



PETA: Werkzeug für Testautomatisierung

Besser testen

>> TORSTEN STOLPMANN

PETA ist ein umfassendes Werkzeug zur Testautomatisierung, das die Bereiche Integrations- und Systemtests sowie Lasttests abdeckt. Der Schwerpunkt liegt dabei auf dem Testen der Kommunikation von verteilten Systemen auf Protokollebene. Mit der Integration in die Eclipse-Plattform bietet PETA alle Voraussetzungen für eine effiziente Entwicklung und Verwaltung von Testprojekten.

Die Themen Qualitätssicherung und automatisiertes Testen von Software standen lange Zeit nicht im Fokus der Öffentlichkeit und wurden eher stiefmütterlich behandelt, dementsprechend gering ist die Menge an professionellen Werkzeugen zur Testautomatisierung. Da der Markt von wenigen kommerziellen, meist sehr teuren Produkten und einzelnen, oft stark spezialisierten Open-Source-Lösungen geprägt ist, arbeiten immer noch viele Firmen an ihren individuellen Automatisierungslösungen, die häufig auch noch für jedes Projekt neu erstellt werden.

Zur Testautomatisierung bieten sich generell drei verschiedene Schnittstellen an, die in der Regel bestimmten Testarten im Projektverlauf zuzuordnen sind:

- GUI-Schnittstellen (Systemtests)
- Kommunikationsschnittstellen (Integrations-tests)
- API-Schnittstellen (Modultests)

PETA wurde entwickelt, um speziell den Bereich Integrationstests abzudecken. Moderne Softwaresysteme wie SOA, Web-2.0-Applikationen oder Multi-Tier Enterprise-Architekturen auf Basis von J2EE oder .NET bestehen aus einzelnen Komponenten, die über Protokolle wie HTTP, JMS oder TCP Nachrichten austauschen. Diese Kommunikation ist im einfachsten Fall ein Request/Response-

Szenario, wie beispielsweise der Aufruf einer Webseite. In komplizierteren Fällen sind eine Vielzahl verteilter Systeme mit mehreren Clients, Servern und unterschiedlichen Kommunikationsprotokollen beteiligt.

Zur frühzeitigen Behebung und Lokalisierung von Fehlern in der Spezifikation und Implementierung derart komplexer Systeme ist es notwendig, umfassende Tests bereits in der frühesten Entwicklungsphase durchzuführen. Aufgrund der großen Menge an Testfällen und der üblichen, häufigen Änderungen im Projektverlauf empfiehlt es sich, diese Tests zu automatisieren, um zeitnah verlässliche und reproduzierbare Qualitätsaussagen treffen zu können.

Die automatisierte Ausführung von Softwaretests verlangt in der Regel einen mehr oder minder großen Anfangsaufwand, bevor erste Testergebnisse zur Verfügung stehen können. Besonders im Integrationsbereich wird durch die Vielzahl der Schnittstellen und die Problematik verteilter Zustände das Aufsetzen einer automatisierten Testumgebung zur anspruchsvollen Aufgabe. Um diese Aufgabe zu lösen, muss ein Werkzeug gezielt auf die hier auftretenden Problematiken zugeschnitten sein und unter anderem folgende Eigenschaften aufweisen:

- *Prozessverbesserung:* Im Idealfall sollte das Werkzeug nicht nur die Mög-

lichkeit zum Testen des fertig gestellten Gesamtsystems bieten. Der gesamte Entwicklungsprozess sollte durch die frühzeitige Verfügbarkeit von Tests profitieren können, auch wenn einzelne Integrationsbestandteile noch nicht verfügbar sind.

- *Wartbarkeit:* Redundanzfreiheit in der Testimplementierung und Wiederverwendbarkeit von gemeinsam benutzten Testbestandteilen sind Grundvoraussetzungen für eine effektive Umsetzung von Automatisierungsmaßnahmen. Dies ermöglicht, die von Entwicklern wie Testern gleichermaßen gefürchteten Anforderungsänderungen im Projekt mit vertretbarem Aufwand umzusetzen.
- *Erweiterbarkeit:* Das Werkzeug sollte projekt-übergreifend einsetzbar sein und durch Plug-ins und definierte Schnittstellen einfach an die jeweiligen Erfordernisse angepasst werden können. Dadurch kann ein maßgeschneidertes Testumfeld hergestellt werden.
- *Anpassbarkeit:* Das Werkzeug sollte sich nahtlos in vorhandene Umgebungen wie automatisierte Build- und Release-Management-Systeme einfügen lassen, ohne gewohnte Prozesse oder Tool-Ketten verwerfen zu müssen.

