Calculating BPEL Test Coverage through Instrumentation

Daniel Lübke, Leif Singer, Alex Salnikow

Software Engineering Group, Leibniz Universität Hannover
leif.singer@inf.uni-hannover.de

May 19th 2009
Agenda

• BPEL

• BPELUnit

• Coverage Metrics for BPEL

• BPEL Instrumentation

• Case Study

• Conclusions
BPEL
BPEL

- Business Process Execution Language
  - for service orchestration

- Specifies control flow & data transformations
  - client communication
  - service calls

- XML-based
  - BPEL itself
  - WSDL
  - XML Schema
  - XPath
  - XSL(T)
# BPEL

- **block-based and flow-based**
  - created from two competing languages

- **handlers for errors & compensations**
  - supporting *business* transactions

- **gets deployed to a BPEL-Engine**
  - executes the process
  - correlates instances
  - ...
  - also, many vendor-specific additions

<table>
<thead>
<tr>
<th>process</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence</td>
</tr>
<tr>
<td>receive</td>
</tr>
<tr>
<td>repeatUntil</td>
</tr>
<tr>
<td>invoke</td>
</tr>
<tr>
<td>assign</td>
</tr>
<tr>
<td>reply</td>
</tr>
</tbody>
</table>

**blocks**

**flow**
BPELUnit

- Ant Task & Eclipse-Plugin
  - similar to JUnit
  - transparent deployment of PUT to engine for supported engines
  - manual / self-scripted deployment otherwise

![Diagram of BPELUnit](image_url)
BPELUnit

• client and partner tracks

• client: caller of process

• partners are Web Services
  – might not be available for testing
  – Mock Services: specify their output in test suite

• assertions: XPath expressions on the messages
  – e.g., //ns1:customer[@id] == 3
Coverage Metrics for BPEL
Coverage Metrics

• How *good* are my tests?
  – (... for a definition of *good*)

• coverage metrics as a common indicator for test quality

• Statement Coverage, Branch Coverage, Path Coverage, ...

• might be useful for BPEL, too
Coverage Metrics

• instrument the PUT before deployment to the engine

• use a logging Web Service to record execution paths

• need coverage metrics first
  – defined by the paper presented here
  – based on traditional metrics
  – with BPEL-specific additions
Activity Coverage

• ~ Statement Coverage

• basic BPEL activities
  – the atomic ones

\[
\text{basicActivities}_{\text{executed}} / \text{basicActivities}_{\text{overall}}
\]

• simple to implement: just add sequence element with logger

• essential, but weak coverage: no control flow
Branch Coverage

• ~ Branch Coverage

\#branches_{executed} / \#branches_{overall}

• minor instrumentation challenges
  – implicit else-blocks
  – loops
  – ...
Link Coverage

- measures coverage of transition conditions
  - just the variable ones

- 100% if all links with variable transition conditions have evaluated to true and false at least once

\[
\left( \#t_{c_{\text{true}}} + \#t_{c_{\text{false}}} / 2 * t_{c_{\text{variable}}} \right)
\]
Handler Coverage

• two sub-coverages, counting executed handlers
  – similar to Activity Coverage, but more specific

• Fault Handler Coverage
  – might be Exception Coverage in Java

• Compensation Handler Coverage
  – business transactions

• both are important tools
  – errors & transactions may have huge impact on business
BPEL Instrumentation
Instrumentation

- maintains process semantics using a few tricks
- e.g., Link Coverage
  - no boundary crossings allowed in BPEL

```
flow
source activity
cond

target activity

flow
source activity
not cond
cond
logger

flow
logger
cond

flow
target activity
```
Case Study
Case Study

- student project with two test suites
  - macro flow and micro flows

<table>
<thead>
<tr>
<th></th>
<th>macro flow</th>
<th></th>
<th>micro flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Coverage</td>
<td>62%</td>
<td>Activity Coverage</td>
<td>50%</td>
</tr>
<tr>
<td>Branch Coverage</td>
<td>69%</td>
<td>Branch Coverage</td>
<td>55%</td>
</tr>
<tr>
<td>Link Coverage</td>
<td>75%</td>
<td>Link Coverage</td>
<td>25%</td>
</tr>
<tr>
<td>Handler Coverage</td>
<td>0%</td>
<td>Handler Coverage</td>
<td>0%</td>
</tr>
</tbody>
</table>

→ coverage metrics would have uncovered incomplete tests
Conclusions
Benefits

- tester / QA may assess test quality
- directed test improvements
- integration into build process possible
  - e.g. CruiseControl: use Ant task to reject processes with low coverage
- better tests $\Rightarrow$ better processes $\Rightarrow$ less losses
Integration

• recent effort to provide an integrated tool suite

  • Eclipse BPEL Designer
    – project lead: IBM

  • ODE BPEL Engine
    – Apache Software Foundation

  • BPELUnit
    – Daniel Lübke / Leif Singer

• Google Summer of Code project underway
  – BPELUnit + ODE
  – deployment, coverage
Next Steps

• basic metrics available now with preliminary tool support
  – better integration on the way

• test case generation for BPEL processes
  – Universität Rostock

• stress testing

• other coverages possible
  – not our focus
Thank you!

- BPEL
- BPELUnit
- Coverage Metrics for BPEL
- BPEL Instrumentation
- Case Study
- Conclusions

Thank you for your attention!

➡ Software Engineering Group: http://www.se.uni-hannover.de
➡ BPELUnit: http://bpelunit.net