

ORACLE[®]

Performance Tests in der Praxis am Beispiel von Oracle BPM/BPEL

Alexander Fox <alexander.fox@oracle.com>
Senior Berater

Agenda

Dauer ca. 45 Minuten

- Begriffsbestimmung
- Software-Tests – Eine Übersicht
- Durchführung von Performance Tests
- Tuning-Massnahmen
- Performance-Test Planung
- Beschreibung eines Performance-Tests mit Oracle BPEL
 - Embedded Java
 - Request-Handler
 - Sensor
 - Mediator

Meinungen zu Performance und Optimierung

„Tuning is part of development that starts before the first line of code is ever written and ends the day before deployment“

– Tom Kyte, Expert Oracle (2005)

„... about 97% of the time: premature optimization is the root of all evil“

– Donald E. Knuth, Structured programming with go to statements (1979)

Performance-Tuning Maßnahmen

Synchronität

Mehr
Infrastruktur

Änderung in
Anwendungslogik

Caching

Asynchrone
Verfahren

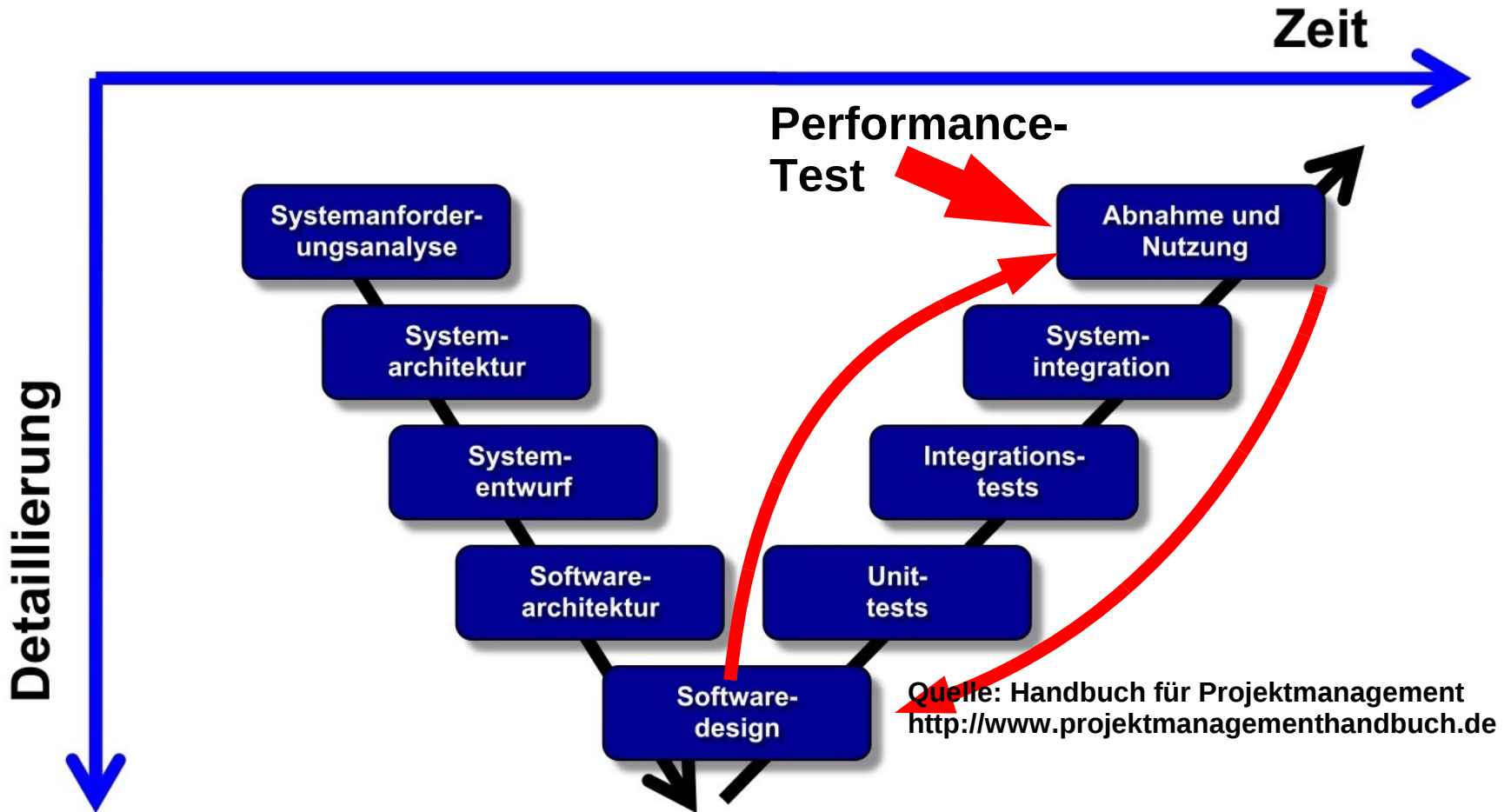
Meist Read-Only,
Bzw. Write-
Through

Skalierbarkeit?