Bei der Entwicklung von PETA wurde besonders auf die Praxistauglichkeit Wert gelegt; die Mehrzahl der hier umgesetzten Konzepte resultieren aus eigenen Erfahrungen in großen Testprojekten. PETA wurde entworfen, um effizientes Arbeiten über den gesamten Entwicklungsprozess zu gewährleisten. Gerade in größeren Softwareentwicklungsprojekten, bei denen oft verschiedene Komponenten an verschiedenen Standorten entwickelt werden, sind Übersicht, Flexibilität und Wartbarkeit entscheidende Faktoren für einen erfolgreichen Projektverlauf.

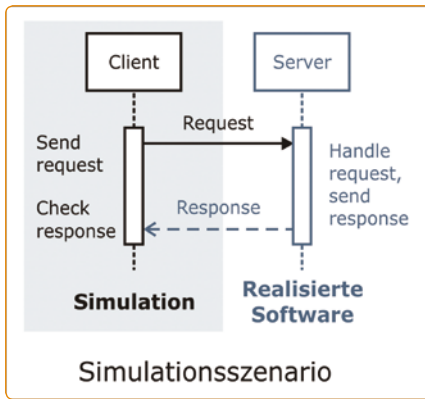


Abb. 1: Simulation von Komponenten

Simulationsfähigkeit

Normalerweise stehen zu Beginn eines Entwicklungsprojekts noch keine funktionsfähigen Komponenten zum Testen zur Verfügung. Deshalb beginnt der Testprozess oft erst spät im Entwicklungszyklus, und die Fehlerbehebung kurz vor Abnahme durch den Kunden wird zum Stressfaktor in jedem Projekt.

Um diese Situation zu entschärfen, lassen sich mit PETA noch nicht fertig gestellte Komponenten simulieren. Die nötigen Informationen über das zu erwartende Verhalten des Systems können dazu aus der Spezifikation entnommen werden. PETA-Testfälle werden mittels einer XML-basierten, deklarativen Beschreibungssprache formuliert. Diese Sprache erlaubt es, sich auf die Modellierung der Nachrichtensequenzen und die Behandlung der Nachrichten auf den beteiligten Systemen zu konzentrieren. So kann die Erstellung von Testfällen bereits mit Verfügbarkeit der ersten technischen Spezifikation beginnen.

Die Sprache orientiert sich stark an den aus der UML bekannten Sequenzdiagrammen zur Beschreibung von Kommunikationsvorgängen. Sie umfasst zur Zeit 29 Sprachelemente und enthält Konstrukte zum Versand und Empfang von Nachrichten sowie deren

PETA-Komponente	Aufgabe
PETA Core	Ausführungsumgebung
PETA Designer	Grafischer Editor
PETA Webrecorder	Recording-Komponente
PETA Load	Lasttest-Komponente

Tabelle 1: Die einzelnen PETA-Komponenten und ihre Aufgaben

Manipulation und Überprüfung. Der Funktionsumfang der Sprache kann bei Bedarf mithilfe von Java-basierten Plug-ins zusätzlich erweitert werden.

Die Entwicklung von Testfällen kann so parallel zur Implementierung der zu testenden Systeme stattfinden. Solange diese noch nicht verfügbar sind, werden sie entweder alle oder aber einzeln durch PETA simuliert. Auf diese Weise können die geschriebenen Testfälle ausgeführt und auf Korrektheit hin überprüft werden. Ist das zu testende System soweit fertig gestellt, dass es getestet werden kann, lassen sich die Tests einfach vom simulierten auf das echte System übertragen. Damit können auch Test-First-Entwicklungsansätze einfach unterstützt werden.

