

OBJEKTSpektrum

Die Zeitschrift für Software-Engineering und -Management

Quality
Management

plan → do

act

- Testing: wann und wie?
- Testen in agilen Projekten
- Organisationsstrukturen für effizientes Testen
- Checkliste für den Testprozess

check

DAS A UND O FÜR HÖCHSTE QUALITÄT IN DER SOFTWARE-ENTWICKLUNG LAUTET

- A.** ein umfassendes Code-Verständnis, direkt in der Entwicklungsumgebung bereitgestellt
- O.** die Definition und Überwachung von unternehmensspezifischen Qualitätsrichtlinien

Mit der **PKS Application Evolution Suite** (kurz **PAES**) erhalten Sie die Grundlage für beides.

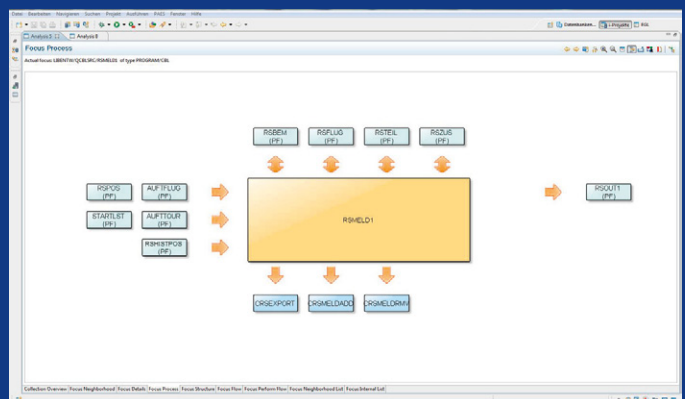
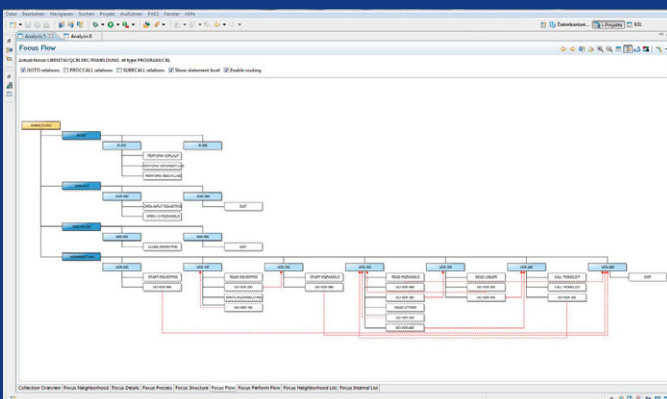
Unsere Parser-Technologie wird dazu nahtlos in RDP oder RDz integriert und Sie tappen bei der Analyse und Optimierung gewachsener Applikationen nicht länger im Dunkeln.

Knipsen Sie das Licht an und beleuchten Sie Ihre Quellen von allen Seiten!



MIT PAES VERFOLGEN UND ERREICHEN SIE DAUERHAFT DAS ZIEL EINER BESSEREN QUALITÄT IHRER SOFTWARE-ENTWICKLUNG, DENN:

- Die Lesbarkeit des Codes wird erhöht
 - Die Wartbarkeit des Codes wird verbessert
 - Die Fehleranfälligkeit des Codes wird reduziert
 - Die Einarbeitung neuer Entwickler wird vereinfacht
 - Code-basierte Performanceprobleme werden frühzeitig erkannt
- ... damit mehr Zeit bleibt für's Wesentliche – der Implementierung neuer Geschäftsanforderungen



Kundenzitat: „Nur durch die Feldverfolgung auf Basis PAES waren wir in der Lage, alle Stellen effizient ausfindig zu machen, die von der Feldlängen-änderung eines Primärschlüsselfeldes der Anwendung betroffen waren. Das hat uns enorme Testaufwände und manuelle Arbeiten gespart.“



PKS Software GmbH
Georgstraße 15, D-88214 Ravensburg
Tel: +49 (0) 751 56140-229
schmidt@pks.de, www.pks.de

Kann man aus „Fehlern“ lernen?

