/10/wsba/Faulting' />  
    <cmp:ReplacementService exists="true" isDirectReplacement="true" />
    <cmp:RequestSequence>
      <cmp:Request identifier="transferSalaryMethod" />
    </cmp:RequestSequence>
  </cmp:CompensationCondition>
  <cmp:CompensationPlan>
    <cmp:Compensation>
      <cmp:ServiceReplacement/>
    </cmp:Compensation>
    <cmp:Compensation>
      <cmp:RequestResending lastN="1" />
    </cmp:Compensation>
  </cmp:CompensationPlan>
</cmp:InternalCompensationRule>
```

Condition of the compensation rule:

Condition 1: The event must have been a failure of the concrete service.

Condition 2: The state in which the concrete service has to be.

Condition 3: A direct replacement concrete service has to exist.

Condition 4: The last request must have called this method.

Execution plan of the compensation rule:

Step 1: Replace the current concrete service.

Step 2: Resend the last request.
New Components - Adapter

Encapsulates coordinator-specific functionality. Functions as a coordinator for the concrete service. Manages messaging:

- Forwards normal messages between the real coordinator and the concrete service.
- Intercepts failure messages and informs the abstract service.
- Creates additional notifications as part of a compensation process.
Internal Compensation Handling – No Action


11. Process request
12. Signal failure
13. Report event
14. Fail
15. Forward failure notification
16. Abort transaction
Internal Compensation Handling – Replacement

Concrete service fails. Abstract service checks its compensation rules and contract. Concrete service is replaced. Coordinator was not notified!
Evaluation

Multiple scenarios for internal and external compensation handling have been implemented and tested. An evaluation model has been created, which calculates net values for the standard environment and the abstract service environment.

- Allows an assessment whether the utilization of the new design is economical and beneficial.

Experiment performed on a simulated environment

More in ACM TWEB paper
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Compensation Types

<< Concept >> Compensation
<< Concept >> CompensationPlan
<< Concept >> Compensation Action
<< Concept >> NoCompensation
<< Concept >> Repetition
<< Concept >> Replacement
<< Concept >> Forwarding
<< Concept >> AdditionalService
<< Concept >> AdditionalRequest
<< Concept >> SessionRestart
Compensation Features

<< Concept >> Compensation

<< OptionalFeature >> Compensation

<< OptionalFeature >> ExternalCompensation Handling

<< OptionalFeature >> SessionRestart

<< VariationPoint >> {Kind = AND}

<< MandatoryFeature >> ServiceAbort

<< OptionalFeature >> AdditionalService

<< VariationPoint >> {Kind = OR}

<< OptionalFeature >> AdditionalRequest

<< OptionalFeature >> PartialRequest Repetition

<< MandatoryFeature >> LastRequest Repetition

<< OptionalFeature >> RequestSequence Change

<< VariationPoint >> {Kind = OR}

<< OptionalFeature >> InternalCompensation Handling

<< OptionalFeature >> Forwarding

<< VariationPoint >> {Kind = OR}

<< OptionalFeature >> AdditionalActions

<< OptionalFeature >> NoCompensation

<< MandatoryFeature >> Repetition

<< OptionalFeature >> Replacement

<< OptionalFeature >> AdditionalActions

<< VariationPoint >> {Kind = AND}

<< OptionalFeature >> ServiceAbort

<< OptionalFeature >> AdditionalService

<< OptionalFeature >> PartialRequest Repetition

<< MandatoryFeature >> LastRequest Repetition

<< OptionalFeature >> AllRequest Repetition

<< OptionalFeature >> AdditionalRequest

<< MandatoryFeature >> ResultResending
Capability Feature Model

Consists of:
- **functionality feature model**
- **compensation feature model**

The compensation feature model can contain custom features.
Service Capabilities

<< Concept >>
SalaryTransferService

<< OptionalFeature >>
ExternalCompensation Handling
<< OptionalFeature >>
SessionRestart

<< OptionalFeature >>
Forwarding
<< VariationPoint >>
{Kind = AND}
<< MandatoryFeature >>
ServiceAbort
<< MandatoryFeature >>
RequestSequence Change

<< MandatoryFeature >>
TransferSalary
<< OptionalFeature >>
AdditionalService
<< MandatoryFeature >>
TelephoneCall
<< VariationPoint >>
{Kind = OR}
<< OptionalFeature >>
AdditionalRequest

<< MandatoryFeature >>
GetAccountBalance
<< OptionalFeature >>
PartialRequest Repetition

<< MandatoryFeature >>
InternalCompensation Handling
<< MandatoryFeature >>
NoCompensation
<< OptionalFeature >>
LastRequest Repetition

<< MandatoryFeature >>
Replacement
<< OptionalFeature >>
Replication
<< OptionalFeature >>
ResultResending
Consumer Requirements

<< Concept >>
SalaryTransferService

<< MandatoryFeature >>
ExternalCompensation Handling

<< MandatoryFeature >>
TransferSalary

<< MandatoryFeature >>
GetAccountBalance

<< MandatoryFeature >>
InternalCompensation Handling

<< MandatoryFeature >>
SessionRestart

<< OptionalFeature >>
Forwarding

<< OptionalFeature >>
AdditionalActions

<< MandatoryFeature >>
NoCompensation

<< MandatoryFeature >>
Replacement

<< VariationPoint >>
{Kind = AND}

<< VariationPoint >>
{Kind = OR}

<< MandatoryFeature >>
ServiceAbort

<< OptionalFeature >>
AdditionalService

<< OptionalFeature >>
AdditionalRequest

<< OptionalFeature >>
PartialRequest Repetition

<< MandatoryFeature >>
LastRequest Repetition

<< OptionalFeature >>
AllRequest Repetition

<< MandatoryFeature >>
RequestSequence Change

<< MandatoryFeature >>
ResultResending

SOAPL 2008: Feature Based Design of Web Service Transaction Compensations
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Matchmaking between service and consumer feature models

Compatibility score calculation
Iteratively compares feature models
Features must appear at the same place in the graph
Mandatory features must all match but do not contribute to the compatibility score
If a mismatch is found in a mandatory feature, algorithm stops and a negative score is returned
Optional features add to the compatibility score when a match is found (in our case +1)
Additional features may contribute with different scores
Restriction Feature Model

<< Concept >> Compensation

<< Feature >> InternalCompensation Handling

<< Feature >> NoCompensation
<< Feature >> Repetition
<< Feature >> Replacement

<< Feature >> PartialRequest Repetition
<< Feature >> LastRequest Repetition
<< Feature >> AllRequest Repetition

<< Feature >> ResultResending
Example: Internal Compensation Rule