Trennung der Testbestandteile

Um Wiederverwendbarkeit und Redundanzfreiheit und damit eine einfache Wartung von Testfällen zu gewährleisten, werden diese in verschiedene Elemente zerlegt. Die Trennung von Infrastruktur und Test erlaubt den Einsatz eines Testprojekts in unterschiedlichen Umgebungen, ohne dass hierfür die Testfälle geändert werden müssen:

- Physische Informationen über die beteiligten Systeme sind in einer eigenen Konfigurationsdatei gekapselt. Dort werden Daten über die Infrastruktur, wie beispielsweise die verwendeten IP-Adressen oder die Information, welche der Komponenten simuliert werden sollen, abgelegt.
- Die Übermittlung der Nachrichten erfolgt über eine austauschbare Kommunikationsschicht, welche die jeweilige Abbildung auf ein Transportprotokoll (HTTP, TCP etc.) vornimmt.
- Die Testfälle selbst enthalten – in enger Anlehnung an UML-Sequenzmodelle – die Ablaufbeschreibung der Tests inklusive der enthaltenen Zusicherungen (Assertions). Die Testfälle setzen sich aus wiederverwendbaren Modulen zusammen, die so in verschiedenen Tests mehrfach genutzt werden können. Immer wiederkehrende Abläufe wie z.B. Login/Logout müssen so nur ein einziges Mal kodiert werden. Die Testfälle können, bei Bedarf auch zur Laufzeit, dynamisch erzeugt und manipuliert werden. Dies ermöglicht einen Modellgetriebenen Ansatz zur Generierung von Tests aus Meta-Daten.

- Die ausgetauschten Nachrichteninhalte werden in Templates gespeichert. Templates sind Dateien im XML-Format und werden getrennt von den einzelnen Testfällen abgelegt. Bei Bedarf können die Inhalte zur Laufzeit erzeugt und auf verschiedene Arten manipuliert werden. Zwischen den Templates und der Kommunikationsschicht können zusätzlich eigene Filter zur Formatwandlung und Überprüfung implementiert werden.

Plug-in-Architektur

Sowohl der ständige Fortschritt im technologischen Bereich als auch die daraus resultierenden Änderungen in der jeweiligen Umsetzung führen dazu, dass kaum noch ein Projekt im Hinblick auf eingesetzte Tools und Technologien seinem Vorgänger gleicht. Deshalb bietet PETA die Möglichkeit, die Plattform durch eine Vielzahl von Erweiterungsmöglichkeiten an die jeweilige projektspezifische Umgebung anzupassen.

Einige häufig benötigte Plug-ins werden bereits mitgeliefert, andere können durch den Anwender selbst implementiert werden. Die Erweiterungsmöglichkeiten lassen sich in vier Gruppen aufteilen:

- *Integrationserweiterungen* sind Schnittstellen, die Zugriff auf die Kernplattform selbst erlauben. Dadurch kann diese beispielsweise sowohl innerhalb der Eclipse Workbench als auch als Kommandozeilen-basierte Standalone-Applikation verwendet werden. Diese Schnittstellen ermöglichen so die Einbindung in andere Applikationen. Außerdem können auf diese Weise einfach externe Datenquellen wie Excel-Tabellen als Testgeneratoren eingebunden werden.
- Mit *Spracherweiterungen* lässt sich der Umfang der Beschreibungssprache durch den Aufruf beliebiger Java-Methoden erweitern. So existieren beispielsweise Plug-ins zur Erzeugung dynamisch generierter Inhalte oder der Definition von Massendaten oder Timeouts. Des Weiteren unterstützen frei definierbare Datentypen die Formulierung von Zusicherungen.
- Transportprotokolle werden durch *Protokollerweiterungen* implementiert. Momentan unterstützt die Plattform HTTP, HTTP/S und TCP. Andere, wie SFTP, Telnet oder eine SQL-Anbin-

