Sonderausgabe

www.hanser-automotive.de

SPEZIAL

HANSER automotive

Sonderdruck

aus Hanser automotive 9/2019

Stephan Schramm (Sogeti) und Jochen Eberle (extern)

Effizientes Testen von agilen Softwareprodukten

© Shutterstock

sogeti
Part of Capgemini

Sogeti Deutschland GmbH
Balcke-Dürr-Allee 7
40882 Ratingen
Telefon: +49 2102 101 4000
Homepage: www.sogeti.de

Impressum: Verlag: Carl Hanser Verlag GmbH & Co. KG, Kolbergerstr. 22, 81679 München; Druck: Alpha-Teamdruck GmbH, Haagener Str. 9, 81671 München
© Carl Hanser Verlag, München. Alle Rechte, auch die des Nachdrucks, der photomechanischen und elektronischen Wiedergabe sowie der Übersetzung dieses Sonderdruck behält sich der Verlag vor.

HANSER

>>> SW-QUALITÄTSSICHERUNG UND -TEST

Effizientes Testen von agilen Softwareprodukten

Agile Vorgehensmodelle kommen zunehmend auch in der Automobil- und deren Zulieferindustrie für die Produktentwicklung von Software zum Einsatz. Das hier vorgestellte Testarchitekturmodell zeigt, dass sich aus einer klassischen Testplanung sehr wohl eine effiziente Testautomatisierungslinie entwickeln lässt. Auswahl, Anordnung und Gebrauch der Testtools bestimmen den Wirkungsgrad innerhalb der Deployment-Pipeline.