```xml
<cmp:InternalCompensationRule identifier="internalFailureLastRequestResending">
  <cmp:CompensationCondition>
    <cmp:ParticipantState stateType='http://schemas.xmlsoap.org/ws/2004/10/wsba/Faulting' />
    <cmp:ReplacementService exists="true" isDirectReplacement="true" />
    <cmp:RequestSequence>
      <cmp:Request identifier="transferSalaryMethod" />
    </cmp:RequestSequence>
  </cmp:CompensationCondition>
  <cmp:CompensationPlan>
    <cmp:Compensation>
      <cmp:ServiceReplacement/>
    </cmp:Compensation>
    <cmp:Compensation>
      <cmp:RequestResending lastN="1" />
    </cmp:Compensation>
  </cmp:CompensationPlan>
</cmp:InternalCompensationRule>
```
Feature Model

```xml
<feature name="Compensation" type="NONE" id="compensation">
  <feature name="InternalCompensationHandling" type="NONE" id="internalCompensationHandling">
    ...
    <feature name="PartialRequestRepetition" type="NONE" id="reference3IXIpartialRequestRepetition">
      <feature name="ResultResending" type="NONE" id="reference3IXIreferenceIXIresultResending">
        ...
      </feature>
    </feature>
  </feature>
</feature>

<feature name="Replacement" type="NONE" id="replacement">
  <feature name="LastRequestRepetition" type="NONE" id="reference4IXIlastRequestRepetition">
    ...
    <feature name="PartialRequestRepetition" type="NONE" id="reference5IXIpartialRequestRepetition">
      <feature name="ResultResending" type="NONE" id="reference5IXIreferenceIXIresultResending">
        ...
      </feature>
    </feature>
    <feature name="AllRequestRepetition" type="NONE" id="reference6IXIallRequestRepetition">
      <feature name="ResultResending" type="NONE" id="reference6IXIreferenceIXIresultResending">
        ...
      </feature>
    </feature>
  </feature>
</feature>

...
Layers of Abstraction

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Features and Configurations</td>
<td></td>
<td></td>
<td>x̂ or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restrictions</td>
<td></td>
<td></td>
<td>and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigational and Interaction</td>
<td></td>
<td></td>
<td>x̂ or</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOAPL 2008: Feature Based Design of Web Service Transaction Compensations 39
Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges
Workflows vs. Middleware

Compensations and adaptations can be specified at the design level in workflows
Compensations and adaptations can be encoded in an intelligent middleware
How to combine them
How to compose them
How to ensure consistency
…
FP7 ICT EU idSpace: Tooling of and training for collaborative, distributed product
References

Thanks!!! Questions?

