Performance Tests in der Praxis am Beispiel von Oracle BPEL

Alexander Fox
Oracle Deutschland B.V. & Co. KG
Frankfurt

Schlüsselworte:

Oracle BPEL BPM Performance Test

Einleitung

Agilität gewinnt in der Software-Entwicklung einen immer größer werdenden Stellenwert. Es ist mittlerweile in der Software-Entwicklergemeinschaft allgemein bekannt, wie Komponententests (Unit-Tests) zur Agilität der Software-Entwicklung beitragen können. Performance-Tests gewinnen zu einem späten Zeitpunkt des Software-Entwicklungsprozesses an Interesse, sind aber nicht weniger wichtig für die Agilität von Software-Projekten. Trotzdem führen sie meist ein Schattendasein.

Im Vortrag "Performance Tests in der Praxis am Beispiel des Oracle Business Process Managers (BPM)" wird ein genereller Überblick über das Thema Software-Performance-Tests gegeben. Die generellen Problemstellungen und Lösungen bei der Durchführung von Performance-Tests werden am Beispiel des Oracle Business Process Managers (BPM) aufgezeigt. Performance Tests werden als Bestandteil des Software-Entwicklungsprozesses dargestellt und es wird erörtert, wie sie die Software-Entwicklung unterstützen können.

Einführung in das Thema Performance-Tests

Performance-Tests stehen meist im Software-Entwicklungsprozess am Ende. Oftmals finden Performance-Tests erst nach Produktivstart statt. Das führt dazu, dass in vielen Projekten der Produktivbetrieb anfangs gestört sein kann. Ein derart ungetestetes unter starken Störungen leiden , die bis zum Totalausfall führen können. Die finanziellen Kosten, die mit einem ungetesteten System verbunden sind, wurden in keiner Schätzung berücksichtigt und sind in ihrer Höhe nur schwer abschätzbar. Hinzu kommt der Verlust an Image und Akzeptanz.

Im V-Modell der Softwareentwicklung gehört die Planung für Performance-Tests schon in die Phase der Anforderungsanalyse. Dort sollten schon erste Werte zu maximalen Antwortzeiten, Anzahl gleichzeitiger Benutzer spezifiziert werden. Performance-Tests sollte sich von der Architektur-Phase bis in die Abnahme- und Produktiv-Phase erstrecken. Während der eigentlichen Implementierungsphase geben die Performance-Tests wertvolle Hinweise auf die Effektivität von Performance-Tuning-Massnahmen, anhand welcher man die Entwickler die Notwendigkeit und den Erfolg solcher Maßnahmen belegen können.

Auch nach Produktivstart leisten die in der Entwicklungsphase erstellten Performance-Tests noch wertvolle Dienste. Die im Gesamtsystem verwendeten Software-Komponenten unterliegen ihrerseits wieder einem Software-Development-Lifecycle und der unabhängige Upgrade aller Komponenten ist im Produktivbetrieb angefangen von der Datenbank bis zum Application Server und Client-Implementierungen an der Tagesordnung. Performance-Tests erhöhen nicht nur die Stabilität und das

Vertrauen in die gelieferte Version, sondern helfen den Patch-Lifecycle verkürzen und agiler zu gestalten, was vor allem im Hinblick auf IT-Sicherheit Vorteile bringt, da Systeme auf neuerem Patch-Stand laufen können.