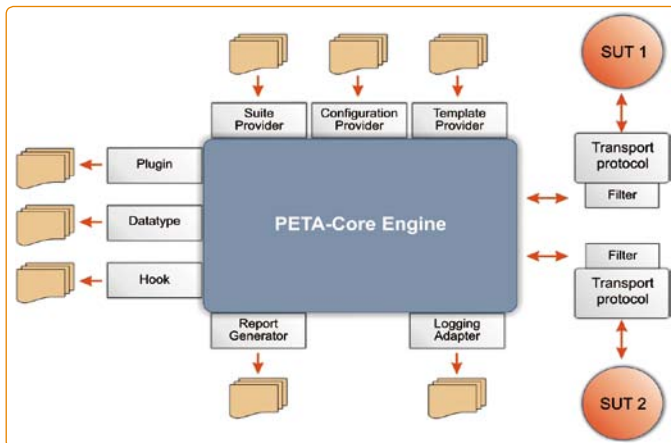


Abb. 2: Die PETA Core-Architektur

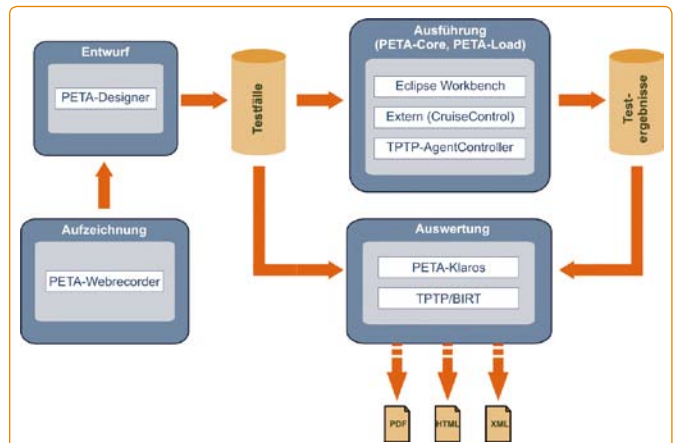


Abb. 3: Plattform-Übersicht

derung, können zugestellt werden. Da die Einzelheiten zur Transportschicht als physische Information in der Konfigurationsdatei beschrieben werden, sind sie für die Testfälle in der Regel nur sichtbar, wenn auf spezielle Eigenschaften der Transportschicht im Test direkt zugegriffen wird (beispielsweise HTTP-Header-Parameter). So können bei heterogenen Systemen die Protokolle einfach ausgetauscht werden, ohne dass der Testfall geändert werden muss. Außerdem bieten Filter-Plug-ins wie ein HTML-nach-XML-Konverter oder ein XML-Schema-Validator die Möglichkeit, weiteren Einfluss auf den Datenstrom zu nehmen.

- **Reporting-Erweiterungen:** Um die Integration der Plattform mit anderen Reporting Tools oder Datenbanken zu ermöglichen, kann das Format des Testergebnis-Reports an individuelle Erfordernisse angepasst werden. So können maßgeschneiderte Berichte in verschiedenen Formaten wie HTML, XML, PDF usw. ausgegeben werden.

Übersicht

Die PETA-Plattform besteht aus fünf Komponenten, die je nach Anforderung individuell zusammengestellt werden können.

Installation

Als Systemvoraussetzung werden Windows 2000/Windows XP oder Linux IA32 als Betriebssystem sowie Eclipse SDK 3.2 mit Java 1.5.0 benötigt. Das Produkt ist sowohl als Stand-alone-Installation als auch als Add-on zu einer bestehenden Eclipse-3.2-Installation mit allen benötigten Plug-ins (BIRT, EMF, GEF, TPTP, WTP) verfügbar. Die Einrichtung erfolgt

über ein eigenes Installationsprogramm. Unterstützung für Eclipse 3.3 wird für die kommende Version 2.1 zugesagt, die mit Erscheinen des Artikels vorliegen wird.

Testerstellung

Am Anfang der Testerstellung steht die Definition der beteiligten Softwaresysteme, hier *Aktoren* genannt. Im einfachsten Fall wären dies ein Client und ein Server, der über ein definiertes Protokoll angesprochen werden kann (beispielsweise SOAP über HTTP). Dazu werden im PETA-Designer mithilfe von Wizards die benötigten Daten eingegeben. Diese Daten werden in einer Konfigurationsdatei festgehalten, die bei Bedarf später noch über einen Formular-basierten Editor oder direkt im XML-Quellcode editiert werden kann. Neben vielfältigen, meist Protokoll-spezifischen Eigenschaften wie beispielsweise der zu verwendenden URLs der als Server fungierenden Aktoren können hier in verschiedenen Setups die Rollen der einzelnen Aktoren (simuliert, zu testen, nicht vorhanden) zugeordnet werden. Der Wechsel der Testumgebung (Entwicklung/QA/Staging/Produktion) kann so an zentraler Stelle durch Wechsel der Konfiguration ausgeführt werden, ohne dass hierfür Testfälle angepasst werden müssen.

Die Erstellung von Tests erfolgt mithilfe eines GEF-basierten grafischen Editors, der auch Anwendern ohne Programmierkenntnisse das einfache Erstellen von Testfällen mittels Drag & Drop erlaubt. Die Testerstellung wird durch eine Vielzahl von Wizards und Dialogen sowie eine aufwendige Fehlerprüfung und -markierung während der Bearbeitung unterstützt. Alternativ kann die Erstellung aber auch direkt im XML-Quelltext erfolgen.