dolog@cs.aau.dk
http://www.cs.aau.dk/~dolog
http://iwis.cs.aau.dk
Combining Product Line Engineering and Service Oriented Architecture in Health Care Infrastructure Systems: Experience Report

Jörg Bartholdt, Bernd Franke, Christa Schwanninger Michael Stal
Corporate Technology & Health Care
Siemens AG
Business Case

Hospitals have a HIS (Hospital Information System). Data is shared between departments (intra-hospital)

But what if it comes to transferring a patient to another hospital? You carry your X-ray images with you

Soarian IC targets
§ inter-hospital communication
§ Special scenarios of external data integration
In future: target residential doctors, too
History

Product development was serialized
Previous version forms the bases for the next version (architecture erosion)
Results in monolithic application, interwoven dependencies

Assumptions:
Increased customer base (no serialization possible anymore)
Focus on main selling assets
Make system ready for integration

Goal:
*Introduce SOA-approach:* import/export via interfaces, composition of features via service chaining
*Introduce PLE:* focus on core assets, allow for customer specific variations, introduce new features in core if proven at one customer
Challenges

1. Increasing variability
2. Configurability/Subset-ability
3. Extensibility
4. Increased testability
5. Outsourcing
6. Risk effect mitigation
7. Exploitation of COTS (Common-Off-The-Shelf) products
8. Prioritization of features to be integrated in the platform
9. Positioning in the market (guide the customer)
10. Acceleration of tender preparation
11. Clinical workflows
12. Traceability
Approach

1. Scoping (2,8,9,10):
   § Increasing customer base requires focus on most profitable features
   § Starting point: Group current requirements to features
   § Use feature model for reasoning with product mgmt, sales, development, etc („common language“)

2. Variability Management (1,3,4,12):
   § Reduce variability points (expensive!) pre-configurations

3. Building re-use culture (1,2,4,10):
   § Keep clear product portfolio strategy
   § Focus to market commonalities
   § Quick hacks forbidden in the core assets
Approach

4. Self-containment (2,3,4,5,6,12):
   - Fosters decoupling of components
   - Allows for exchange to third-party components
   - Allows to be used as a system, not only by humans via Web-Interface
   - Improves testability

5. Integration (2,7):
   - More freedom to tailor to customer needs
   - Face the fact that Siemens is not the only supplier

6. Flexibility (5,11):
   - Adding workflow or rule engines
   - support specifics of each customer (ideally by the customer)
   - Late (dynamic) binding
Other projects showed the likelihood of failure in a big-bang approach.
We favor a migration strategy.
Conclusion

😊 SOA builds a prominent, natural variation point with late (dynamic) binding capabilities.

😊 Services as a variation point means flexible tooling available (Workflow engines, BPEL)

😊 Self-containment reduces coupling and fosters variation

We will not follow the total unawareness of the usage context implied by SOA protagonists.

Future challenges

☐ Data model can not be changed as long as old application components exist

☐ Restructure the organization (nobody wants to lose influence, learning-curve)

☐ Wrap legacy system with new service interface without side-effects
Questions & Answers

Now, or later …

Joerg.Bartholdt@Siemens.com
Semantic Web Services-based Reasoning in the Design of Software Product Lines

J. Jeffrey Rusk and Dragan Gasevic
Athabasca University
Canada
Research Goal

To evaluate the suitability of the Web Service Modeling Ontology (WSMO) in the encoding of product configurations and related constraints from a software product line (SPL) in such a manner as to better enable reasoning approaches which facilitate higher automation of service discovery, composition, invocation, and monitoring in service oriented architectures (SOA).
Outline

• Background and Motivation
• Feature Models (FM)
• Web Service Modeling Ontology (WSMO)
• Model Transformations
  – FM to WSMO
  – Product Configuration to WSMO
• Orchestration in WSMO
• Reasoning
• Implementation, Conclusion and Future Work
Background Issues

- Impediments to successful implementation of SPL when considering SOA
- Challenges representing SOA as SPL
- Limits to the expressiveness of FM
- Limited reasoning capabilities
- Ontology-related technology exists to support
Deliverables

- Mappings between FM and WSMO
- Transformation implementation
- Reasoning framework
What do the deliverables make possible?