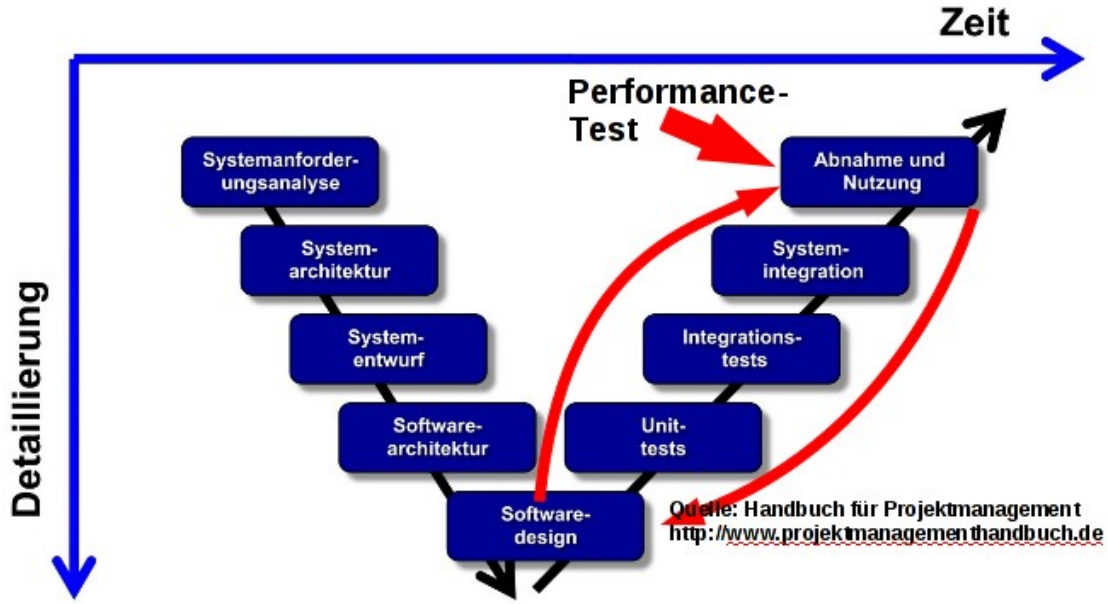

Abb. 1: Performance-Tests im V-Modell

Bei Performance-Tests unterscheidet man zwischen Lasttests, die das System im „normalen“ Bereich testen. Weiterhin gibt es Stress-Tests, die auf das System eine Last bringen, die in einem nicht mehr normalen Bereich befindet, währen Crash-Tests das Ziel haben das System einer nicht mehr zu bewältigenden Last auszusetzen.

Im BPEL-Bereich werden häufig asynchrone Prozesse bei der Verarbeitung favorisiert. Das bedeutet aber nicht, dass es nicht auch hier zu Performance-Problemen kommen kann. Es ist durchaus möglich das die maximale Verarbeitungszeit eines asynchronen Prozesses in einem Performance-Test zu spezifizieren, der dann die Erwartung des Benutzers widerspiegelt, bis wann ein asynchroner Request verarbeitet werden sollte. Z.B. rechnet man beim Versandt einer E-Mail heute eine maximale Zustellzeit von einigen Sekunden bis zu einer halben Stunde.

Planung von Performance-Tests

Bei der Planung von Performance-Tests steht im Vordergrund Tests durchzuführen, die sich auch an den Realen Gegebenheiten orientieren. Mehrere Faktoren arbeiten aber gegen eine „realistisches“ Szenario. Zuallererst ist die Verfügbarkeit von Testern zu nennen. In einer komplexen Enterprise-Anwendung ist es fast unmöglich und unbezahlbar alle Benutzer zur Durchführung von Performance-Tests „zu reservieren“. Oftmals können aufgrund von gesetzlichen Gegebenheiten zum Schutz personenbezogener Daten, keine Echtdaten für die Tests eingesetzt werden. Daraus ergibt sich, dass jeder Tests in bestimmten Aspekten nicht die reale Systemnutzung widerspiegelt. Es müssen Vereinfachungen vorgenommen und technische Implementierungsentscheidungen getroffen werden.

- Datenbasis (Echtdaten, Synthetische Daten, Obfuskierte Datenbasis)
- Simulation des Benutzerverhaltens (normale Nutzung, Ausnahmenutzung zu Stichtagen etc.)

- Lasterzeugungswerkzeuge und System-Architektur
- Zugriffs-Schicht für Lasttest (GUI, API-Layer, Service-Layer, DB-Layer)
- Verfügbarkeit des Testsystems
- Software-Stand
- Test-UNDO: Rücksetzung und Wiederherstellung
- Zu sammelnde Metriken (Bandbreite, Transaktionen pro Sekunde, Payload-Size, Anzahl konkurrierender Benutzer)
- Integration der Performance-Tests in den Continuous-Integration-Lifecycle

Automatische Code-Instrumentierung

Im folgenden wird speziell auf die Problematik der Statistiksammlung eingegangen. Die Sammlung von Performance-Statistiken ist, ähnlich wie IT-Sicherheit, eine Querschnittsfunktion (Cross-Cutting-Concern) in einem Software-Projekt. Performance-Statistiken haben funktional keine Bedeutung. Idealerweise sollten die Software zu Sammlung der Statistikdaten und Metriken unabhängig von der Software-Entwicklung entwickelt und gepflegt werden. Wichtig ist, dass die Sammlung von Statistikdaten automatisch eingebunden werden kann und evtl. auch wieder entfernt werden kann. In Oracle BPEL gibt es verschiedene Funktionalitäten, die zum bei der Erzeugung der Performance-Statistiken unterstützen können.