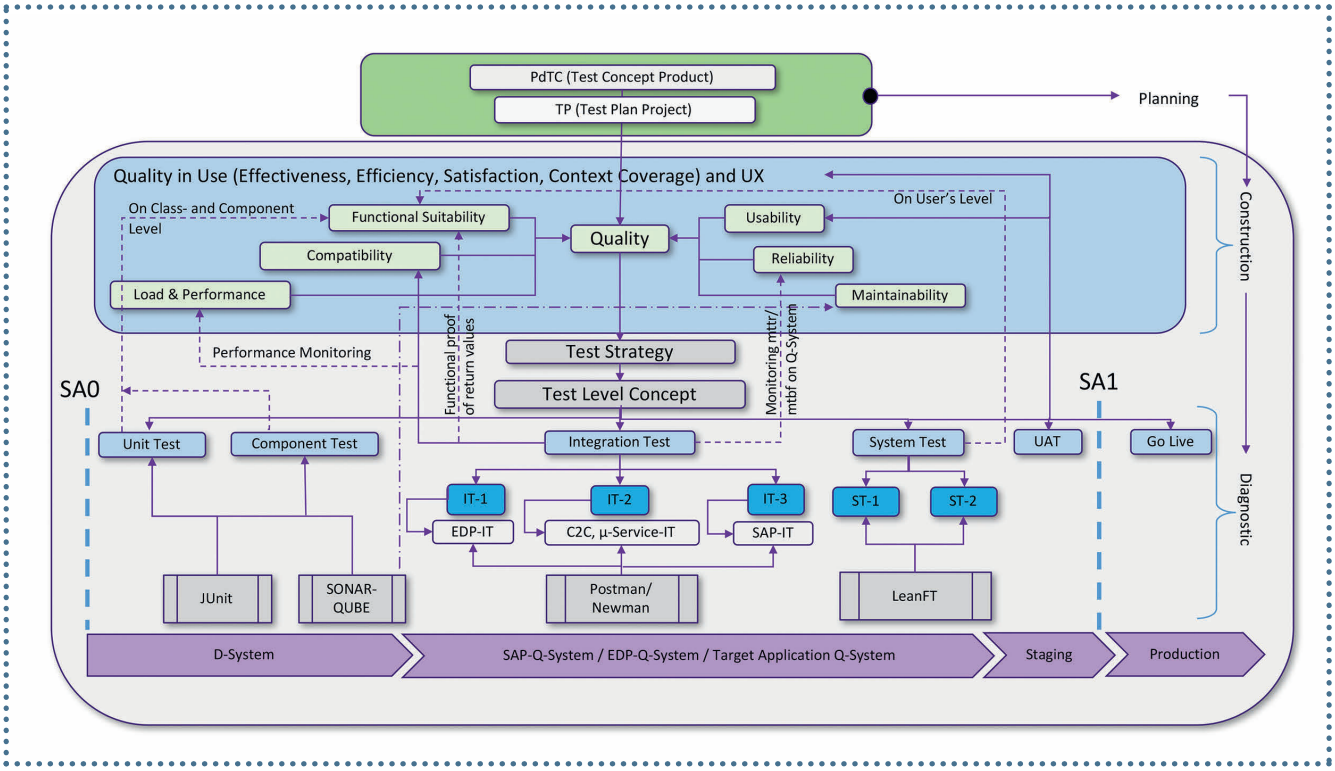

Bild 1: Ganzheitliche Darstellung der Testarchitektur. (Quelle: S. Schramm)

Dieser Artikel betrachtet die Softwareentwicklung einer Intralogistikapplikation in der Automobilzulieferindustrie aus der Perspektive der Qualitätssicherung, wobei die Testarchitekten die Chance ergriffen haben, ein Modell mit hohem Automatisierungsgrad zu entwickeln und zu implementieren. Anlass zu dieser Entscheidung gab der DevOps orientierte Softwareansatz mit agilen Service- und Plattformteams, Microservice-Architekturen und einer Infrastruktur basierend auf einer Continuous Integration- und Continuous Delivery-Pipeline. Wie aus Bild 1 ersichtlich, entwickelt sich die Testarchitektur aus der Betrachtung der folgenden 4 Aktionsebenen:

- *Planning*: normgerechte Planung des Produkttestprozesses (z. B. angelehnt an IEEE-829)
- *Construction*: aus dem Architekturwürfel abgeleitete Auswahl der „build-in quality" Kriterien
- *Test Strategy*: Ansatz zu einem systematischen, skalierbaren Vorgehen

© Carl Hanser Verlag, München
© Carl Hanser Verlag, München. Vervielfältigungen, auch auszugsweise, sind ohne Lizenzierung durch den Verlag nicht gestattet.

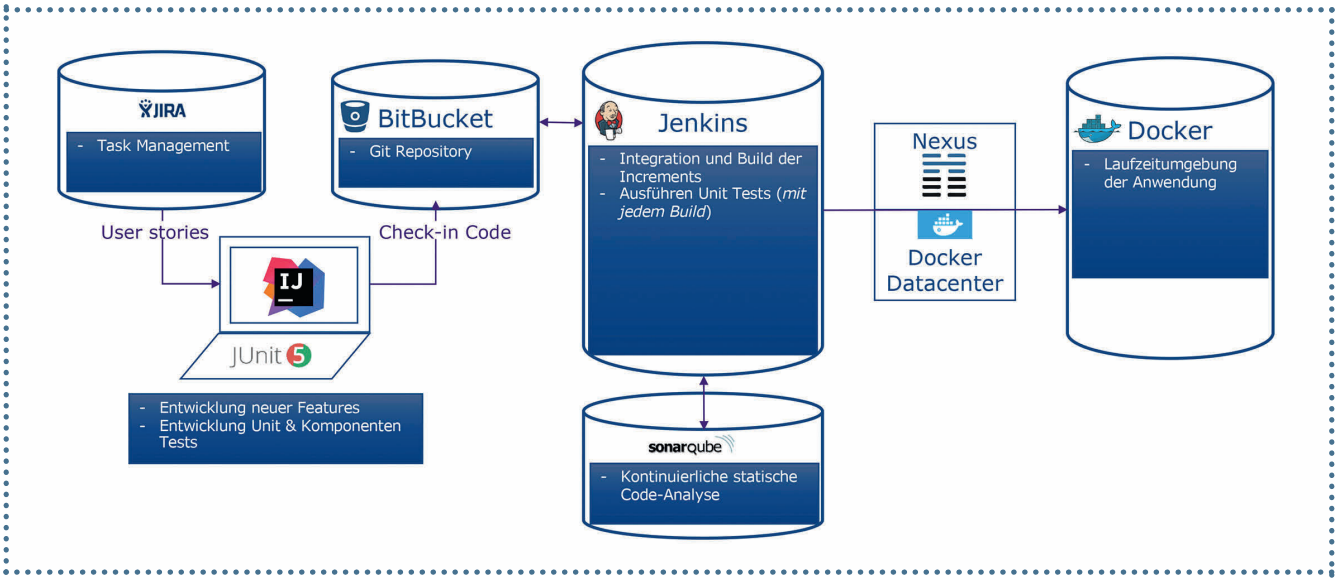

Bild 2: Entwicklungs- und Deployment-Pipeline nur mit Unit-Tests. (Quelle: J. Eberle)