The ability to explore and evaluate:

- accuracy of the mapping possible between the two formalisms.
- level of automation supported during transformation
- support or guidance that the ontology can provide to feature modeling.
Themes of this Workshop

- Variability and variability mechanisms
- Product composition

How does this work relate to these themes?
Overall Flow of Information

[Diagram showing the flow of information from an existing ontology to SOA, involving steps like incorporate, transform, derive, and product configuration.]
Feature Models

• SPL implementations typically feature-based
• FM ideal representation for SOA
• Using Czarnecki et al. notation and rendering
• Metamodel of FM and product configurations
• Tool support
Web Service Modeling Ontology (WSMO)

- Semantic describes all aspects of SWS
- Relatively new framework
- Tool support
- Four core elements
  - Ontologies
  - Web Services
  - Goals
  - Mediators
WSMO Metamodel
XML Formats

Feature Plugin XML Export

```xml
<feature min="1" max="1" name="VirtualWholesale" type=""/>
<feature min="1" max="1" name="RewardsProgram" type=""/>
<feature min="1" max="1" name="Provider" type=""/>
<featureGroup min="1" max="1" id="group">
  <feature min="0" max="1" name="AtlanticProduct" type=""/>
  <feature min="0" max="1" name="NorthernDesigns" type=""/>
  <feature min="0" max="1" name="lakewoodReturbish" type=""/>
  <feature min="0" max="1" name="QualityImports" type=""/>
</featureGroup>
</feature>
```

WSML-in-XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsml>
<ontology name="http://www.example.org#Virtual">
<nonFunctionalProperties>
</attributeValue>
</nonFunctionalProperties>
<concept name="http://www.example.org#Virtual">
<attribute name="http://www.example.org#Registration">
<range>http://www.example.org#RewardsProgram</range>
<minCardinality>1</minCardinality>
<maxCardinality>1</maxCardinality>
</attribute>
</concept>
</ontology>
</wsml>
```

ATL