In Anlehnung an JUnit [1] können einzelne Tests zu Testsuiten zusammengefasst werden, um diese gemeinsam auszuführen. Die Wiederverwendung einzelner Testabschnitte wird durch Verwendung von Modulen geregelt, die wiederkehrende Instruktionen in einzelnen Dateien kapseln.

Grundlage der versendeten Nachrichteninhalte bilden Templates, die während des Testablaufes noch dynamisch verändert werden können. Zur Modifikation und Prüfung der versendeten Nachrichten können deren Inhalte mittels XPath [2] adressiert werden.

Für die Erstellung der Java-basierten Plug-ins steht die gewohnte JDT-Entwicklungsumgebung zur Verfügung, die durch eigene Wizards ergänzt wird.

Webrecorder

Neben der händischen Konstruktion von Testfällen unterstützt die Plattform durch einen Webrecorder auch die Aufzeichnung von Browserinteraktionen. Mit der Recording-Funktion werden automatisch lauffähige Tests sowie eine vollständige Konfigurationsdatei aus den gewonnenen Daten generiert. Dabei verlässt sich der PETA Webrecorder im Gegensatz zu vielen anderen Capture/Replay Tools nicht auf die Aufzeichnung von Mausbewegungen/-positionen, Tastendrucke und die Auswertung von Bildschirmdarstellungen, sondern protokolliert die übermittelten Nachrichten zwischen Browser und Webserver. Somit wird eine größere Unabhängigkeit von kleineren Änderungen der Weboberfläche erreicht.

Die so gewonnenen Tests können direkt weiterbearbeitet, in Module zerlegt und weiter verfeinert werden. Je nach Komplexität der zu testenden Webappli-

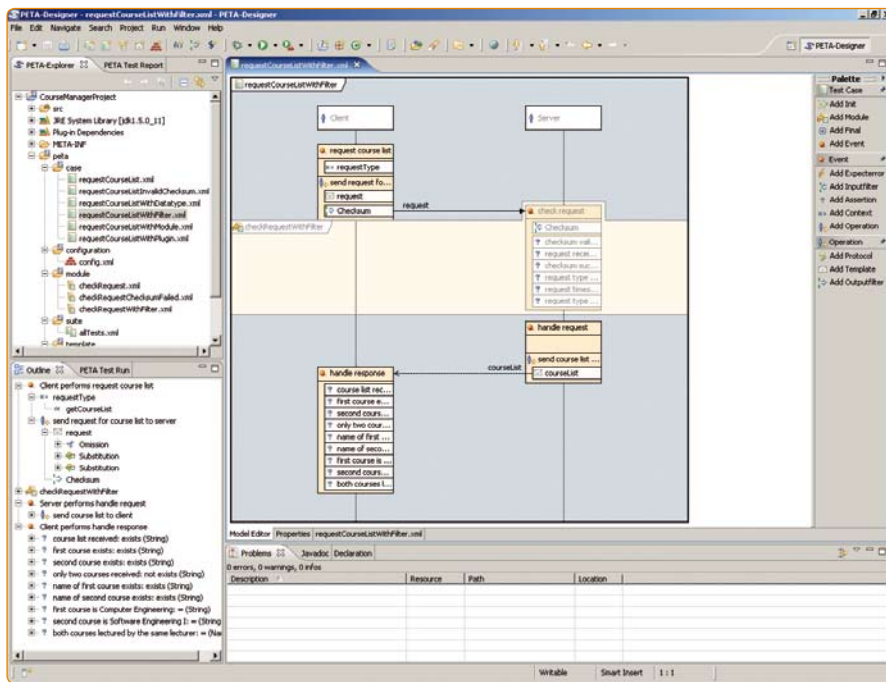


Abb. 4 Screenshot PETA-Designer - Testfalleditor

kation ist eine weitere Bearbeitung möglich, um den korrekten Umgang mit von der Webapplikation dynamisch erstellten Informationen wie beispielsweise Authentifizierungsinformationen zu gewährleisten. Besonders effektiv können sie als Testfallmenge für Lasttests weiter verwendet werden.