Partner-Link-Request-Handler
Eine Funktion von Oracle BPEL ist die Orchestrierung verschiedener Services zu einem Composite-Service. Die Antwortzeit/Verarbeitungszeit von Composite-Sevices hängt von den Einzelkomponenten des Composite-Services ab. Als Bestandteil eines Composite-Services kann z.B. ein eigener Web-Service dienen, aber auch ein Service eines externen Dienstleisters. Im laufenden Betrieb lassen sich externe Dienste nur schwer überwachen.

Der Partner-Link-Request-Handler ist ein Proxy der bei jedem Aufruf des verbunden Services involviert ist. Der Partner-Link-Request-Handler ist ursprünglich dazu gedacht Daten zum Request hinzuzufügen, bzw. den Request zu ändern. Über einen Partner-Link-Request-Handler lassen sich aber auch mit nur minimalen Änderungen am Business-Flow, Metriken über die laufende Anwendung und deren Verarbeitungszeiten sammeln.

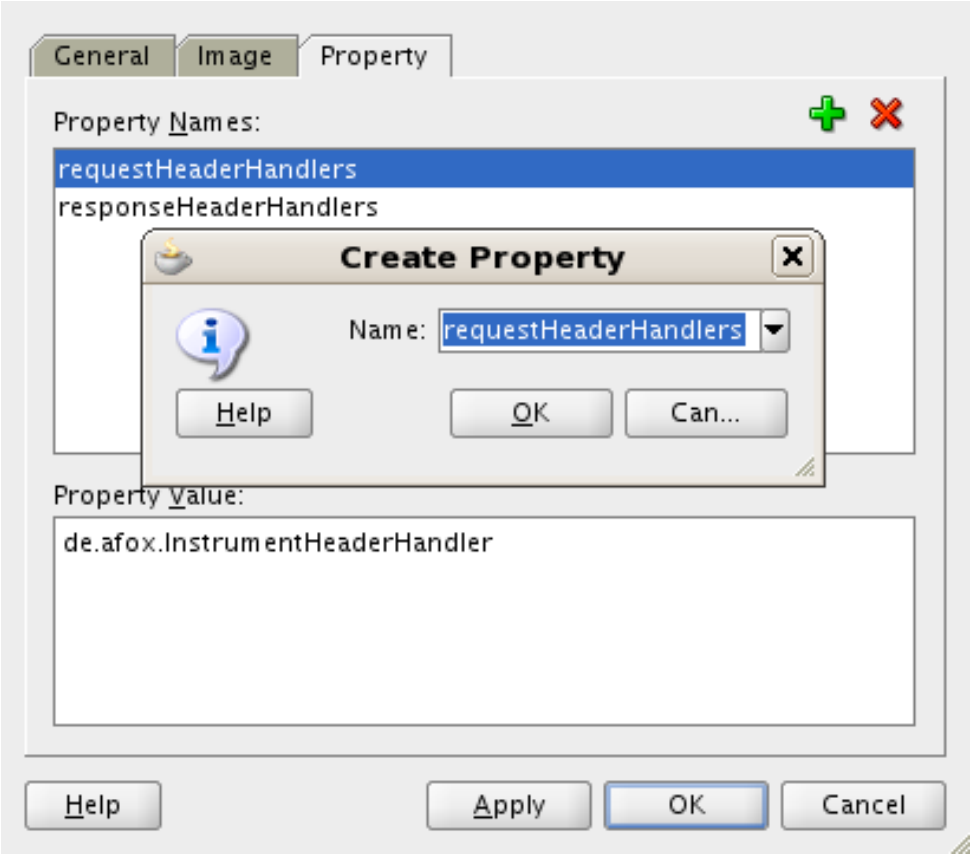

Abb. 2: Code-Instrumentierung über InstrumentationHeaderHandler

Embedded Java

Oracle BPEL bietet die Möglichkeit in den BPEL-Ablauf Java-Code einzubetten. Der Java-Code kann an beliebigen Stellen und Zeitpunkten im Business-Prozess eingefügt werden. In einem Java-Fragment können die gewünschten statistischen Daten gesammelt werden. Die Methode der Sammlung von statistischen Daten über Embedded Java ist nicht transparent, d.h. der Business-Flow wird bei dieser Methode um Embedded Java erweitert. Ein Vorteil von Embbedded Java ist, dass er, anders als der Partner-Link-Request-Handler an beliebiger Stelle im Business-Flow platziert werden kann.