```
FM2WSMO : Module
OUT : OclModel
IN : OclModel
FM1 : MatchedRule
FM : MatchedRule
```
Feature Model to WSMO

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

ontology VirtualWholesale

concept VirtualWholesale
  Registration ofType (1 1) Registration
  RewardsProgram ofType (0 1) RewardsProgram
  Payment ofType (1 1) Payment
  Provider ofType (1 4) Provider
  Shipping ofType (1 4) Shipping

concept Registration
concept Payment
concept Provider
concept Shipping
concept RewardsProgram
concept Visa subConceptOf Payment
concept Mastercard subConceptOf Payment
concept AtlanticProductsLtd subConceptOf Provider
concept NorthernDesigns subConceptOf Provider
concept LakewoodRefurbishing subConceptOf Provider
concept QualityImportsLtd subConceptOf Provider
concept CanadaPost subConceptOf Shipping
concept Purolator subConceptOf Shipping
concept FederalExpress subConceptOf Shipping
concept Midland subConceptOf Shipping

axiom DisjointPayment
  definedBy
    !- ?x memberOf Visa
    and ?x memberOf Mastercard.
Product Configuration to WSMO

- Most accurately represented as orchestration
- Overall executable business process that can be defined through interaction between Web services
- Choreography may be a factor as well
Orchestration in WSMO

• Unlike choreography, orchestration in WSMO is still under development
• Both based on abstract state machine
• Composed of state and set of guarded transitions
• State in form of ontology providing
  – Vocabulary for transition rules
  – Set of instances that change state.
Reasoning

• Effects of:
  – Product configuration choices
  – Adding, moving, deleting features
  – Assigning values to attributes

• Guidance for:
  – Constraints in ontology not present in FM
  – FM relationships not represented in ontology
  – Orchestration dependencies
  – Orchestrating services required
Implementation

- Feature Model Plugin
- WSMO Studio and KAON2 Reasoner
- ATL
- Eclipse
Conclusion

• Mappings between FM and WSMO
  – Accuracy of mappings
  – Level of automation attainable
  – Precision of feature discovery
  – Guidance provided by ontology

• Suitability of WSMO
  – Expressiveness
  – Related work
Future Work

- Refine mappings
- Improve transformation
- Explore further the available reasoning
- Integrate the various utilities into comprehensive plugin working in Eclipse environment
Thank you!
Questions?

Jeff Rusk
Director, Technical Services
Nunavut Impact Review Board
Cambridge Bay, Nunavut, Canada
(also MSc Candidate at Athabasca University)

jrusk@nirb.ca

Acknowledgements
The authors acknowledge the support of Athabasca University through the Graduate Student Research Fund.
Service-Oriented Product Lines: Towards a Development Process and Feature Management Model for Web Services

SOAPL 2008
Introduction

§ **What is software development?**
  - Usage of a software development process
  - Transform requirements into different artifacts (architectural descriptions, interface descriptions, source code…)

§ **How to manage artifacts?**
  - Apply changes to existing artifacts
  - Reduce coupling of source code

§ **What about reuse?**
  - Commonality and variability

§ **Combination of Software Product Lines and Service-Oriented Architectures provides solutions to many common software problems**
Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions
Definitional Background: Software Product Lines

- Withey: „Product Lines is a group of products sharing a common, managed set of features“ [1]
- Specifically, manage variability among features that represent requirements
- Goal: Structure and reuse software development artifacts
Definitional Background: Service-Oriented Architectures

- Loosely coupled and autonomous services
- Properties according to Josuttis: self-containment, coarse-grained interfaces, reusability and composability [2]
- Implementation: Web Services or Enterprise Service Bus
Definitional Background: Web Services

§ „Software applications that can be discovered, described and accessed based on XML and standard Web protocols“ [3]

§ Described by a WSDL
  - Abstract definition describes interface, operations and messages
  - Concrete definition describes bindings to operations

§ Distinguish into service broker, provider and consumer
Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions
Domain Engineering

- **Develop a software family**

- **Analysis**
  - Capture domain specific knowledge
  - Develop a domain model
  - Represent domain concepts and requirements in a central feature model
  - Identify variants with their distinguishing features

- **Design**
  - From architectural description to software entities
  - Decide used frameworks, libraries and programming languages
  - Form technological foundation for implementation of variants

- **Implementation**
  - Make or buy decision for software entities
Application Engineering

§ Develop individual member (of the software family)

§ Five steps
  • Problem Analysis (overall problem specification)
  • Product Specification (concrete set of selected features)
  • Collateral Development (Documentation)
  • Product Implementation (Executables and test cases)
  • Deployment
Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions
Service-Oriented Product Lines

- Implement SPL with an SOA

- Different impacts on development phases
  - Analysis:
    - Select SOA-specific modeling languages
    - Software requirements can be modeled as features or part of the ESB
  - Design
    - ESB as routing and messaging backbone, and also implements e.g. compliance requirements
    - ESB mostly forms common part of SPL
    - Web Service abstracts whole applications, databases or fine granular software entities
  - Implementation
    - Careful choice of purchased ESB
    - Wrap existing software with Web Services or use web service repositories