Liebe Leserinnen und Leser,

glaubt man den Statistiken, scheitern die meisten Softwareprojekte oder überziehen zumindest den Zeit- oder Budgetrahmen. Leider liest man nur selten von diesen Projekten, obwohl man doch aus Fehlern bekanntlich lernen kann. Ich möchte Ihnen daher von einem gescheiterten Softwareentwicklungsprojekt erzählen.

Ein Kunde bestellte ein Plugin für den Microsoft Windows Explorer. Das Plugin sollte Inhalte aus einem Directory-Service anzeigen und bearbeiten können. Ich schätzte den Aufwand auf zehn Wochen. Mein Chef reagierte nur mit einem Lächeln, was kein gutes Zeichen war. Ich rechnete damit, dass er mir nur fünf bis sechs Mannwochen geben würde (der Anblick von entspannten Mitarbeitern war nicht so sein Ding).

Was hat er auf meine Schätzung geantwortet? Ungefähr folgendes: „Du hast eine Woche und wir haben gerade keine weiteren Ressourcen, was positiv ist, denn du kannst Zeit sparen, da du keine Projektleitung machen musst und die Unit-Tests machst du on the fly.“ – Stille.

Was hat er da alles gesagt? Hielt er mich für Scotty von der Enterprise? Aus Tagen werden auf der Enterprise gerne Stunden und Scotty im Maschinenraum vollbringt diese Wunder meist alleine.

Nach einer hitzigen Diskussion, die die Zimmertemperatur (Sommer, 30 Grad) zumindest gefühlt kräftig steigen ließ, hat er sich auf sieben Tage bis zur Abgabe hochhandeln lassen. Anfang nächster Woche sollten ein Freelancer mit COM-Erfahrung und ein Tester zum Projektteam stoßen. Das bedeutete also, nochmal fünf Tage zusätzlich, minus der Einarbeitungszeit für den Freelancer. Und der Tester? Was kann er testen, wenn noch nichts zum Testen da ist? Zumindest kann er Testfälle und einen Testplan aufstellen.

Bis zum Ende der Woche hatte ich immerhin schon ein Plugin, das sich im Explorer durch ein freundliches „Hello World!“ zu erkennen gab. Zwar war damit immer noch 0% Funktionalität implementiert, aber da ich noch nie ein Explorer-Plugin geschrieben hatte, war ich trotzdem zufrieden. Zu diesem Zeitpunkt wusste ich nicht, dass sich der Freelancer zwar auf MFC-Programmierung verstand, aber von der Plugin-Programmierung keine Ahnung hatte. Außerdem musste ich die unangenehme Erfahrung machen, dass mir einige Untiefen der Explorer-Schnittstelle noch nicht bekannt waren.

Was bedeutet die 80-20-Regel (80% der Funktionalität mit 20% Aufwand), wenn man zwei von sieben Tagen verbraucht, aber noch immer keine Funktionalität implementiert hat? Nichts Gutes. Aber wir kannten keine gescheiterten Projekte. Gescheiterte Projekte waren schlicht keine Option. Es wurde immer geliefert, auch wenn es sich um Bananen-Software handelte, die noch nicht ganz gelb war. Dieses Mal war sie giftgrün.

Was habe ich aus dem Projekt gelernt?

