

# Calculating BPEL Test Coverage through Instrumentation

Daniel Lübke, **Leif Singer**, Alex Salnikow

Software Engineering Group, Leibniz Universität Hannover

[leif.singer@inf.uni-hannover.de](mailto:leif.singer@inf.uni-hannover.de)

May 19th 2009

# Agenda

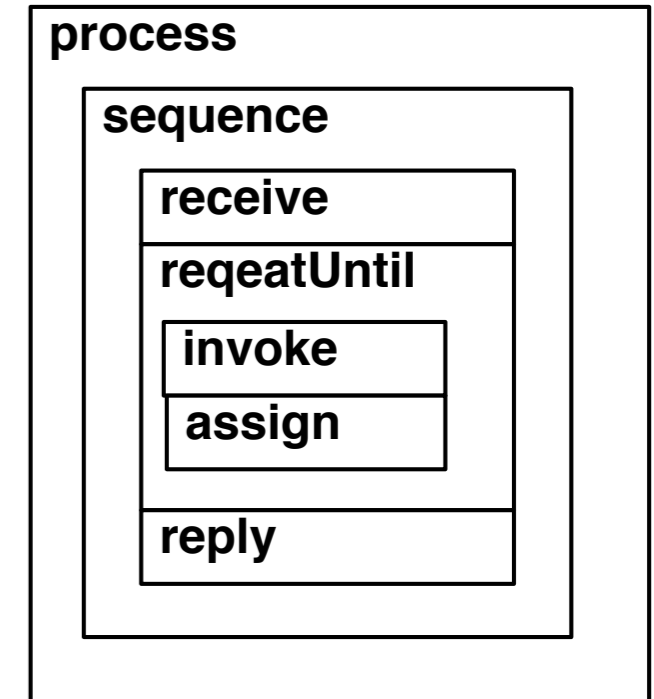
- BPEL
- BPELUnit
- Coverage Metrics for BPEL
- BPEL Instrumentation
- Case Study
- Conclusions

# BPEL

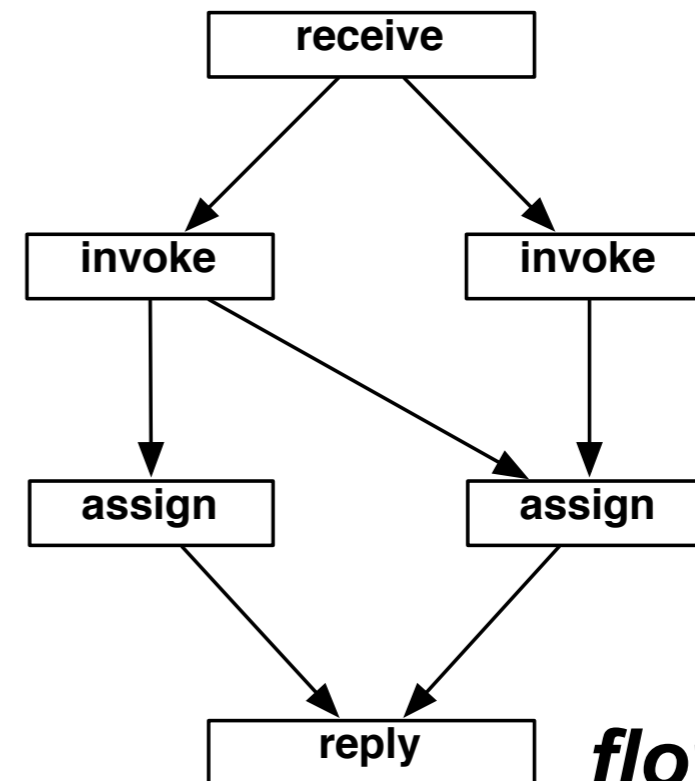
- Business Process Execution Language
  - for service orchestration
- Specifies control flow & data transformations
  - client communication
  - service calls
- XML-based
  - BPEL itself
  - WSDL
  - XML Schema
  - XPath
  - XSL(T)

