Detection of Process Antipatterns: An BPEL Perspective

Francis Palma\textsuperscript{1,2}

Supervisors: Dr. Naouel Moha\textsuperscript{2} and Dr. Yann-Gaël Guéhéneuc\textsuperscript{1}

April 12, 2013

\textsuperscript{1}Ptidej Team, École Polytechnique de Montréal, Canada
\textsuperscript{2}Latece, Université du Québec à Montréal, Canada
Background

- Service Oriented Architecture (SOA)
- Service-based systems (SBSs)
- Business Process Modeling Notation (BPMN)
- Business Process Execution Language (BPEL)
- Service orchestration
- Design patterns and Antipatterns
Outline

Background

Motivation

Related Work

Approach

Experiments

Conclusion
Outline

Background

Motivation

Related Work

Approach

Experiments

Conclusion
Motivation

• Antipatterns:
  - From wrong design decisions to poor solutions

• Poor solutions:
  - bad quality of service (QoS)
  - less maintainability, evolvability etc.

• Detect antipatterns within processes

• Improve design and QoS
Why BPEL?

- BPEL processes are **off-the-rack** entities
- Antipatterns in models (BPMN) already got much attention in the literature

**Transformation errors:**
- business analysts create the processes, technical developers implement the technology
- translation, adaptation, and-or implementation errors

**Early design errors:**
- errors by analysts, eventually transferred to the process
Outline

Background

Motivation

Related Work

Approach

Experiments

Conclusion
Model Antipatterns:
- Onoda et al. (1999) catalog of five deadlock patterns
- Persson et al. (2006) and Stirna et al. (2009) provided six process patterns and 13 process antipatterns
- Koehler and Vanhatalo (2007) described 14 structural antipatterns in process models
- Trcka et al. (2009) formalized 9 process antipatterns using temporal logic
Model Antipatterns:
- Onoda et al. (1999) catalog of five deadlock patterns
- Persson et al. (2006) and Stirna et al. (2009) provided six process patterns and 13 process antipatterns
- Koehler and Vanhatalo (2007) described 14 structural antipatterns in process models
- Trcka et al. (2009) formalized 9 process antipatterns using temporal logic

Detection of Model Antipatterns:
- Gruhn and Laue (2010) proposed a heuristic-based approach for discovering problems in BPMs
- Laue and Awad (2010) visually represented process antipatterns
Model Antipatterns:
- Onoda et al. (1999) catalog of five deadlock patterns
- Persson et al. (2006) and Stirna et al. (2009) provided six process patterns and 13 process antipatterns
- Koehler and Vanhatalo (2007) described 14 structural antipatterns in process models
- Trcka et al. (2009) formalized 9 process antipatterns using temporal logic

Detection of Model Antipatterns:
- Gruhn and Laue (2010) proposed a heuristic-based approach for discovering problems in BPMs
- Laue and Awad (2010) visually represented process antipatterns

Process Patterns:
- Wohed et al. (2002) analyzed BPEL4WS based on workflow and communication patterns
- Aalst et al. (2003) discussed 26 control flow, branching-synchronization, and structural patterns
Related Work (2/2)

Identified gaps from the literature:

- **Antipatterns** and detection approaches were considered only for BPMN models
Related Work (2/2)

Identified gaps from the literature:

- **Antipatterns** and detection approaches were considered only for BPMN models

- No other **formal specifications** for process antipatterns except the one by Trcka *et al.* (2009)
Identified gaps from the literature:

- **Antipatterns** and detection approaches were considered only for BPMN models
- No other **formal specifications** for process antipatterns except the one by Trcka et al. (2009)
- Various **quality aspects** (e.g., availability or response time of Web services) were not considered
Related Work (2/2)

Identified gaps from the literature:

- **Antipatterns** and detection approaches were considered only for BPMN models

- No other **formal specifications** for process antipatterns except the one by Trcka *et al.* (2009)

- Various **quality aspects** (e.g., availability or response time of Web services) were not considered

- No **automatic detection approach** for BPEL process antipatterns until now
Outline