- Man sollte die Anforderungen kennen. OK, die kannte ich, aber hätte ich sie mal lieber hinterfragt: Unser Chef meinte, ein Plugin müsste schneller zu entwickeln sein, als eine vollständige Applikation (ist ja nur ein Teil, muss also schneller zu entwickeln sein). Die Integration in den Explorer, die uns die meiste Zeit gekostet hat, war keine Kundenanforderung! Der Freelancer hat innerhalb von drei Tagen einen funktionierenden Prototyp als MFC-Applikation gebaut, den wir nicht verwendeten, weil wir seinen Source Code nicht in das Plugin integrieren konnten. Hätten wir nicht am Plugin festgehalten, hätten wir eine qualitativ hochwertige Lösung abgeben können.
- Wenn man die Architektur nicht versteht, kann sich das Produkt unvorhersehbar verhalten. Uns war zum Beispiel nicht klar, unter welchen Umständen der Explorer ein Display-Update anfordert. Das hat zu interessanten Flacker-Effekten geführt, weil unser Plugin selbst solche Ereignisse angestoßen hat.
- Zum Testen muss ausreichend Zeit sein.
- Wenn man sich externes Know-how einkauft, dann bitte das passende. MFC ist eher nebensächlich, wenn man ein Explorer-Plugin schreiben möchte.
- Lass dich nicht auf Projekte ein, deren Budget bei 10-20% deiner Aufwandsschätzung liegen. Die fehlenden 80% kann man nicht mit Engagement und Multitasking ausgleichen.

Was haben die oben genannten Punkte mit Testen zu tun? Nicht viel, aber viel mit Qualitätssicherung. Fast jede Entscheidung, die man in einem Projekt trifft, hat Auswirkungen auf die Qualität. Deshalb beschäftigen wir uns im vorliegenden Heft nicht nur mit der Tätigkeit des „Testens“, sondern beleuchten wichtige Aspekte der Qualitätssicherung.

Viel Spaß bei der Lektüre des vorliegenden Heftes wünschen Ihnen der Verlag und die Autoren. ■

Helmut Mestrovic

Teamleiter IBM Rational Technical Sales
(helmut.mestrovic@de.ibm.com)

editorial	3	Testlabor – effizient verwaltet	26
inhaltsverzeichnis	4	Durchgängige Web-Application-Security	27
qualitätssicherung im unternehmen			
IBM Next Generation Testing	5	erfahrungsberichte	
Managed-Test-Services: Herausforderung für den Auftraggeber	9	4CS-Blackbox-Gerätetest komplettiert die IBM Rational Toolkette	31
qualitätssicherung in der praxis			
Testen in Agilen Projekten – Tipps für eine erfolgreiche Umsetzung	12	Testdatenermittlung für Geschäftsanwendungen: SAP-Praxisbeispiel/Erfahrungsbericht	34
Testen Sie noch oder liefern Sie schon?	18	Warum die Tester oft unschuldig sind – häufige Fehler beim Modultest	37
Software testen mit „rechtskonformen“ Daten!	21	inserenten	20
Source-Code-Richtlinien leichter einhalten	23	impresum	30

Liebe Leserinnen und Leser,

Menschen haben die Fähigkeiten sich anzupassen und besser zu werden. Dies ist in Zeiten der Globalisierung auch dringend notwendig, denn wir befinden uns in einem Wettlauf der asiatischen Industrie und der Schwellenländer gegen die etablierten Industrienationen.

Dieser Wettlauf beginnt an jedem Tag von neuem. Wir glauben, dass wir weiterhin in einer guten Startposition sind, wenn es sich um komplexe Themen handelt. Drei Maßnahmen sind aus unserer Sicht besonders wichtig für den Ausgang des Rennens:

- Wir müssen den Trend zur Automatisierung forcieren, um die Kostenschere klein zu halten.
- Wir müssen uns durch höhere Qualität von der Konkurrenz abheben.
- Wir müssen mehr für Daten- und Applikationssicherheit tun.

Aus diesem Anlass haben wir uns entschlossen das Sonderheft „Quality Management“ für Sie zu gestalten. Es umfasst neueste Trends im Bereich Qualitätsmanagement und Testen, sowie Artikel zum Thema Testautomatisierung und Security.

Wir wünschen Ihnen beim Lesen der Fachbeiträge viel Freude und hoffen, Ihnen damit eine neue Perspektive auf das Thema zu eröffnen!