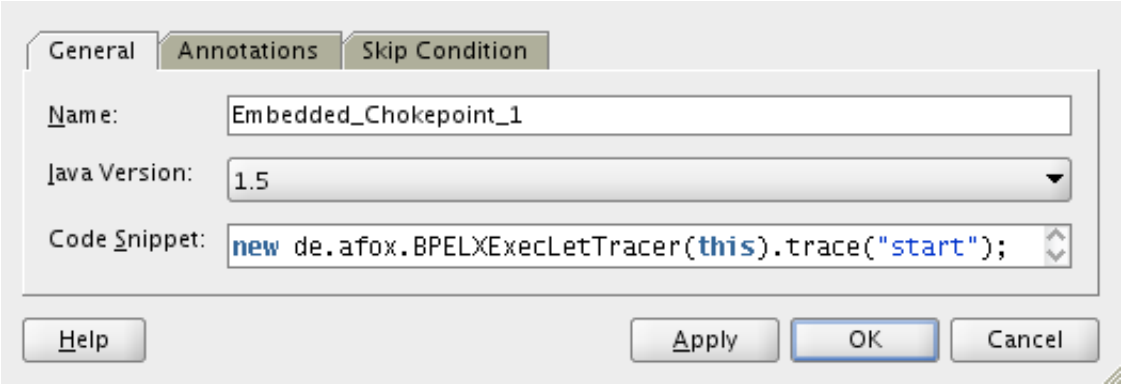

Abb. 3: Code-Instrumentierung über Embedded Java

Sensoren
Über Oracle BPELs Sensoren lässt sich die Aktivität von BPEL-Services überwachen. Ein Sensor ist eine Art „Trigger“, der auf bestimmte Ereignisse reagieren kann. Man unterscheidet im wesentlichen Activity-Sensors, Variable-Sensors und Fault-Sensors. Die Daten können an ein Publish-Target veröffentlicht werden. Für den Zweck der Performance-Daten-Sammlung könnten die Daten des Service-Aufrufs in einem Activity-Sensor gesammelt werden.

Sensoren können transparent in den Business-Flow eingebettet werden, allerdings gestaltet sich die Sammlung feingranularer Statistik-Daten schwierig.