    - Full SOPL process (design interface and implementation) vs. light SOPL process (design only interfaces)
Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions
Domain Engineering for a Web Store (base taken from [4])

Web store contains 7 modules:

- Inquiry
- Acquisition
- Availability Checking
- Credit Ranking
- Order
- Ordering
- Payment Checking
- Billing
- Shipping
- Bill
Abstract representation of the overall product line
Uses set-like notations for features
- $Base = \{Acq, Chk, Crd, Ord, Shp, Bil, Pay\}$

Detail out features
- Credit Ranking: Use an independent agency (Agc) or explanation of the bank (Bak)
  - $Crd = \{Agc, Bak\}$
- Shipment via surface (Sur) or airmail (Air)
  - $Shp = \{Sur, Air\}$
- Surface shipment with standard (Std) or Express (Exp) Mail
  - $Sur = \{Std, Exp\}$

Individual member is a composition of specific features
- $Store1 = Base$
- $Acq \cdot Chk \cdot Agc \cdot Bak \cdot Ord \cdot Std \cdot Exp \cdot Air \cdot Bil \cdot Pay$
Customers demand new features
- Discounting for bigger quantities of ordered goods
- Traceability of features

Impacts existing services of the Web Store

- Different implementations and possibly different interfaces
§ **Discounting concepts refines four basic features**

- \[ \text{Disc} = \{ \text{Crd} = \{ \Delta \text{Agc}, \Delta \text{Bak} \}, \Delta \text{Bil}, \Delta \text{Pay} \} \]

§ **Build a new member**

- Include discounting feature
- Limit shipment to standard surface mail
- \[ \text{Store2} = \{ \text{Base} - \{ \text{Exp}, \text{Air} \} \} \cdot \text{Disc} \]
- \[ \text{Store2} = \text{Acq} \cdot \text{Chk} \cdot \text{Agc} \cdot \text{Bak} \cdot \text{Ord} \cdot \text{Std} \cdot \text{Bil} \cdot \text{Pay} \cdot \text{Disc} \]
- \[ \text{Store2} = \text{Acq} \cdot \text{Chk} \cdot \text{Agc} \cdot \text{Bak} \cdot \text{Ord} \cdot \text{Std} \cdot \text{Bil} \cdot \text{Pay} \cdot \Delta \text{Agc} \cdot \Delta \text{Bak} \cdot \Delta \text{Bil} \cdot \Delta \text{Pay} \]

§ **Combination of a basic and refined feature leads to the final representation**

- \[ \text{Store2} = \text{Acq} \cdot \text{Chk} \cdot \text{Agc}´ \cdot \text{Bak}´ \cdot \text{Ord} \cdot \text{Std} \cdot \text{Bil}´ \cdot \text{Pay}´ \]
Description of Web Services with WSDL gives a high level view

Feature granularity must manage WSDL descriptions

Example: WSDL for Billing

- `<element name="CalcBillOutput">
  - <!-- Other definitions omitted -->
  - `<xsd:sequence>
    - `<xsd:element name="customerName" type="xsd:string"/>
    - `<xsd:element name="customerAddress" type="xsd:string"/>
    - `<xsd:element name="items" type="ItemOrder" minOccurs="1" maxOccurs="unbound"/>
    - `<xsd:element name="totalPrice" type="xsd:integer"/>
  - </xsd:sequence>
  - <!-- Other definitions omitted -->
  - </element>

Variability Management with XAK [5]

- `<element name="CalcBillOutput" xak:artifact="STOREBillOutput">
  - <!-- Other definitions omitted -->
  - `<xsd:sequence xak:module="billOutput">
    - `<xsd:element name="customerName" type="xsd:string"/>
  - <!-- Other definitions omitted -->
  - </xsd:sequence>
WSDL Refinement

- `<xak:refines xak:artifact="STOREbillOutput">`
- `<xak:extends xak:module="billOutput">`
- `<xak:super xak:module="billOutput"/>`
- `<xsd:element name="discount" type="xsd:integer"/>`
- `<xsd:element name="discountPrice" type="xsd:integer"/>`
- `</xak:extends>`
- `</xak:refines>`

Combined WSDL

- `<element name="CalcBillOutput">`
- `<xsd:sequence>`
  - `<xsd:element name="customerName" type="xsd:string"/>`
  - `<xsd:element name="customerAddress" type="xsd:string"/>`
  - `<xsd:element name="items" type="ItemOrder" minOccur="1" maxOccur="unbound"/>`
  - `<xsd:element name="totalPrice" type="xsd:integer"/>`
  - `<xsd:element name="discount" type="xsd:integer"/>`
  - `<xsd:element name="discountedPrice" type="xsd:integer"/>`
  - `</xsd:sequence>`
- `<!-- Other definitions ommitted -->>`
Conclusions

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions
Conclusions

- Feature models and variability management models can be used for Service-Oriented Product Lines as well

- XML refinements allow practical solution to feature management

- Focus on models leads to a high-level view

- Promising
  - If existing code base can be reused efficiently: focus on light SOPL process (only define interfaces)
  - Introduce Domain Specific Languages for domain modeling and SPL configuration, allowing participation of end-users
Thanks for your attention!
References


Imaging Services on the Grid as a Product Line: Requirements and Architecture

M. ACHER, Ph. COLLET, Ph. LAHIRE, J. Montagnat

Workshop SOAPL 2008
September 8th
Context: Services for the Grid

- **Grid**
  - sharing data, algorithms
  - computation power, data-intensive

- **Workflows for the e-Science Grid**
  - process chain, pipeline, data flow
  - reuse and compose (black) boxes

- **Implemented as Services**
Requirements Overview

Functional
- Format: DICOM
- Acquisition Model: MRI
- Anatomic Structure: Brain
- Noise: Not significant

QoS
- Accuracy: 80%
- Security: None
- Reliability: 50%
- Execution time: Not specified

Services for image segmentation:
- S1
- S2

Functional
- Format: DICOM
- Acquisition Model: MRI
- Structure Anatomique: Stomach
- Bruit: Dedicated to noisy images

QoS
- Evaluation: Statistical
- Accuracy: 90%
- Reliability: 30%

Functional
- Format: DICOM
- Acquisition Model: MRI
- Structure Anatomique: Brain
- Bruit: Noise sensitive

QoS
- Evaluation: Statistical
- Accuracy: 85%
- Reliability: 50%
Composing Services on the Grid

- How to deploy Grid Services?
  - needs fine-grained information

- How to manage QoS (Quality of Service)?
  - such as execution time, availability, reliability, etc.

- To give information to ...
  - workflow engine, software architect, scheduler

- Our position: a variability problem!
From Service to Product Line (1)
From Service to Product Line (2)