Ihr Helmut Mestrovic

Teamleiter IBM Rational Technical Sales
(helmut.mestrovic@de.ibm.com)



Ihre Susanne Herl

Redaktionsleitung OBJEKTSpektrum
(susanne.herl@sigs-datacom.de)

PS: Wenn wir Ihnen mit dem Sonderheft nicht alle Ihre Fragen beantwortet haben, dann lassen Sie es uns gerne wissen!

Unter <http://www.ibm.com/software/de/rational> können Sie weitere Informationen abrufen.

IBM Next Generation Testing

Seit einigen Jahren unterliegt die Art und Weise, wie Software getestet wird, einem umfassenden Transformations- und Professionalisierungsprozess und heute zählt das Testen von Anwendungen zu den aktuellen Fokusthemen der IT. In diesem Kontext ist der Ansatz zu sehen, den IBM unter dem Leitgedanken Next Generation Testing mit Kunden diskutiert. Dabei beschreibt Next Generation Testing, wie sich das Testen konzeptionell verändern und weiterentwickeln muss, um den gestiegenen Anforderungen an die Professionalisierung beim Testen von Geschäftsanwendungen zu begegnen.

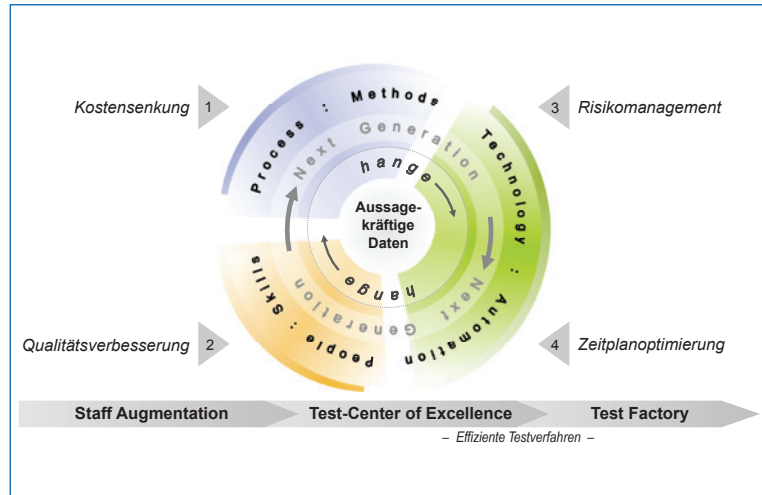


Abb. 1: Next Generation Testing im Kontext der Einflussfaktoren

In der Vergangenheit wurde das Testen von Anwendungen vielfach als notwendiges, aber wenig nutzenbringendes Anhängsel klassischer Softwareentwicklungsprojekte gesehen. Diese Betrachtungsweise hat sich nicht zuletzt vor dem Hintergrund geändert, dass das Testen über den Lebenszyklus einer Applikation betrachtet zwischen 25 und 50 Prozent der Gesamtkosten in Anspruch nimmt.

Aber nicht nur die Kosten des Testens machen ein professionisiertes Testen notwendig. Softwarequalitätsprobleme können schwerwiegende Konsequenzen nach sich ziehen. Ein oft zitiertes Beispiel ist die qualitätsbedingte Projektverzögerung, die in vielen Fällen die Time-to-Market verlängert. Größte kommerzielle und materielle Schäden können entstehen, wenn beispielsweise eine Buchungsplattform aus softwaretechnischen Gründen ausfällt, wichtige Daten verloren gehen oder verfälscht werden oder, wie aktuell den Medien zu entnehmen ist, Sicherheitslücken im System den Diebstahl von Kunden- und Kreditkarteninformationen ermöglichen.

Darüber hinaus ist eine immer engere Verzahnung von Business und IT zu beobachten. Damit steigen die Anforderungen an Time-to-Market und Qualität, was den Leistungsdruck auf die Testorganisationen erhöht. Zugleich werden die Systeme und IT-Landschaften immer komplexer und die Softwareentwicklungs- und -lebenszyklen schneller. Hinzu kommen erweiterte regulatorische und sicherheitsspezifische Anforderungen.