- *Diagnostic*: Implementierungsmodell der Test-Automatisierungs-Pipeline

Klassischer Ansatz einer Testplanung

In der Regel erwartet der Auftraggeber in der Automobilindustrie einen soliden Qualitätsnachweis der zu liefernden Software, wobei alle leistungsrelevanten Qualitätskriterien zuverlässig ausgeprägt sein müssen. Somit werden neben der funktionalen Qualitätskomponente (Korrektheit, Vollständigkeit und adäquate Aufbereitung der Daten) auch Zuverlässigkeit und Effizienz von der Software verlangt. Eine solide und verlässliche Qualitätsabsicherung kann aber nur erbracht werden, wenn der technische Testprozess auf einem soliden methodischen Fundament steht.

Die Testarchitektur von Sogeti basiert auf dem Konzept, die zu verifizierenden Testobjekte aus einer In Scope/Out of Scope Analyse der Geschäfts- bzw. Applikationsarchitektur herauszuarbeiten, um eine für das Testdesign solide Testbasis zu gewinnen.

Implementierungsmodell

Ausführlich beschrieben wird die konkrete technische Testarchitekturlösung sowie der Testinformationsfluss durch die Deployment-Pipeline mit folgenden Prozessbausteinen:

- Aufbau der Deployment-Pipeline
- 4-stufiger Continuous Integration Prozess
- Integrationstests von SAP-Schnittstellen
- Einbindung eines dazu gehörenden Prozessschemas
- Aufbau eines automatischen Systemtests

Ist-Zustand

Die Deployment-Pipeline des Projekts folgt dem klassischen Ansatz mit einem Source Code Repository (BitBucket) mit Versionierung, einem Continuous Integration Tool (Jenkins), welches den Prozess orchestriert, sowie einer Containervirtualisierung (Docker) zum Ausführen der Build- und Testprozesse. Ergänzend dazu wird SonarQube zum Ausführen und Verwalten von Unit Tests sowie zum Durchführen der statischen Code-Analyse verwendet. Als Testmanagement Tool wird Jira/Xray eingesetzt, d. h. zur Erstellung, Einplanung und Nachverfolgung von User Stories (Bild 2).

Der bisherige Testprozess hat einen sehr starken Fokus auf Unit Tests. Durch die Verschiebung des Testaufwands in Richtung Anwendungsentwicklung soll eine möglichst hohe Feedbackrate sichergestellt werden. Der Unit Test als solcher bietet als einziger Testtyp die Möglichkeit, einen Fehlerzustand im Quellcode zu lokalisieren. Er ermöglicht dadurch eher eine genauere Rückmeldung als eine reine Information über Bestehen oder Nichtbestehen. Ergänzt werden diese Entwicklertests durch einen manuellen und explorativen Abnahmetest am Ende eines jeden Sprints.

Auf diesem bestehenden Entwicklungs- und Testvorgehen aufbauend gilt es, einen vollautomatisierten Testprozess über alle Testebenen zu installieren. Der Testprozess soll ausgebaut werden, da mit dem bisherigen Vorgehen Integrations- und Systemtests vernachlässigt wurden. Dies hat letztlich zu einer Abnahme der Softwarequalität geführt.

In einer solch hochautomatisierten Deployment-Umgebung muss der Testprozess einen mindestens genauso hohen Automatisierungsgrad wie der Build- und Transportprozess aufweisen, um den Prozessfluss in die Produktionsumgebung nicht signifikant zu verlangsamen.

Continuous Integration Prozess

Grundsätzlich findet der in Bild 3 dargestellte Continuous Integration Prozess in vier Stufen statt:

- Einchecken einer Änderung des Quellcodes durch den Entwickler in einen Feature Branch des Source Code Repositories.
- Pull Request: Antrag auf Merge (Zusammenführung) der Änderungen zum Development Branch des Source Code Repositories.
- Sog. Pull Request: Builder auf Jenkins führt ein Deployment des Codes im Development Branch auf dem Development Server aus. Unit Tests und statische Code Analyse werden in diesem Zuge durchgeführt.