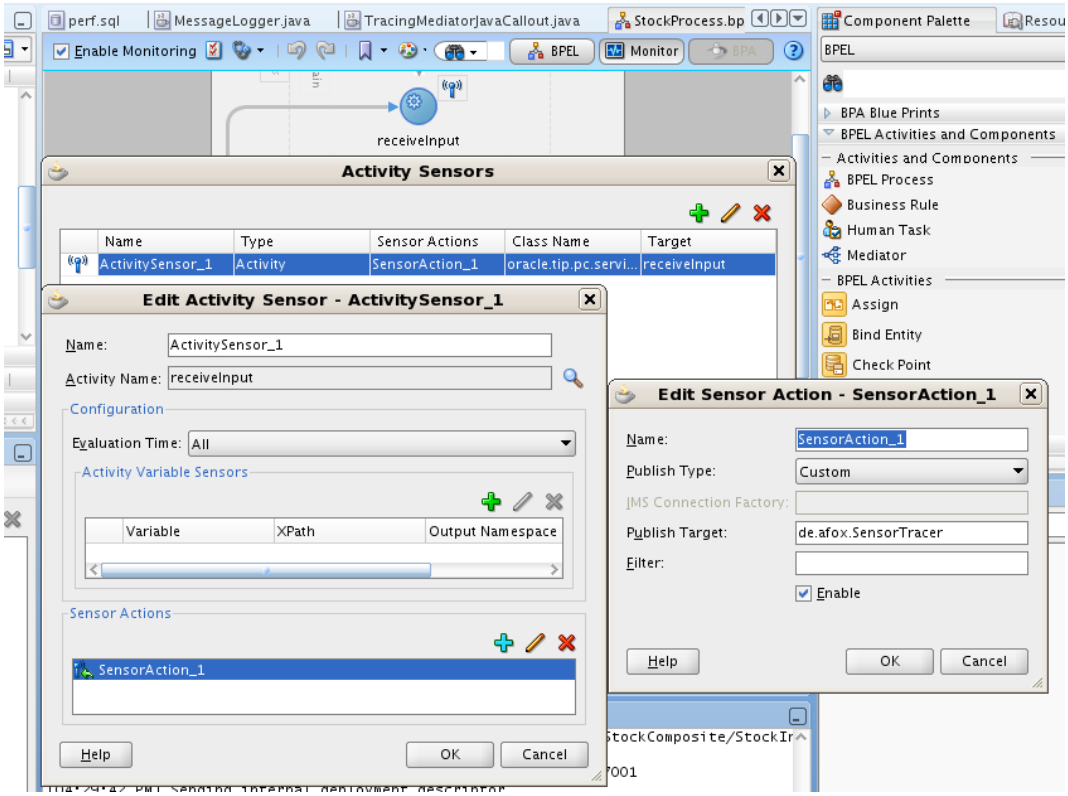

Abb. 4: Code-Instrumentierung über Sensoren

Mediator
Ein Mediator ist ein Proxy-Service für BPEL-Services. Das Interface eines Mediators entspricht dem Interface des zugeordneten Services. Der Mediator kann über einen Java-Call-Out erweitert werden, in dem die relevanten Statistikdaten gesammelt werden können. Der Mediator im Business-Flow transparent.

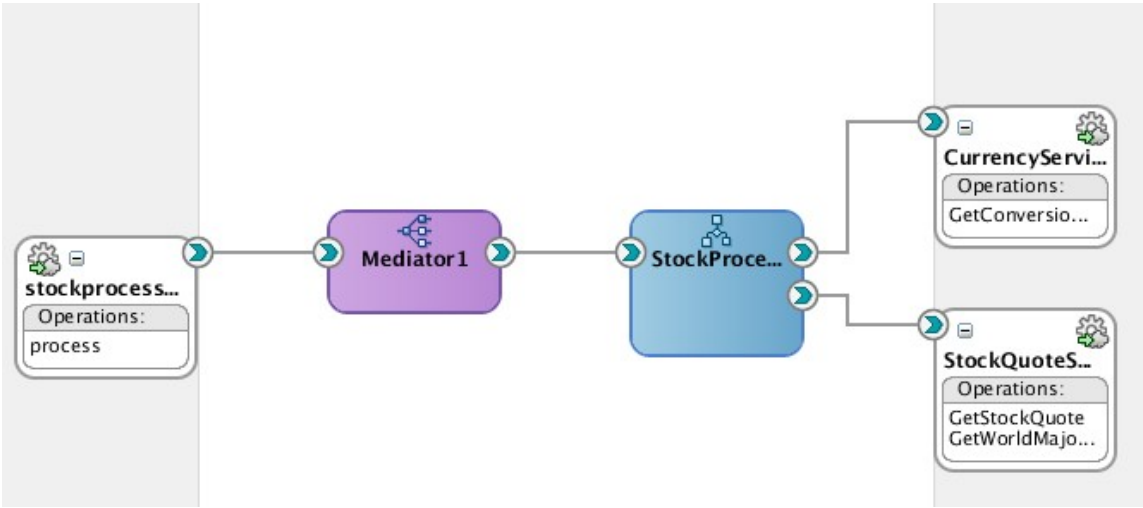

Abb. 5: Code-Instrumentierung über Mediator

Fazit

Oracle BPEL verfügt über Hilfmittel mit deren Hilfe sich die Performance von Produktiv und Testsystemen messen und die Einführung von Performance-Tuning-Massnahmen objektiv bewerten lassen. Durch die flächendeckende Verwendung von Performance-Tests lassen sich Synergien erzeugen, die sich auf andere Projektteile positiv auswirken. Das Vertrauen in die verwendeten Systeme, Software und Architekturen steigt. Die Software-Entwicklung kann agiler gestaltet werden und Refactorings leichter durchgeführt werden. Die Bereitschaft zum Patch und Upgrade von Systemen steigt, was die Sicherheit der Systeme erhöht. Letztendlich ist es das Ziel, Performance-Tests als ein Test-Bestandteil in den Continuous-Integration-Lifecycle des Unternehmens zu integrieren.

Kontaktadresse:

Alexander Fox
Oracle Deutschland B.V. & Co. KG
Robert-Bosch Straße, 5
D-63303 Dreieich

Telefon:	+49 (0) 6103 397 178
Fax:	+49 (0) 6103 397 397
E-Mail	alexander.fox@oracle.com
Internet:	www.oracle.com/de/index.html