Insofern ist es nur konsequent, dass seit einigen Jahren einerseits die Art und Weise, wie Software getestet wird, und andererseits die Testorganisationen selbst einem umfassenden Transformations- und Professionalisierungsprozess unterliegt und das Testen von Anwendungen zu den aktuellen Fokusthemen der IT zählt.

Das Testen von Geschäftsanwendungen zählt heute zu den Fokusthemen der IT.

In diesem Kontext ist der Ansatz zu sehen, den IBM unter dem Leitgedanken Next Generation Testing mit Kunden diskutiert.

Next Generation Testing

Next Generation Testing beschreibt, wie sich das Testen konzeptionell verändern und weiterentwickeln muss, um den gestiegenen Anforderungen an die Professionalisierung beim Testen von Geschäftsanwendungen zu begegnen. Der Ansatz umfasst den gesamten Testlebenszyklus einer Applikation und adressiert vier Ziele: Kostensenkung, Qualitätsverbesserung, Risikomanagement und Optimierung der Testdurchlaufzeiten. Werden diese vier Ziele nicht in ihrer Gesamtheit berücksichtigt, ergibt sich ein verzerrter Blick auf den Beitrag, den das moderne Testen leisten kann.

Die Erfahrung aus einer Vielzahl von Testprojekten hat gezeigt, dass Next Generation Testing Unternehmen dabei unterstützt, signifikante Kostensenkungen und Qualitätsverbesserungen zu erreichen, Testdurchlaufzeiten zu verkürzen und gleichzeitig die geschäftlichen Risiken zu minimieren.

Im Mittelpunkt von Next Generation Testing stehen Daten. Diese werden gefiltert und analysiert, um daraus detaillierte und verwertbare Informationen zu erhalten. So erlaubt beispielsweise die Analyse der Datenbasis von Softwarefehlern Rückschlüsse auf strukturelle Probleme im Entwicklungsprozess. Eine empirisch untermauerte Kenntnis über eine prognostizierte Fehlerverteilung hilft dabei, den Testprozess effizienter zu managen. Diese und weitere Informationen dienen als Basis zur Ableitung intelligenter Handlungsalternativen und Maßnahmen, die nicht zuletzt zum Einsatz neuer Servicemodelle, Tools und Methoden führen.

Neue Servicemodell und Testinnovationen

Viele nationale und internationale Unternehmen haben erkannt, dass die Professionalisierung des Testens mit einer Transformation der heutigen Organisation einhergeht. In diesem Gestaltungsprozess kommen neue Servicekonzepte, häufig in Form von sogenannten Test-Factory-Modellen, zur Anwendung. Dabei werden aus dem Testportfolio der Kun-

denorganisation exakt definierte Bereich „herausgeschnitten“ und in Form eines industrialisierten „Managed-Service-Modells“ umgesetzt. Diese Serviceleistung wird entweder vom Kunden selbst im Rahmen einer internen Profitorganisation erbracht oder als Auftrag an einen spezialisierten externen Dienstleister vergeben.

Die Ausgestaltung der Test-Factory-Services hängt von den individuellen Kundenwünschen ab und zeigt sich entsprechend vielfältig: Breite der Testverfahren, Skalierbarkeit der Testvolumina, flexible Testinhalte und Abrechnungsmodelle auf Testfallbasis kennzeichnen diese Lösungsmodelle. Darüber hinaus zählen auf den Kundenwunsch zugeschnittene Service-Level-Vereinbarungen zu den wichtigen Bausteinen einer Test-Factory-Implementierung.

Egal welches Servicemodell letztlich zum Einsatz kommt: Im Vordergrund der Betrachtung alternativer Szenarien sollte immer die integrierte Sicht auf Kostensenkung, Qualitätsverbesserung, Risikominimierung und die Optimierung der Testdurchlaufzeiten stehen.