© Carl Hanser Verlag, München. Vervielfältigungen, auch auszugsweise, sind ohne Lizenzierung durch den Verlag nicht gestattet.

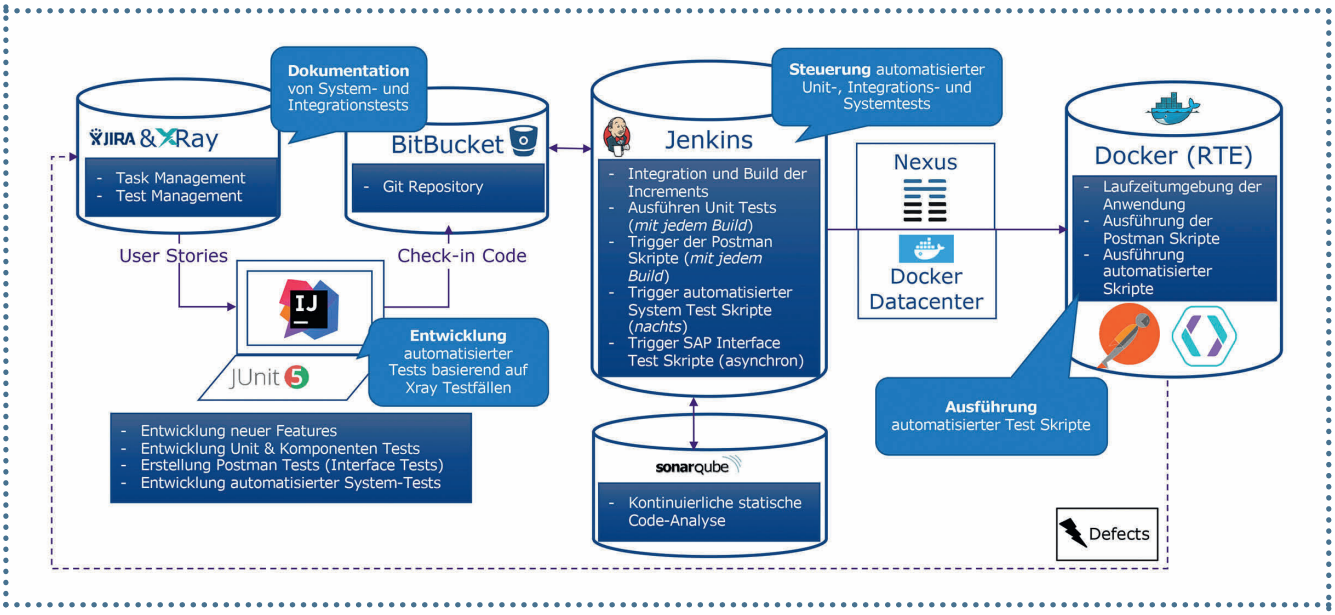

Bild 3: Vollständige Entwicklungs-, Deployment- und Test-Pipeline. (Quelle: J. Eberle)