# 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



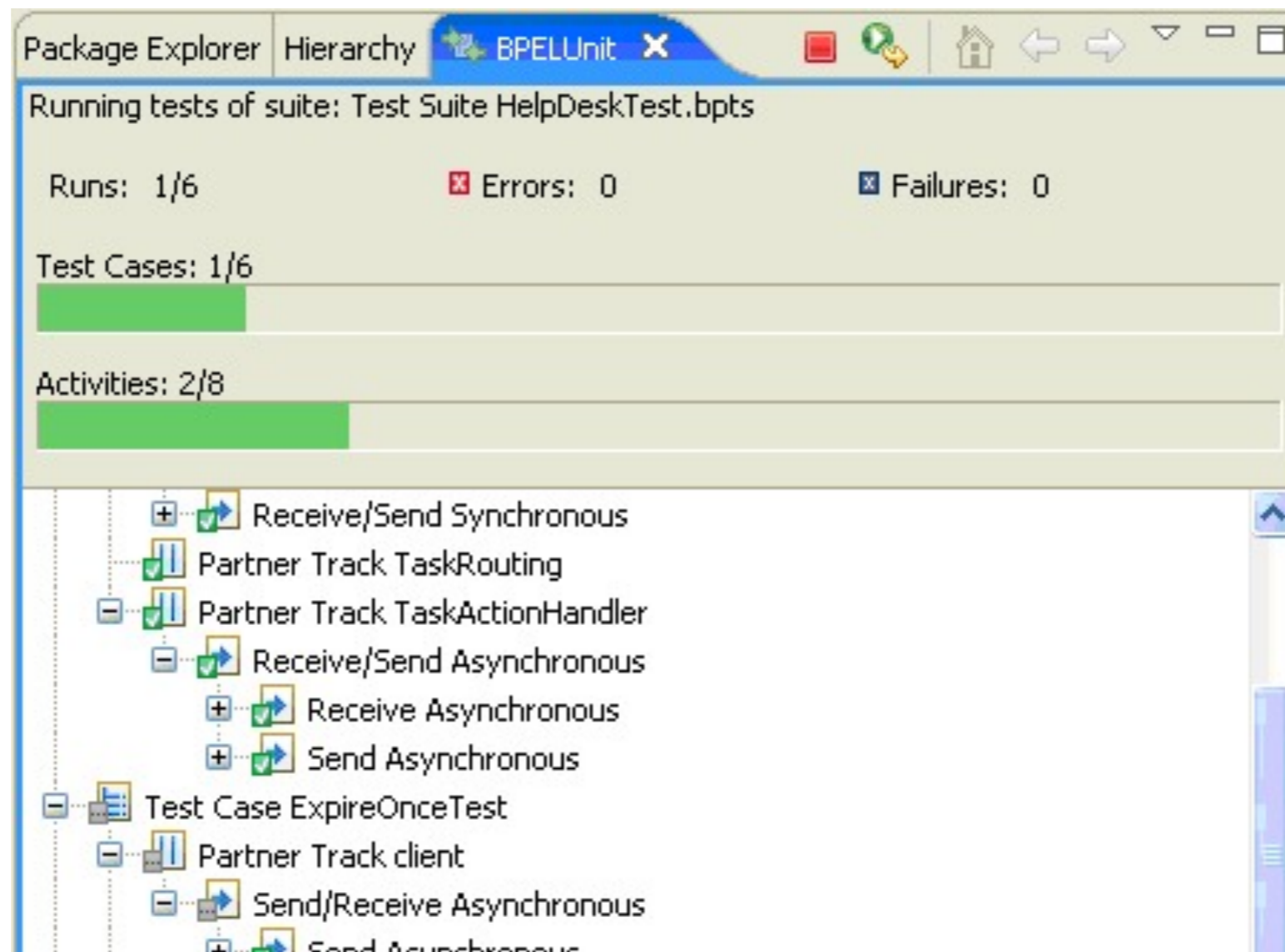
**blocks**



**flow**

# BPELUnit

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



- 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

**basicActivities<sub>executed</sub> / basicActivities<sub>overall</sub>**

- simple to implement: just add sequence element with logger
- essential, but weak coverage: no control flow