Fehler-
behandlung

Kontrollfluss-
ablauf

Einordnung Performance- Tests in das V-Modell



Klassifizierung Tests

Fehlertests

Komponenten-
tests

Interoperabilitäts-
tests

Schnittstellen-
tests

Lasttest

Oberflächen-
tests

Datenkonsistenz-
tests

Stress-Test

Netzwerk-
tests

Sicherheits-
tests

Crash-Test

Performance-Planung

- Datenbasis (Echt-, Synthetische, Obfuskierte Daten)
- Benutzer-Verhalten (Normale Nutzung, Ausnahmenutzung zu Stichtagen)
- Werkzeuge zur Lasterzeugung (Verteilte Lasterzeugung)
- Zugriffsschicht
 - GUI
 - Service Schicht
 - API (Java)
 - Datenbank
- System-Verfügbarkeit

Performance-Planung

- Software-Stand
- Test-UNDO: Rücksetzung, Löschung
- Zu sammelnde Metriken (verbrauchte Bandbreite, Transactionen pro Sekunde, Payload size, Throughput, # concurrent User)
- Integration von Performance-Tests in den Continuous-Integration-Infrastruktur

Automatische Code-Instrumentierung

- Die Erstellung von Performance-Metriken sind sog. Querschnittsfunktion „Cross-Cutting-Concern“ der Software
- Performance-Metriken haben funktional keine Bedeutung
- Die Software zur Erstellung dieser Metriken sollte extern gepflegt und erstellt werden

Infrastruktur für Statistik Logging

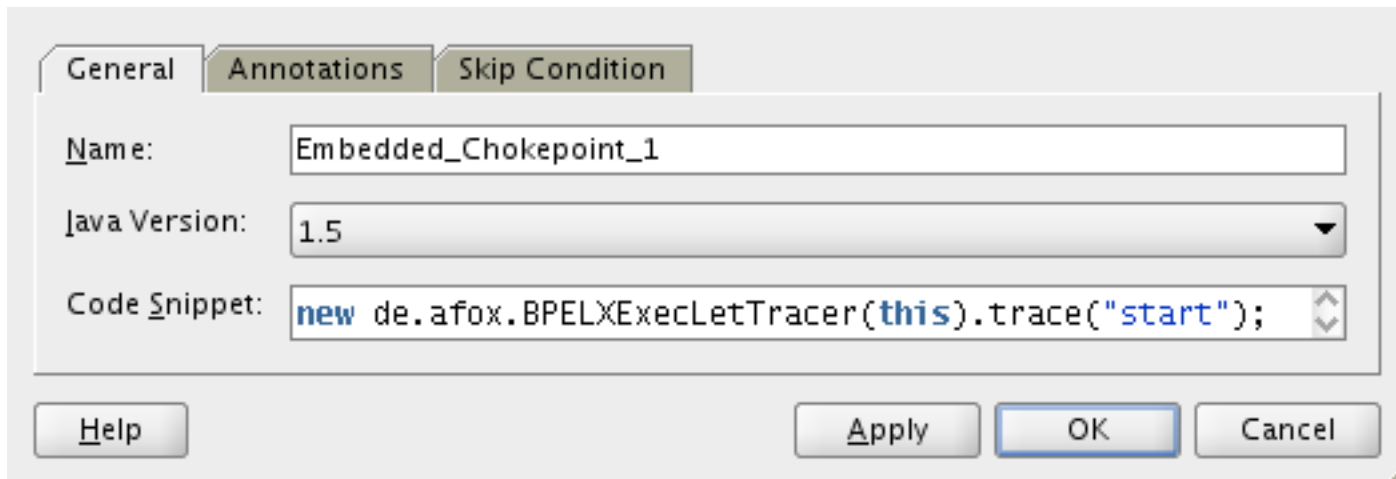
- Einfaches Logging für Statistik Logs
 - Log-Name
 - Log-ID
 - Timestamp
- Logging über JDBC-DataSource
- Aggregation der Log-Daten
- Beispiel Treiber
 - Request Senden
 - Log auswerten
 - Unittest-Integration