Background

Motivation

Related Work

Approach

Experiments

Conclusion
Solution towards detection

- We propose to:
  - specify process antipatterns using classical Rules of Inference
  - define a concrete approach

- We perform a small detection experiment
  - two antipatterns, three example BPEL processes
Step 1: Rule specification
Step 2: Process transformation
Step 3: Detection of process antipatterns
Approach (2/5): Specify Rules

**Rule:** Deadlock_Through_Decision-Join

**If:** ((Start_Node PRECEDE Decision_Node) PRECEDE Join) OR ((A_Task PRECEDE Exclusive_Gateway) PRECEDE A_Task_All_Output_Required)

**Then:** DEADLOCK
Approach (2/5): Specify Rules

**RULE: Deadlock_Through_Decision-Join**

IF: 
((Start_Node PRECEDE Decision_Node) PRECEDE Join) OR 
((A_Task PRECEDE Exclusive_Gateway) PRECEDE A_Task_All_Output_Required)

THEN: DEADLOCK

**RULE: Dangling Input_And_Output**

IF: 
((InputVar DEFINED) AND (COUNT(Connection( InputVar))==0))

THEN: DANGLING INPUT

IF: 
((OutputVar DEFINED) AND (COUNT(Connection( OutputVar))== 0))

THEN: DANGLING OUTPUT
Approach (3/5): Process Transform

- **Process transformation** (more abstract and simplified)
  
  (a) from the *original BPEL* to a *simplified BPEL*
  
  (b) from the *simplified BPEL* to a *generic model*

- **Goal** of this transformation is to **ease**:
  
  - implementation of the rules
  
  - further **analysis** of the processes
Approach (4/5): Process Transform

Background

Motivation

Related Work

Approach

Experiments

Conclusion

Francis Palma | Detection of Process Antipatterns: An BPEL Perspective
Approach (4/5): Process Transform

Background

Motivation

Related Work

Approach

Experiments

Conclusion

April 12, 2013

Francis Palma | Detection of Process Antipatterns: An BPEL Perspective

(a)

```
<bpws:invoke inputVariable="StockRequest" name="Stock_Invoke"
    operation="initiate" partnerLink="StockPartner" portType=""
    <bpws:correlations>
        <bpws:correlation initiate="yes" set="IdCorrelationSet"/>
    </bpws:correlations>
</bpws:invoke>

<bpws:receive name="Stock_Receive" operation="onResultStock"
    partnerLink="client" portType="sales:sales" variable="StockResponse"/>
```

(b)

```
<bpws:invoke inputVariable="StockRequest">
</bpws:invoke>

<bpws:receive variable="StockResponse">
</bpws:receive>
```

```
<bpws:invoke inputVariable="StockRequest">
</bpws:invoke>

<bpws:receive variable="StockResponse">
</bpws:receive>
```

```
<sending-receiving-task inputVariable="StockRequest">
</sending-receiving-task>
<receiving-task variable="StockResponse">
</receiving-task>
```
The implementation of rules
Applying implemented algorithms on transformed models
Detection is now semi-automatic
Experiments (1/5): Input

**travelProcess**
- 3 Web services
- 7 I/O

**auctionProcess**
- 3 Web services
- 6 I/O

**salesProcess**
- 2 Web services
- 4 I/O

Background

Motivation

Related Work

Approach

Experiments

Conclusion
Rules:

1. **Deadlock Through Decision-Join**
   - IF: (((Start Node PRECEDE Decision Node) PRECEDE Join) OR ((A Task PRECEDE Exclusive Gateway) PRECEDE A Task All Output Required))
   - THEN: DEADLOCK
   - (a) Deadlocks Through Decision-Join Pairs

2. **Lack Of Synchronization**
   - IF: (((COUNT(Fork) ≥ 1 AND COUNT(Merge) ≥ 1) AND (Fork PRECEDE Merge)) PRECEDE Merge)
   - THEN: LACK OF SYNCHRONIZATION
   - (b) Lack of Synchronization Through Fork-Merge Pairs