Functional Variability

extract: inputs

Magnetic Resonance Imaging
Functional description: example

**Acquisition Model**
MRI = MRI T2

**Resolution**
Spatial Resolution
- Dimension = 2D
- Color = B&W
- Noise = none

**Anatomic Structure** = brain

**Format** = DICOM
QOS Variability

How to **characterize**
How to **measure**
How to **compute**

- Time
- Cost
- Security
- Accuracy
- Reliability
QoS description : example

Metric
- measurable = true
- unit = %
- comparable = true
- type = numeric

Dimension
- accuracy = high
- time = any

Computation
- dynamic = true
- rely_on = output
- accuracy = good

Measurable
- Metric
  - Value Type
    - Comparable
      - Unit
        - %
  - Accuracy
    - Numeric
  - Dynamic
    - Output
      - good
Segmentation: refining classification

QoS depends on application domain:
  goal of segmentation
  body region
  imaging protocol

“A particular segmentation may have high performance in determining the volume of a tumor in the brain on an MRI image,
... but may have low performance in segmenting a cancerous mass from a mammography scan of a breast”

Dimensions: time and space complexity, accuracy, robustness, precision, specificity, sensibility

Interdependency between QOS and Computation of QoS:
  costly but precise
  quick but uncertain
evaluation has a QoS too
Towards SPL: big picture

Functional QOS

Medical imaging

Acquisition Model

Resolution

Anatomic Structure

Format

MRI

MRI T2

3D

Black&White

Noise

None

DICOM

Functional QOS

QOS Property

Metric

Value Type

Accuracy

Computation

Comparable

Numeric

Accuracy

Output

QOS

Contrôle

Adaptation

Selection

Services de segmentation

S1

Sn
Towards Service product line

Service Product Line

+ variability

Behaviour + QoS

Service Metamodell
QoS Metamodell
Grid Metamodell

Service Repository

Medical imaging computation expert

Grid workflow expert

Workflow

One Workflow
An MDE Approach

**Approach**
- Model Driven Engineering (MDE)
- Platform independent, abstraction
- Model transformation and/or model composition

**Equipping Service/Workflow with meta information**
- A common core (QOS & service metamodels)
- Specific branches

**Building the SPL**
- Describing a generic Domain-Specific service / workflow
- Specifying composition protocol of one service
- Allow to address different workflow
- Includes also variability
An MDE Approach

Model-Driven Engineering

- Service Metamodel
- QoS Metamodel
- GRID Metamodel

\[ \text{SPL Metamodel} \]

- eHealth domain
- Instance of the SPL

Model abstraction of services

- Model abstraction of services

Service Repository

- Workflow?
- Service Composition

Selection

Deployment

- script
- GRID Engine

Platform dependent

Transformation
On-going Work

- QoS multi-views
  - experts collaboration
  - from end users to services
- How to infer a SPL?
- Derivation process
  - who for the reasoning process?
  - heuristics needed
- From Service to workflow
From Service to Workflow

Behaviour + QOS + variability

Service Metamodel
QoS Metamodell

Service Repository

Grid workflow expert
Medical imaging computation expert
Questions?
Business Process Lines to develop Service-Oriented Architectures through the Software Product Lines paradigm

Nicola Boffoli, Danilo Caivano, Daniela Castelluccia, Fabrizio Maria Maggi, Giuseppe Visaggio
SERLAB - Department of Informatics
University of Bari - Italy
{boffoli, caivano, castelluccia, maggi, visaggio}@di.uniba.it
Outline

SPL + SOA
- Why?
- What?
- How?

Our proposal
- Business Process Line
- Decision Models
- Case Study
SPL + SOA: Why?

Two common perspectives

- **Software reuse**
  - implementing new software systems reusing existing software resources rather than developing the same software capabilities again

- **Software flexibility**
  - allowing to adapt the systems to the different customers of a whole market segment
    - **SPL** focuses on the commonality and variability to build a set of software products
    - **SOA** allows to compose, orchestrate and maintain solutions based on services, implementing business processes
SPL + SOA: What?

Our Proposal

- transferring peculiarities/advantages from SPL to SOA
- build a SOA systems line suitable to customers or market segments needs in a specific application domain
SPL + SOA: How?

We start from a deep analysis of the business processes identifying in them commonality and variability typical of the SPL paradigm.

Business Process Line + Decision Models
Business Process Line (BPL)

A BPL realizes processes able to adapt themselves to different customer needs

- Each process of a BPL can be then transformed into the corresponding SOA system
  - If the business processes are adaptable to the customer needs
  - then the generated SOA system, it will result in its turn suitable to the specific customer requirements
From SPL to BPL: Analogies and Tailoring ...

**SPL**

- Collection, organization and systematic refinement of the assets (invariant or variant)
- Automatic building of the products
  - **Product Configuration**: through asset integration procedures
  - **Product Specialization**: through the specification of the assets parametric part
... From SPL to BPL: Analogies and Tailoring

BPL

- **Asset** concept is referred to activities and work definitions
- **Product Configuration** ⇣ **Process Configuration**
  - the assets (activities and work definitions) can be added to a basic business process in order to configure the target business process
- **Product Specialization** ⇣ **Process Specialization**
  - each asset of the target business process can be specialized through attributes indicating specific architectural characteristics to implement them
Hypothesis: two kind of relations

1. between the business capabilities (characterizing the customer needs) and the suitable processes elements (that have to be integrated in the target business process)