Code-Instrumentierung über Embedded Java

- Erweitern der Klasse und Implementierung eines Adapters/Wrappers für `com.collaxa.cube.engine.ext.bpel.v1.nodes.BPELXExecLet`
- Überschreiben dieser Methoden und Aufruf der privaten Methoden der eingehüllten Klasse
 - `public long getInstanceId()`
 - `public ICubeInstance getCubeInstance()`
 - `public Object getVariableData(String name)`
 - `public String getParentId()`
 - `public String getTitle()`
 - `public void execute()`
- Minimaler Code-Ausführung in embedded Java

Code-Instrumentierung über Embedded Java

```
<bpelx:exec name="Embedded_Chokepoint_1" version="1.5"
language="java">
  <![CDATA[
    new de.afox.BPELXExecLetTracer(this).trace("start");
  ]]></bpelx:exec>
```



The screenshot shows a configuration dialog box with three tabs: "General", "Annotations", and "Skip Condition". The "General" tab is selected. The dialog contains the following fields:

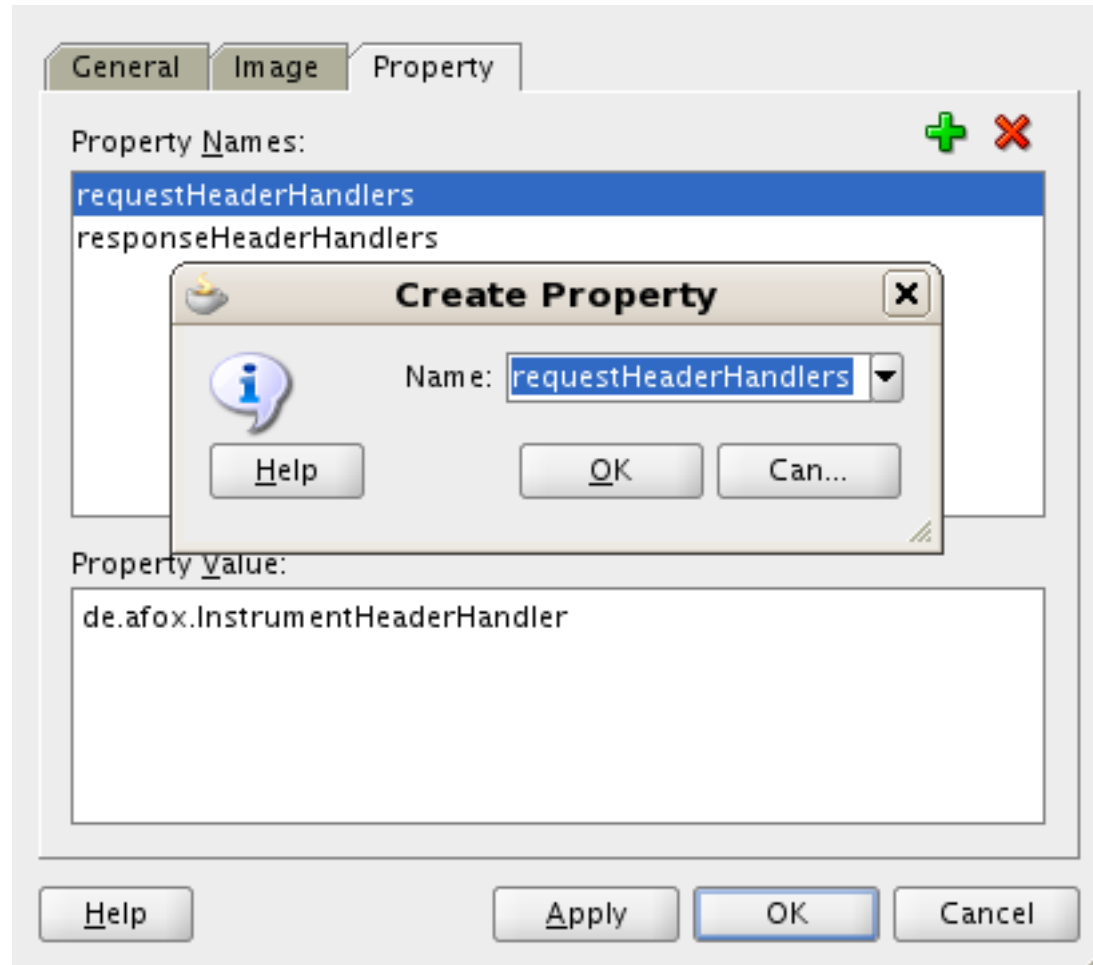
- Name:** Embedded_Chokepoint_1
- Java Version:** 1.5
- Code Snippet:** new de.afox.BPELXExecLetTracer(this).trace("start");

At the bottom of the dialog, there are four buttons: "Help", "Apply", "OK", and "Cancel".