3. **Cyclic Deadlock**
   - IF: (((Join PRECEDE A Task) PRECEDE Exclusive Decision) AND (Exclusive Decision BACKCONNECT Join)) OR (((Join PRECEDE A Task) PRECEDE Fork) AND (Fork BACKCONNECT Join))
   - THEN: CYCLIC DEADLOCK
   - (c) Cyclic Deadlocks Through Join-Fork and Join-Decision Pairs

4. **Cyclic Lack of Synchronization**
   - IF: ((Fork NOT_PRECEDE Join) AND ((Fork BACKCONNECT Merge) OR (Fork BACKCONNECT Inclusive Decision)))
   - THEN: CYCLIC LACK OF SYNCHRONIZATION
   - (d) Cyclic Lack of Synchronization Through Merge-Fork Pairs

5. **Dangling Input And Output**
   - IF: ((InputVar DEFINED) AND (COUNT(Connection(InputVar))=0))
   - THEN: DANGLEING INPUT

6. **Dangling Output**
   - IF: ((OutputVar DEFINED) AND (COUNT(Connection(OutputVar))=0))
   - THEN: DANGLING OUTPUT
   - (e) Dangling Inputs and Outputs

7. **Stop-Node In Parallel Branches**
   - IF: ((Fork EXIST) AND (Each Fork-Branch HAS Stop-Node)) OR ((Inclusive Branch EXIST) AND (Each Inclusive-Branch HAS Stop-Node))
   - THEN: STOP-NODE_IN_PARALLEL_BRANCHES
   - (f) The Stop Node in Parallel Execution Branches

8. **Multiple Connections**
   - IF: ((Control-Flow EXIST BETWEEN Tasks) AND (COUNT(Control-Flow) > 1)) OR ((Data-Flow EXIST BETWEEN Tasks) AND (COUNT(Identical Data-Flow) > 1))
   - THEN: MULTIPLE CONNECTIONS
   - (g) Multiple Connections Between Activities

---

**Experiments (2/5): Rule Specification**

---

Francis Palma | Detection of Process Antipatterns: An BPEL Perspective
# Experiments (2/5): Rule Specification

## Background

- **RULE: Deadlock Through Decision-Join**
  
  IF: (((Start_Node PRECEDE Decision_Node) PRECEDE Join) OR ((A_Task PRECEDE Exclusive_Gateway) PRECEDE A_Task All_Output_Required))
  
  THEN: DEADLOCK

- **(a) Deadlocks Through Decision-Join Pairs**

- **RULE: Lack Of Synchronization**
  
  IF: (((COUNT(Fork) \geq 1 AND COUNT(Merge) \geq 1) AND (Fork PRECEDE Merge)) PRECEDE Merge)
  
  THEN: LACK OF SYNCHRONIZATION

- **(b) Lack of Synchronization Through Fork-Merge Pairs**

- **RULE: Cyclic Deadlock**
  
  IF: (((Join PRECEDE A_Task) PRECEDE Exclusive_Decision) AND (Exclusive_Decision BACKCONNECT Join)) OR (((Join PRECEDE A_Task) PRECEDE Fork) AND (Fork BACKCONNECT Join))
  
  THEN: CYCLIC DEADLOCK

- **(c) Cyclic Deadlocks Through Join-Fork and Join-Decision Pairs**

- **RULE: Cyclic Lack of Synchronization**
  
  IF: (((Fork NOT PRECEDE Join) AND ((Fork BACKCONNECT Merge) OR (Fork BACKCONNECT Inclusive_Decision))))
  
  THEN: CYCLIC LACK OF SYNCHRONIZATION

- **(d) Cyclic Lack of Synchronization Through Merge-Fork Pairs**

- **RULE: Dangling Input And Output**
  
  IF: (((InputVar DEFINED) AND (COUNT(Connection(InputVar)) = 0))
  
  THEN: DANGLING INPUT

- **(e) Dangling Inputs and Outputs**

- **RULE: Stop-Node In Parallel Branches**
  
  IF: (((Fork EXIST) AND (EACH Fork-Branch HAS Stop-Node)) OR ((Inclusive_Branch EXIST) AND (EACH Inclusive-Branch HAS Stop-Node))
  