# Branch Coverage

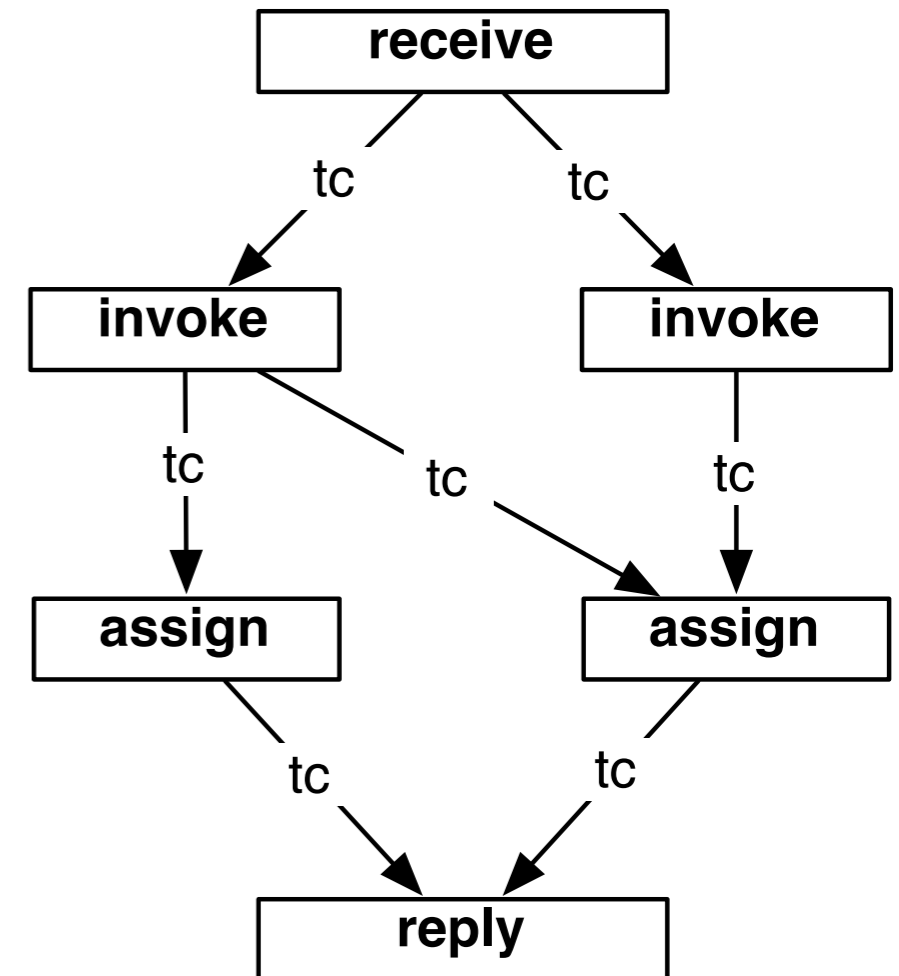
- ~ Branch Coverage

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

- 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



$$\left( \#tc_{\text{true}} + \#tc_{\text{false}} / 2 * tc_{\text{variable}} \right)$$