Code-Instrumentierung über Partner-Link Request Handler

- Implementierung der Klasse `com.collaxa.cube.ws.HeaderHandler`
- Überschreiben der Funktion
 - `invoke(CXPartnerLink pPartnerLink, String pOperationName, Map pMsgPayload, List pMsgHeader, Map pCallProps)`
- Protokollierung von `(partnerLink.getName+"/"+operationName, partnerLink.getConversationId())`
- Registrierung des RequestHandlers unter PartnerLink-Eigenschaften

Code-Instrumentierung über Partner-Link Request Handler



Code-Instrumentierung über Partner-Link Request Handler

composite.xml:

```
<component name="StockProcess">
<implementation.bpel src="StockProcess.bpel"/>
  <property
    name=
      "partnerLink.StockQuoteService.requestHeaderHandlers"
    type="xs:string"
    many="false">de.afox.InstrumentHeaderHandler
  </property>
  <property
    name=
      "partnerLink.StockQuoteService.responseHeaderHandlers"
    type="xs:string"
    many="false">de.afox.InstrumentHeaderHandler
  </property>
  ...
```

Code-Instrumentierung über Sensors

- Java Interface implementieren:
`com.oracle.bpel.sensor.DataPublisher`
- Methode `publish`
 - `public void publish(ITSensorAction action, ITSensorActionData actionData, Element xml)`
- Protokollierung: `(header.getProcessName() + "/" + action.getName(), header.getCompositeInstanceId())`
- Sensor registrieren

Code-Instrumentierung über Sensors

The screenshot displays the Oracle BPEL Designer interface. The main workspace shows a BPEL process diagram with a 'receiveInput' activity. A red circle highlights the 'Monitor' button in the top toolbar. Three dialog boxes are open:

- Activity Sensors:** A table listing the configured sensors.
- Edit Activity Sensor - ActivitySensor_1:** Shows the configuration for the selected sensor.
- Edit Sensor Action - SensorAction_1:** Shows the configuration for the sensor's action.

Name	Type	Sensor Actions	Class Name	Target
ActivitySensor_1	Activity	SensorAction_1	oracle.tip.pc.servi...	receiveInput

Edit Activity Sensor - ActivitySensor_1

Name: ActivitySensor_1
Activity Name: receiveInput
Evaluation Time: All

Variable	XPath	Output Namespace
----------	-------	------------------

Sensor Actions:

- SensorAction_1

Edit Sensor Action - SensorAction_1

Name: SensorAction_1
Publish Type: Custom
JMS Connection Factory:
Publish Target: de.afox.SensorTracer
Filter:
 Enable

Code-Instrumentierung über Sensors

Process_sensorAction.xml:

```
<actions targetNamespace="...">  
  <action name="SensorAction_1" publishName=""  
publishType="Custom" enabled="true" filter=""  
publishTarget="de.afox.SensorTracer">  
    <sensorName>ActivitySensor_1</sensorName>  
  ...
```