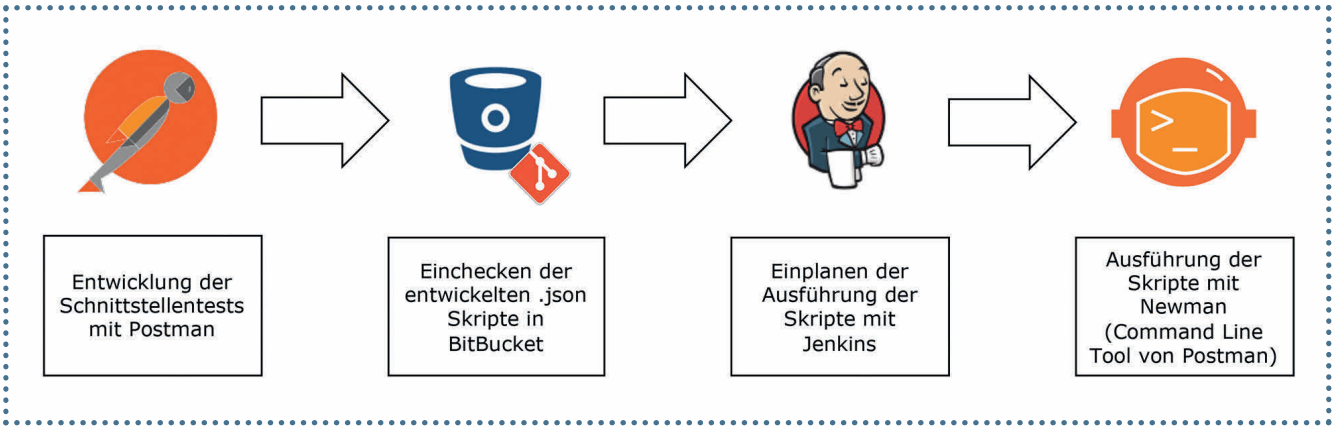

Bild 4: Prozessschema zur Ausführung von automatisierten API-Tests mit Postman. (Quelle: J. Eberle)

- Bei erfolgreichem Build und erfolgreicher Ausführung der Unit Tests wird dem Pull Request stattgegeben. Dieser löst den Merge zum Development Branch aus.

Nach Abschluss dieses 4-stufigen CI-Prozesses findet das Deployment des Codes vom Development Branch auf den Quality Server statt. Auf diesem Quality Server werden die manuellen und explorativen Tests ausgeführt.

Soll-Zustand

Der Kerngedanke ist die Installation weiterer Qualitätskontrollen, welche einen höheren Qualitätsstandard der entwickelten Inkremente sicherstellen. Die zentralen Testebenen hierbei sind Integrationstest und Systemtest.

Der Integrationstest wird aufgeteilt in drei unterschiedliche Typen:

- Interne Schnittstellen
- Test der Schnittstellen zu externen Tools
- Schnittstellen zu SAP

Diese Unterscheidung erfolgt aufgrund unterschiedlicher Herangehensweisen bezüglich des Testdesigns und der Testausführung. Dies ist auch der Grund für die komplette Separation des Tests der SAP-Schnittstellen. Der Test interner Schnittstellen hat im ersten Schritt sowohl hinsichtlich des Tools als auch in der Art der Testerstellung den Charakter von klassischen Unit Tests. Dies bedeutet, dass Szenarien direkt im Quellcode der API mit JUnit definiert werden.

Diese können einen Aufruf der API simulieren und die Korrektheit der Funktionalität validieren. Im zweiten Schritt muss ein Schnittstellenaufruf außerhalb des Tools simuliert werden. Das Tool der Wahl hierfür und auch für die anderen beiden Schnittstellentesttypen ist Postman. Die Gründe hierfür liegen im großen Funktionsumfang der kostenlosen Version, der einfachen und übersichtlichen Handhabung sowie der guten Integrierbarkeit mit Continuous Integration Tools wie Jenkins. Sowohl JUnit als auch Postman Test werden über Jenkins eingeplant und synchron mit den Software Builds ausgeführt (Bild 4).

System Test

Auch die System Tests werden in zwei Subkategorien untergliedert. Zum einen gibt es User Story Tests, mit denen exakt der Funktionsumfang einer User Story getestet wird. Zum anderen gibt es Ende-zu-Ende Tests, welche einen kompletten Weg durchs System und damit ein typisches Nutzerverhalten simulieren. Beide Arten haben unterschiedliche Anwendungsgebiete. Dabei ist der Aufwand zur Erstellung automatisierter Ende-zu-Ende Tests überschaubar, da hierfür große Teile des Codes der User Story Skripte wiederverwendet werden können.

Zum Erstellen der automatisierten Systemtests kommt das Tool *LeanFT* des Herstellers *MicroFocus* zum Ein- »

© Carl Hanser Verlag, München. Vervielfältigungen, auch auszugsweise, sind ohne Lizenzierung durch den Verlag nicht gestattet.

satz. Genau wie die zu testende Applikation ist die Sprache der Tests hier Java. Das Tool kann in die Integrated Development Environment (IDE) der Entwickler integriert werden. Der Funktionsumfang ähnelt damit dem von S*elenium*. Vorteile liegen in einer leichteren Bedienbarkeit, schnelleren Testausführung und einer größeren Vielfalt unterstützter Technologien. So können mit LeanFT auch SAP-Oberflächen automatisiert werden. Gerade im Bereich der Ende-zu-Ende Tests ist dies ein großer Vorteil, wenn der Erfolg durchgeführter Tests automatisiert in SAP überprüft werden kann.