Next Generation Testing hilft Unternehmen dabei, signifikante Kostensenkungen zu realisieren, Qualitätsverbesserungen zu erreichen, Testdurchlaufzeiten zu verkürzen und gleichzeitig die geschäftlichen Risiken zu minimieren.

Die Evaluierung alternativer Servicemodelle spielt – wie aufgezeigt - bei der Transformation der Testorganisation eine sehr wichtige Rolle.

Darüber hinaus muss die moderne Testberatung den Einsatz neuer und innovativer Technologien, Methoden und Tools vor dem Hintergrund einer Kosten-Nutzen-Analyse in die Betrachtung einbeziehen. Die Erfahrung hat gezeigt, dass hier häufig sehr schnelle Erfolge erzielt werden können.

Ein Beispiel für den Einsatz einer innovativen Testtechnologie ist die IBM Testplanning and Optimization Workbench (TPOW). Die TPOW unterstützt eine „risikobasierte“ Testplanung unter expliziter Einbindung empirischer Daten. So hilft beispielsweise eine im Rahmen der TPOW durchgeführte detaillierte Fehleranalyse, systematische Fehlerursachen aufzuzeigen und zu beseitigen. Die Analyse nutzt dabei eine Fülle empirischer Testdaten aus vorangegangenen Testprojekten. Damit können unter anderem Schwächen in wichtigen Risiko- und Qualitätsbereichen pro-aktiv ermittelt werden. TPOW liefert dem Anwender eine optimierte Teststrategie mit Berücksichtigung alternativer, unter Risikogesichtspunkten bewerteter Handlungsempfehlungen, einen dazugehörigen Test-Master-Plan sowie den detaillierten Testprojektplan. Im Ergebnis profitieren die Kunden

von Kosteneinsparungen, einem schnelleren Time-to-Market und einem deutlich höheren Qualitätsniveau.

Ein weiteres Beispiel für eine innovative Testtechnologie bietet das Konzept der „Applikationsvirtualisierung“, das heute verstärkt in die Testdisziplin eingebunden wird. Diese Lösungskomponente basiert auf dem Einsatz spezifischer Software und beseitigt kritische Zugangs- und Kapazitätsengpässe, die beim Testen in heterogenen Umgebungen sehr häufig ein kostspieliges und zeitraubendes Hindernis sind. Im Ergebnis wird die kritische und für den Integrationstest benötigte Zulieferapplikation mittels einer zwischengeschalteten Anwendung simuliert. Der Test kann so unabhängig von der Verfügbarkeit der „echten“ Applikation durchgeführt werden.

Weitere Optimierungen lassen sich durch verschiedene analytisch gestützte Methoden zur verbesserten Testplanung und -durchführung sowie durch kombinatorische Lösungen zur Ermittlung der optimalen Anzahl der Testfälle erzielen.

Auch im Bereich der Testinfrastruktur lassen sich signifikante Verbesserungen realisieren. Cloud-Testing ist hier ein ganz wichtiges Stichwort. Wer heute über Cloud-basierte Testinfrastrukturen nachdenkt, sieht sich automatisch mit dem Thema „Sensibilität der eigenen Testdaten“ und dem damit verbundenen Testdatenmanagement konfrontiert. Insofern ist es nicht verwunderlich, dass das Testdatenmanagement neben der Bereitstellung, Extraktion und Multiplikation von Testdaten auch die Disziplinen Testdatenanonymisierung und -maskierung umfasst. Entsprechende Lösungsangebote und Dienstleistungskonzepte haben sich im Markt etabliert.

Die folgende Grafik liefert einen zusammenfassenden Überblick über das testspezifische Innovationsprogramm von IBM, das unterschiedliche Bereiche des Softwarelebenszyklus adressiert.

