

Aspect-oriented workflow patterns for web service composition

Mathieu Braem
System and Software Engineering Lab (SSEL)
Vrije Universiteit Brussel



- ▶ Web service compositions
- ▶ Plug and play compositions using the SCE
- ▶ Crosscutting concerns in compositions
- ▶ AOP with Padus
- ▶ Directions for Padus

- ▶ Web service compositions
- ▶ Plug and play compositions using the SCE
- ▶ Crosscutting concerns in compositions
- ▶ AOP with Padus
- ▶ Directions for Padus

Keywords

- ▶ AOP
- ▶ Workflows
- ▶ Webservice composition

Web service composition

- ▶ Web services expose existing software as external services, through standardized protocols
- ▶ Compositions add value by providing more advanced services
- ▶ Specialized languages to express these compositions, e.g. WS-BPEL

Service creation environment

- ▶ Create web service compositions on a higher level of abstraction
- ▶ Plug-and-play composition of building blocks in a visual editor
- ▶ Developer is guided in creating valid compositions

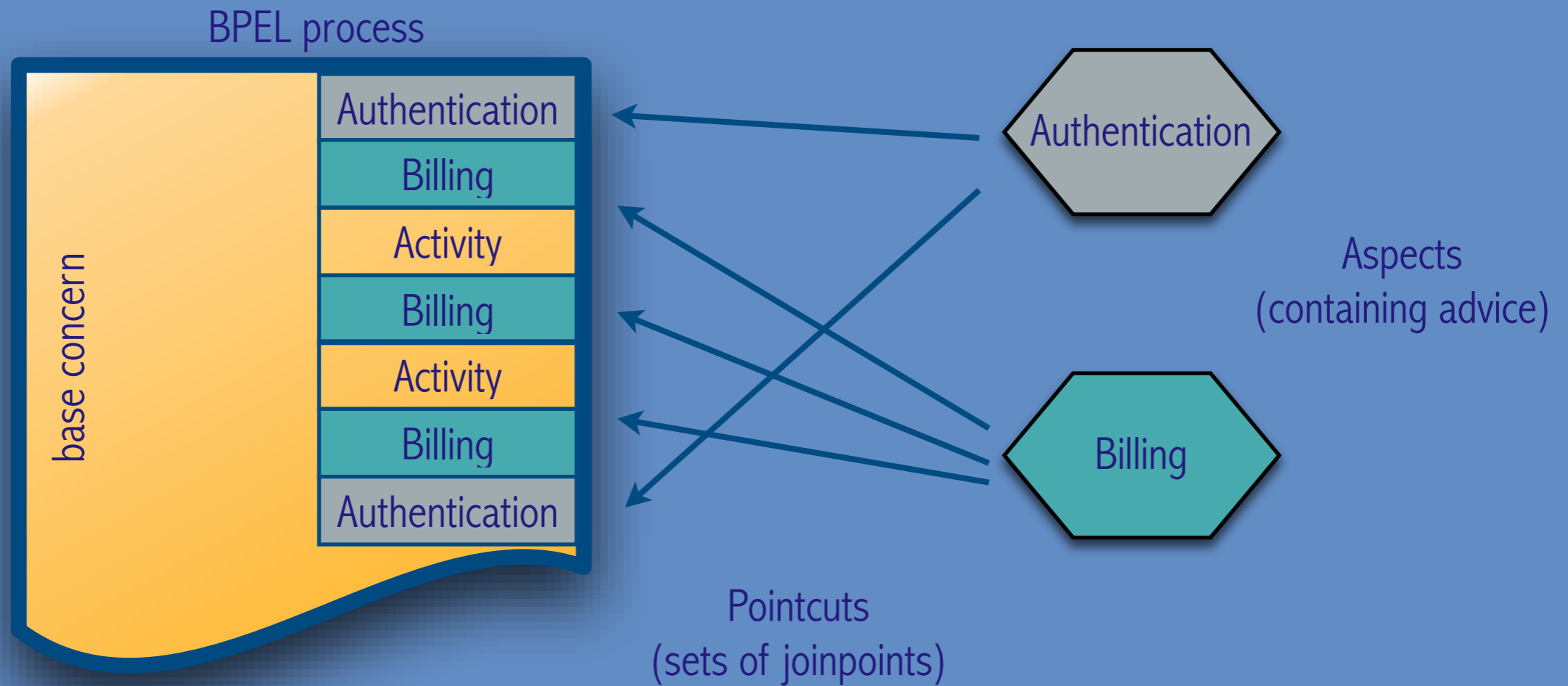
SCE Screenshot

The screenshot displays the Eclipse IDE interface for a Service Composition Editor (SCE) project named "demonstrator.composition". The interface is divided into several key areas:

- Package Explorer:** Shows the project structure, including "ScenProject".
- Palette:** A list of components available for composition, including "Select", "Marquee", "Connection", "Composition Templates", "Demonstrator", "ConferenceCall", "Services", "Directory", "Logging", "NotificationController", "Aspects", "Logging", "CSL Aspects", and "billing (time)", "billing (event)", "billing (data)".
- Canvas:** The main workspace showing a composition diagram. A purple box labeled "Demonstrator" contains two red boxes labeled "RT" (Real-time) and "Directory". A green hexagon labeled "Logging" is connected to the "Directory" component by an arrow.
- Outline:** A tree view showing the project hierarchy: "Demonstrator" (containing "<notificationcontrollerPL>" and "<whoiswhoPL>" which contains "Directory"), "Logging" (containing "<logging>").
- Properties and verification report:** A table showing the properties of the selected "Directory" component.

Property	Value
Description	Allows looking up Alcatel employees by e-mail or telephone number
Name	Directory

Crosscutting concerns



Padus aspects

```
<aspect ...>
  <using>
    <namespace ... />
    <partnerLink ... />
    <variable ... />
  </using>
  <before joinpoint="Jp"
    pointcut="invoking(Jp, 'MyService', 'MyPortType', Op)">
    <bpws:sequence>
      <bpws:assign>...</bpws:assign>
      <bpws:invoke ... />
    </bpws:sequence>
  </before>
</aspect>
```


Padus aspects

```
<aspect ...>
```

```
<using>
```

Adding namespaces, partner links, variables

```
<namespace ... />
```

```
<partnerLink ... />
```

```
<variable ... />
```

```
</using>
```

```
<before joinpoint="Jp"
```

```
  pointcut="invoking(Jp, 'MyService', 'MyPortType', Op)">
```

```
<bpws:sequence>
```

```
<bpws:assign>...</bpws:assign>
```

```
<bpws:invoke ... />
```

```
</bpws:sequence>
```

```
</before>
```

```
</aspect>
```

Padus aspects

```
<aspect ...>
  <using>
    <namespace ... />
    <partnerLink ... />
    <variable ... />
  </using>
  <before joinpoint="Jp"                                Pointcut ("where?")
    pointcut="invoking(Jp, 'MyService', 'MyPortType', Op)">
    <bpws:sequence>
      <bpws:assign>...</bpws:assign>
      <bpws:invoke ... />
    </bpws:sequence>
  </before>
</aspect>
```

Padus aspects

```
<aspect ...>
  <using>
    <namespace ... />
    <partnerLink ... />
    <variable ... />
  </using>
  <before joinpoint="Jp"
    pointcut="invoking(Jp, 'MyService', 'MyPortType', Op)">
    <bpws:sequence>                                     Advice ("what?")
      <bpws:assign>...</bpws:assign>
      <bpws:invoke ... />
    </bpws:sequence>
  </before>
</aspect>
```

Padus key concepts

- ▶ Pointcuts are logic queries
- ▶ Static weaver
Weaving = modifying logic representation of BPEL process
- ▶ Separate connector to apply aspects to processes and specify interaction resolution

Future directions

- ▶ Workflow specific advice types
- ▶ Protocol-based pointcuts
- ▶ Describe AO approach for workflow patterns

Workflow patterns

- ▶ Requirements for workflow languages, not specific to one language
- ▶ Describe control flow constructs
- ▶ Range from simple “sequence” pattern to complex patterns for synchronization

Collaboration

- ▶ Looking for collaboration on these topics
 - ▶ AOP for workflow languages, web services
 - ▶ web service composition, WS-BPEL

Collaboration

- ▶ Looking for collaboration on these topics

- ▶ AOP for workflow languages, web services

WP2/1

- ▶ web service composition, WS-BPEL

Collaboration

- ▶ Looking for collaboration on these topics

▶ AOP for workflow languages, web services **WP2/1**

- ▶ web service composition, WS-BPEL

WP1 Programming languages

WP2 Modelling languages

Collaboration

- ▶ Looking for collaboration on these topics

- ▶ AOP for workflow languages, web services **WP2/1**
- ▶ web service composition, WS-BPEL **WP2/1**

WP1 Programming languages

WP2 Modelling languages

Thank you.