User Story Tests können direkt vom Entwickler erstellt werden. Nachdem die Entwicklung eines Features einer User Story abgeschlossen ist, kann direkt im Anschluss durch dieselbe Person ein automatisierter Oberflächentest erstellt werden. Die Skripte des Testprojektes liegen in einem eigenen Git Repository auf demselben BitBucket Server wie der Anwendungscode. Die Ausführung erfolgt mittels Jenkins auf einem virtuellen Windows Server. Aufgrund langer Laufzeiten dieser Tests kann sich eine Ausführung mit jedem Build als problematisch erweisen. Eine nächtliche Ausführung sollte daher in Betracht gezogen werden.

Die Ende-zu-Ende Tests übernehmen die Funktion eines Go-Live Tests. Nach dem Roll-Out der Software stellen die Tests eine korrekte Funktionalität sicher. Außerdem können die End-User durch realitätsnahe Szenarien gut nachvollziehen, welche Funktionalität der Test adressiert. Dies kann auch das Vertrauen der Nutzer ins System erhöhen.

Fazit

Der hier vorgestellte Prozessweg zeigt, wie aus einem klassischen Testansatz eine hochautomatisierte Test-Pipeline entwickelt werden kann. Diese Lösung lässt sich auch in eine auf DevOps ausgerichtete Deployment-Pipeline anpassen.

Ein rein auf testtheoretischen Betrachtungen beruhendes Testkonzept ist für eine auf Testautomatisierung ausgelegte Testlösung wirkungsarm. Ein rascher Proof of Concept Einsatz der Automatisierungstools führt zu einem initialen Herzschlag im automatisierten Qualitätsflow. In einem über das Testcockpit gesteuerten Step-by-Step-Vorgehen lassen sich dann gezielt Pain Points abstellen und fehlende Testbausteine (Prozess und Technik) hinzufügen. ■ (oe)

» **www.sogeti.de**

Stephan Schramm unterstützt bei Sogeti Unternehmen der Automobilbranche bei der Entwicklung sowie der Praxisumsetzung von Methoden zur Qualitätssicherung von SW-Produkten und der Einführung agiler Vorgehensmodelle.

Jochen Eberle (B. Eng.) arbeitet in der Softwareentwicklung eines großen Automobilzulieferers. Der Aufgabenschwerpunkt liegt bei der Konzeption und Umsetzung automatisierter Testprozesse auf den verschiedenen Testebenen.

Capgemini | sogeti
Part of Capgemini

Die neueste Studie zum
Continuous Testing
ist ab sofort erhältlich!

https://www.sogeti.com/ctr2019

© Carl Hanser Verlag, München
© Carl Hanser Verlag, München. Vervielfältigungen, auch auszugsweise, sind ohne Lizenzierung durch den Verlag nicht gestattet.

sogeti
Part of Capgemini

JETZT AKKREDITIERT!
ISTQB® CTAL Security Tester

Security Schulung?

Aber sicher. Mit Sogeti.

ISTQB® CTFL • ISTQB® CTFL Agile Tester •
ISTQB® CTAL Testmanager •
ISTQB® CTAL Test Automation Engineer •
GTB CTFL Test Data Specialist

iSQI's® CABA • iSQI's® CAE •
iSQI® Scrum Master Pro •
iSQI® Practitioner in Agile Quality

CMAP Mobile App Testing – Foundation Level •
Performance Testing • Test Automation

IREB® CPRE-FL

Certified Selenium Tester Foundation

TMap NEXT® Test Engineer •
Test Manager • Praxisworkshop

TPI NEXT® Foundation •
TPI NEXT® für Testmanager

Individuelle Schulungen: u. a. Security •
Testautomatisierung • Agilität

ISTQB® International Software Testing Qualifications Board

iSQI® International Software Quality Institute

GTB German Testing Board
Software. Testing. Excellence.

Besuchen Sie uns unter
www.sogeti.de/2019