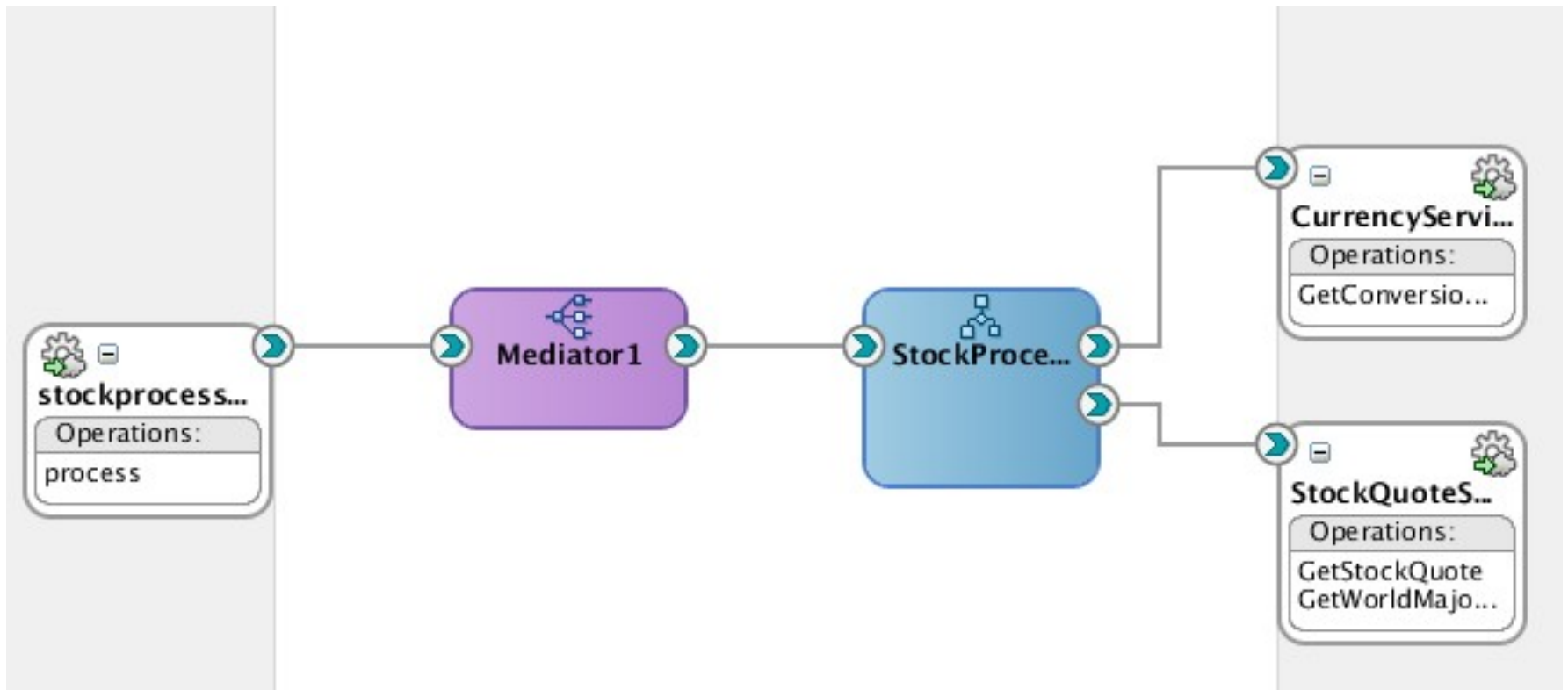
Process_sensor.xml:

```
<sensors targetNamespace="...">  
  <sensor sensorName="ActivitySensor_1"  
classname="oracle.tip.pc.services.reports.dca.agents.BpelActivitySensorAg  
ent" kind="activity" target="receiveInput">  
    <activityConfig evalTime="all"/>  
  </sensor> ...
```

Code-Instrumentierung über Mediator

- Mit einem Mediator lassen sich Proxy-Services für BPEL-Services erstellen
- Mediatoren können über einen Java-Callout erweitert werden:
oracle.tip.mediator.common.api.AbstractJavaCalloutImpl
- Überschreiben der Methoden:
 - public boolean preRouting(CalloutMediatorMessage msg)
 - public boolean postRouting(CalloutMediatorMessage msg1, CalloutMediatorMessage msg2, Throwable t)
- Protokollierung von (msg1.getComponentDN(), , msg1.getId())

Code-Instrumentierung über Mediator



Code-Instrumentierung über Mediator

Routing Rules

Operations

process

Priority 4

Validate Syntax (XSD)



Callout To



Static Routing



Sequential

Validate Semantic



Transform Using



Assign Values



Synchronous Reply



Transform Using



Assign Values



Code-Instrumentierung über Mediator

Mediator.mplan:

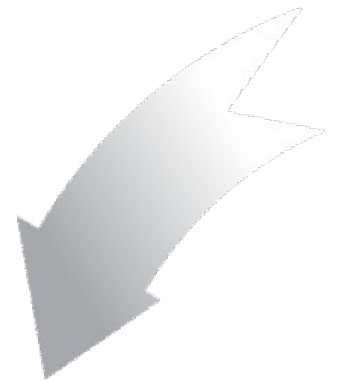
```
<Mediator name="Mediator1" ...>  
  <operation name="process" ...>  
    <customFunction>  
      <javaFunction>de.afox.TracingMediatorJavaCallout</javaFunction>  
    </customFunction>  
    <switch>  
      <case executionType="direct"  
        name="StockProcess.stockprocess_client.process">
```

Composite.xml:

```
<component name="Mediator1">  
  <implementation.mediator src="Mediator1.mplan"/>  
</component>
```

Bibliographie

- The Art of Application Performance Testing, Ian Molyneaux, O'Reilly 2009
- Oracle SOA Suite 11g R1 Developers Guide, A.Reynolds, M.Wright, Packt, 2010
- Oracle® Fusion Middleware Developer's Guide for Oracle SOA Suite 11g Release 1 ([11.1.1](#))



Q & A