  THEN: STOP-NODE IN PARALLEL BRANCHES

- **(f) The Stop Node in Parallel Execution Branches**

- **RULE: Multiple Connections**
  
  IF: (((Control-Flow EXIST BETWEEN Tasks) AND (COUNT(Control-Flow) > 1)) OR ((Data-Flow EXIST BETWEEN Tasks) AND (COUNT(Identical_Data-Flow) > 1))
  
  THEN: MULTIPLE CONNECTIONS

- **(g) Multiple Connections Between Activities**

## Conclusion
Experiments (3/5): Results

**Detection of Dangling Input:**
- Total data-object declared: 6
- Total data-object usage nodes: 6

**Detection of Lack of Synchronization:**
- Total Forks: 2
- Total Merge: 2
- The Lack of Synchronization exists...
  - Within nodes:
    - [parallel-gateway] followed by [receiving-task]
    - [merge] followed by [assignment]
    - [merge] followed by [merge]
Experiments (3/5): Results

Detection of Dangling Input:
Total data-object declared: 6
Total data-object usage nodes: 6

Detection of Lack of Synchronization:
Total Forks: 2
Total Merge: 2
The Lack of Synchronization exists...
Within nodes:
[parallel-gateway] followed by [receiving-task]
[merge] followed by [assignment]
[merge] followed by [merge]

<table>
<thead>
<tr>
<th>Business Process</th>
<th>Added I/O Variables</th>
<th>Added Control Flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>TravelProcess</td>
<td>name=&quot;TempAmericanAir&quot; name=&quot;TempDeltaAir&quot;</td>
<td>merge gateway with an assign task before end node</td>
</tr>
<tr>
<td>sales-bpel</td>
<td>name=&quot;notify&quot; name=&quot;subscribe&quot;</td>
<td>-</td>
</tr>
<tr>
<td>auctionProcess</td>
<td>no-changes</td>
<td>no-changes</td>
</tr>
</tbody>
</table>
Experiments (4/5): Results (cont.)

Detection of Dangling Input:

- Total data-object declared: 6
- Total data-object usage nodes: 4
- "notify" was not used...
- [ "notify" ] is a Dangling Input

- "subscribe" was not used...
- [ "subscribe" ] is a Dangling Input

Detection of Lack of Synchronization:

- Total Forks: 0
- Total Merge: 1
  No detection of Lack of Synchronization.

salesProcess
Experiments (4/5): Results (cont.)

Detection of Dangling Input:

Total data-object declared: 6
Total data-object usage nodes: 4
"notify" was not used...
[ "notify" ] is a Dangling Input

"subscribe" was not used...
[ "subscribe" ] is a Dangling Input

Detection of Lack of Synchronization:

Total Forks: 0
Total Merge: 1
No detection of Lack of Synchronization.

Detection of Dangling Input:

Total data-object declared: 9
Total data-object usage nodes: 7
"TempAmericanAir" was not used...
[ "TempAmericanAir" ] is a Dangling Input

"TempDeltaAir" was not used...
[ "TempDeltaAir" ] is a Dangling Input

Detection of Lack of Synchronization:

Total Forks: 1
Total Merge: 2
The Lack of Synchronization exists...
Within nodes:
[parallel-gateway] followed by [sequence-flow]
[merge] followed by [assignment]
[merge] followed by [assignment]
Experiments (5/5): Threats to Validity

**External validity:** Possibility to generalize the results for other large and realistic business processes

**Construct validity:** Different engineers might define rules differently
Conclusion

Approach (2/5)

1. Rule specification
2. Process transformation
3. Detection of process antipatterns

Step 1: Rule specification
Step 2: Process transformation
Step 3: Detection of process antipatterns
Conclusion

Background

Motivation

Related Work

Approach

Experiments

Conclusion
Conclusion

Approach (2/5)

Experiments (2/4): Rule Specification

Experiments (1/4): Input

Background

Motivation

Related Work

Approach

Experiments

Conclusion

Francis Palma | Detection of Process Antipatterns: An BPEL Perspective
Conclusion
Future work

• Automate the approach
• Detect more process antipatterns
• Perform experiments on other large and complex business processes
• Analyze the processes dynamically
Thanks for your attention! Questions?