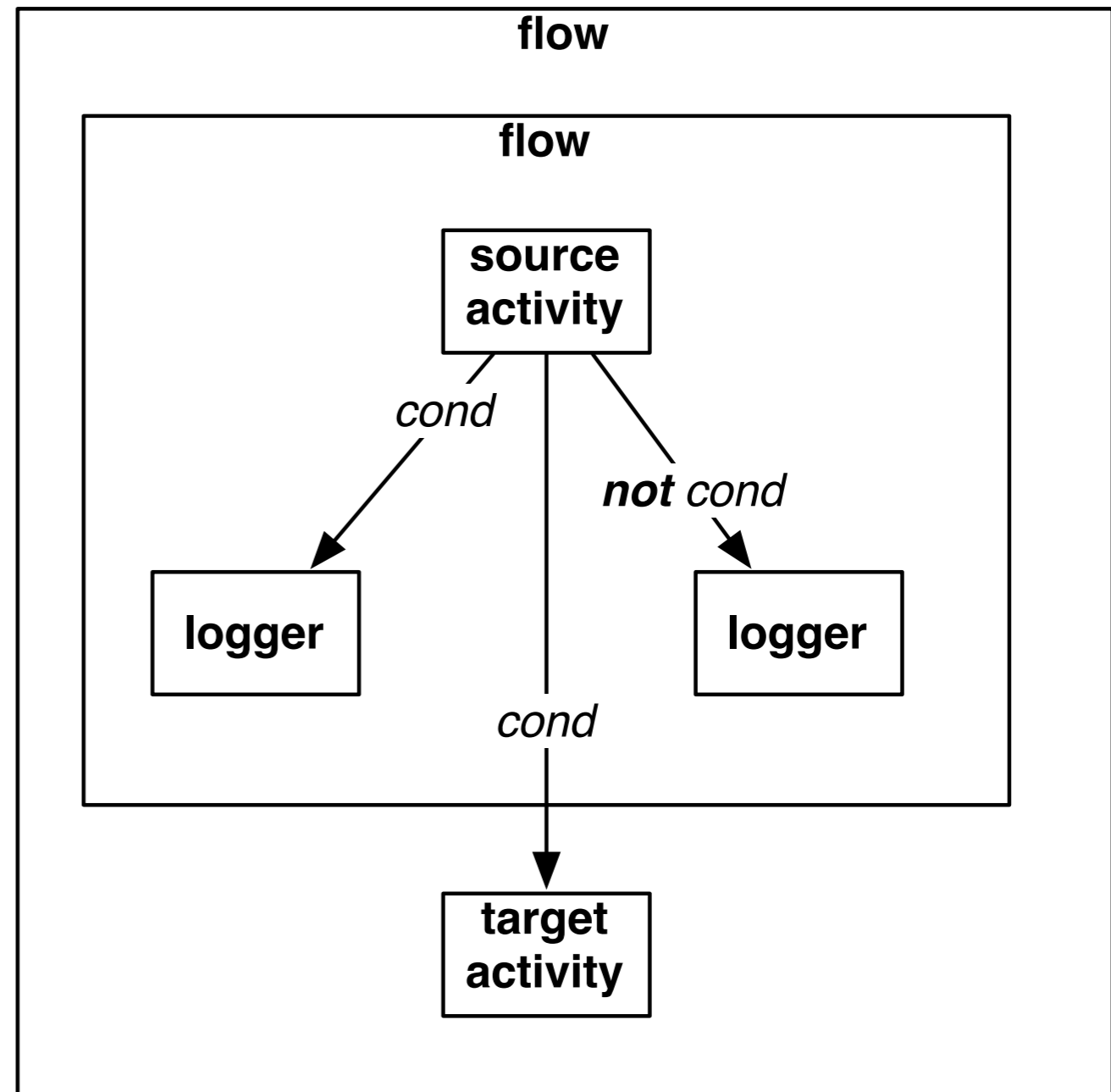
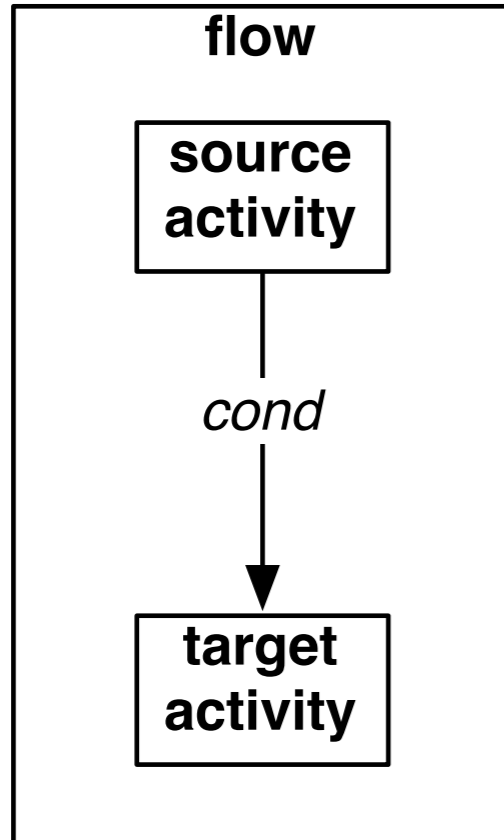
# Handler Coverage

- 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



# Case Study

# Case Study

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

## macro flow

Activity Coverage	<b>62%</b>
Branch Coverage	<b>69%</b>
Link Coverage	<b>75%</b>
Handler Coverage	<b>0%</b>

## micro flows

Activity Coverage	<b>50%</b>
Branch Coverage	<b>55%</b>
Link Coverage	<b>25%</b>
Handler Coverage	<b>0%</b>

➔ coverage metrics would have uncovered incomplete tests

# Conclusions

- 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 ➡ better processes ➡ less losses

- 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>