2. between the customer requirements and the specific peculiarities of the processes elements previously integrated in the target process.
Decision Table Formalism

A decision table (DT) is divided in four quadrants: conditions (Cond), conditional states (S), actions (Act) and rules (x).

The table is defined so that each combination of conditions and conditional states corresponds to a set of actions to carry out.

- Compact overview
- Modular knowledge organization
- Evaluation of consistency, completeness and redundancy
Configuring DT ...

For each BPL a configuring DT is built in order to select the variant assets characteristic of the requested business capabilities.

They have to be composed with the invariant assets (integrated into a basic process) in order to generate the target business process.
The CONDITION quadrant contains a set of business capabilities, \( BC_i \), for \( i = 1, \ldots, n \).

The CONDITIONAL STATE quadrant contains the possible values of each business capability, \( BC_i = \{bc_{i1}, bc_{i2}, \ldots, bc_{iq}\} \).

The ACTION quadrant contains all the possible variant assets \( \{va_1, va_2, \ldots, va_r\} \) that can be added to the process.

The RULE quadrant relates each capabilities profile to the corresponding variant assets.
Business Process Lines to develop Service-Oriented Architectures through the Software Product Lines paradigm

Specializing DT

For each asset, variant or invariant, a specializing DT is built as follows:

- the CONDITION quadrant contains a set of customer requirements, \( CR_j \), \( j = 1, \ldots, m \), to specialize the parametric part of the asset.
- the CONDITIONAL STATE quadrant contains the possible values of each requirement: \( [CR_j] = \{cr_{j1}, cr_{j2}, \ldots, cr_{jt}\} \).
- the ACTION quadrant contains the parameters \( \{p_1, p_2, \ldots, p_s\} \) and their values allowing to specialize the parametric part of the asset.
- the RULE quadrant relates each customer requirements value set to the corresponding specializing parameters.

<table>
<thead>
<tr>
<th>CR_1</th>
<th>fr_{11}</th>
<th>fr_{12}</th>
<th>fr_{13}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR_2</td>
<td>fr_{21}</td>
<td>fr_{22}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR_n</td>
<td>fr_{n1}</td>
<td>fr_{n2}</td>
<td>fr_{n1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_1 = “Y”</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>p_1 = “N”</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>......</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_s = “x”</td>
<td>x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>p_s = “y”</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>p_s = “z”</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Case Study ...

Our proposal has been investigated in an industrial case during the research project "DAMA" (Data Archiving Management and Acquisition)

- A specific part, Document Recognizing, is here summarized

Invariant Part

- the process contains an OCR (Optical Character Recognition) activity requiring a scanned Document Image as input and produces a recognized Text Document as output

![Diagram of Document Image to Text Document process](image)

DIB
... Case Study ...

Configuring DT

- the table provides the following business capabilities: Signature Extraction, Layout Analysis and Image Extraction

<table>
<thead>
<tr>
<th>Signed Documents</th>
<th>Document Type</th>
<th>Document with Images</th>
<th>Digital</th>
<th>Autographic</th>
<th>Without Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Structured</td>
<td>Unstructured</td>
<td>Structured</td>
<td>Unstructured</td>
<td>Structured</td>
</tr>
<tr>
<td>Layout Analysis</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Image Extraction</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Autographic Sign Extraction</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Digital Sign Extraction</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
... Case Study ...

Scenario

"The enterprise needs besides to elaborate and archive typewriting and structured documents, containing images and without signature"

<table>
<thead>
<tr>
<th>Signed Documents</th>
<th>Digital</th>
<th>Autographic</th>
<th>Without Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Type</td>
<td>Structured</td>
<td>Unstructured</td>
<td>Structured</td>
</tr>
<tr>
<td>Documents with Images</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Layout Analysis</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Image Extraction</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Autographic Sign Extraction</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Digital Sign Extraction</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Business Process Lines to develop Service-Oriented Architectures through the Software Product Lines paradigm

... Case Study

Diagram showing the process:
- Document Image
- OCR
- Text Document

Another diagram:
- Document Image
- Layout Analysis
- Doc/Layout Structured
- OCR Typewriting
- Extracted Image
- Text Document
Conclusion ...

This work proposes to apply the good practices of SPL to SOA, the authors introduce

- the concept of BPL in order to identify commonality and variability of SOA systems at the process level
- two kind decision models supporting BPL activities
  - Configuring Decision Model
  - Specializing Decision Model

The case study DAMA is ongoing and encourages further investigations in other applicative domains in order to confirm and generalize the preliminary results
... Conclusion

In order to support the application of the proposal here presented, the authors are developing two tools:

- the former aims to **automate** the decision tables management (design and consulting)
- the latter is able to **transform business process models** in executable workflows for SOA systems