Testausführung

Die Ausführung von Tests kann prinzipiell auf drei verschiedene Arten erfolgen:

- für Ad-hoc-Tests und zur Testentwicklung direkt innerhalb von Eclipse mit dem von JUnit-Tests bekannten Interface (Abb. 4),
- für Regressionstests als separate Java-Anwendung innerhalb eines Buildsystems (Apache Ant [3], Apache Maven [4], einem Continuous-Integration-System wie CruiseControl [5]) oder anderen Testmanagementsystemen via Kommandozeilen Interface oder Apache Ant-Task,

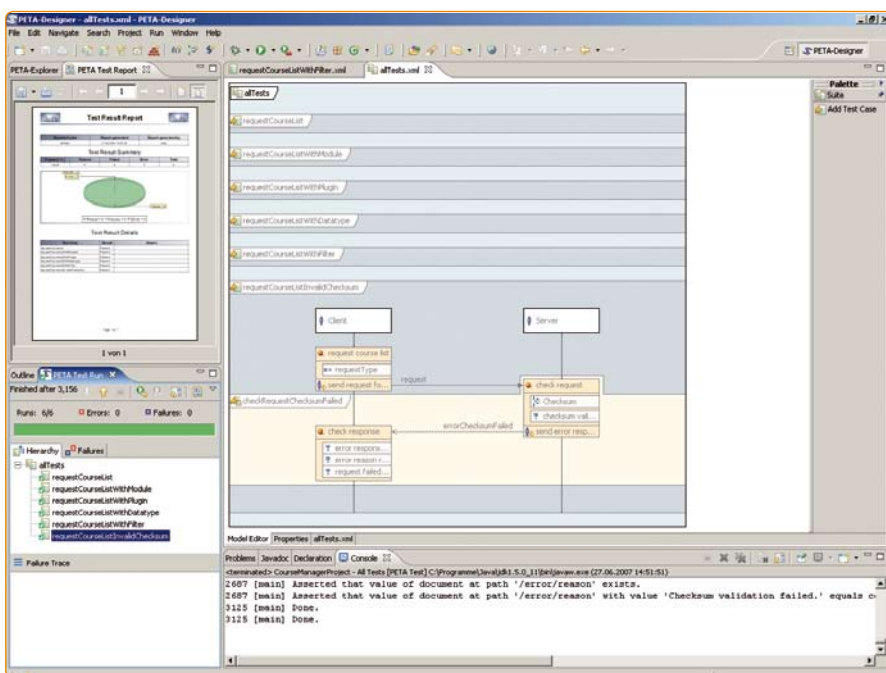


Abb. 5: Screenshot PETA Designer - Testfallausführung

- für Last- und Stresstests in einem oder mehreren Eclipse TPTP Agent Controllern durch Integration in das Eclipse TPTP-Testframework [6].

Testauswertung

Analog zu den vorher angeführten Ausführungsoptionen werden jeweils verschiedene Auswertungsmöglichkeiten geboten. Innerhalb der Eclipse Workbench bietet eine optionale Report-Generierung den Zugriff auf grafische Reports in verschiedenen Ausgabeformaten (PDF, HTML, XML) sowie die gesamte Bandbreite der vom TPTP-Framework gebotenen BIRT-basierten Reporting-Infrastruktur. Bei einer Ausführung außerhalb der Eclipse Workbench werden durch die offene Reporting-Plug-in-Schnittstelle auch individuelle Lösungen unterstützt.

Zusammenfassung

Gerade in größeren Softwareprojekten mit einer Vielzahl von Komponenten sind automatisierte Tests ein wichtiger Bestandteil für frühzeitige und verlässliche Aussagen über die Qualität von Software. Durch die klare Trennung von fachlichen und technischen Bestandteilen wird der Aufwand bei Änderungen minimal gehalten und der Einsatz in verschiedenen Umgebungen unterstützt. Für jedes Projekt steht eine große Auswahl an Konfigurations- und Erweiterungsmöglichkeiten zur Verfügung. So können individuelle Testumgebungen auf Basis einer gemeinsamen Plattform geschaffen werden, die eine effektive Verwendung der eingesetzten Ressourcen unterstützen.



Torsten Stolpmann ist Geschäftsführer der verit Informationssysteme GmbH (www.verit.de). Als Leiter der Entwicklung und Chefarchitekt ist er maßgeblich für die PETA-Architektur verantwortlich. Kontakt: torsten.stolpmann@verit.de

>>Links & Literatur

- [1] JUnit: www.junit.org
- [2] XML Path Language (XPath): www.w3.org/xpath
- [3] Apache Ant: ant.apache.org
- [4] Apache Maven: maven.apache.org
- [5] CruiseControl: cruisecontrol.sourceforge.net
- [6] Eclipse Test & Performance Tools Platform Project: www.eclipse.org/tptp