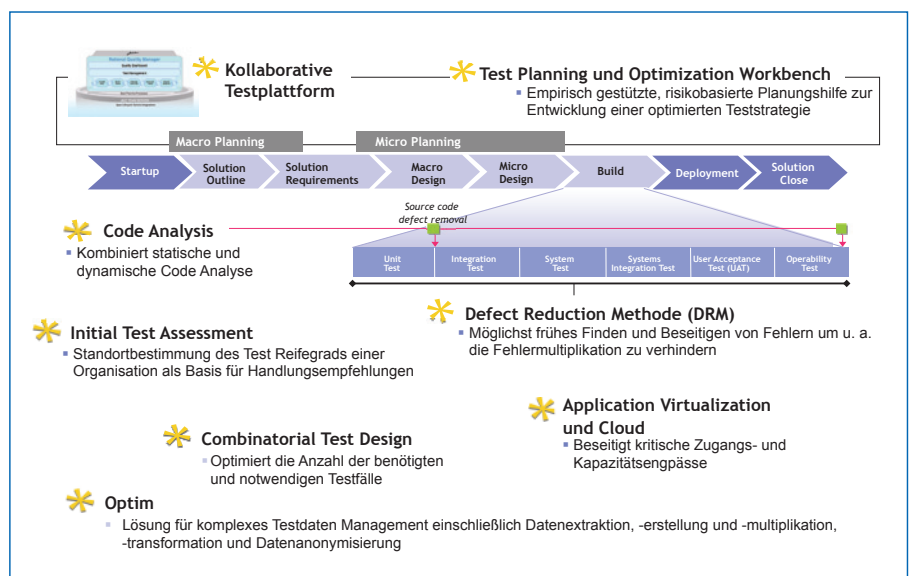


Abb. 2: Einsatz innovativer Produkte in unterschiedlichen Bereichen des Software-Lebenszyklus

Den Nukleus des Testmanagements bildet die „kollaborative Testplattform“.

Die kollaborative Testplattform von IBM Rational Software

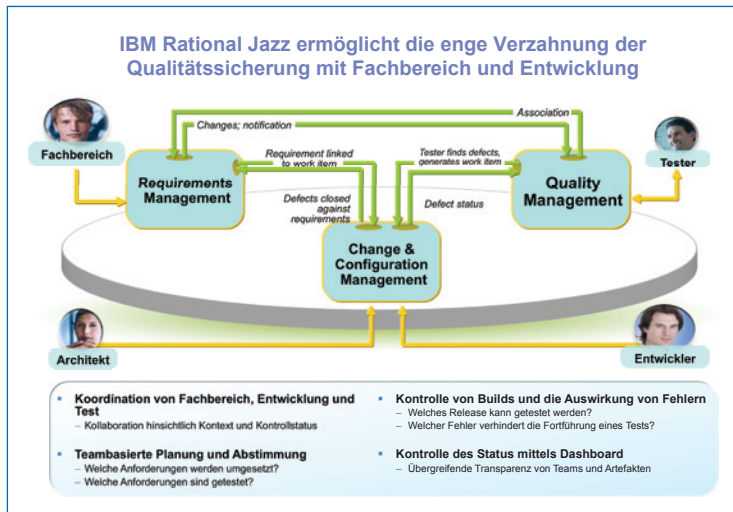


Abb. 3: Zusammenspiel unterschiedlicher Rollen mit der IBM Rational Jazz Plattform

Die Aufgabe der kollaborativen Testplattform besteht darin, sämtliche Testaktivitäten prozessübergreifend zu managen. Die Testaktivitäten umfassen Aufgaben, Arbeitsergebnisse sowie die beteiligten Personen unter Berücksichtigung der individuellen Rollen- und Verantwortlichkeitsbeschreibung. Dabei besteht die Herausforderung nicht zuletzt darin, angrenzende Prozesse wie Anforderungs- und Projektmanagement sowie Softwareerstellung und -deployment in den Gesamtprozess zu integrieren. Unter Verwendung der Jazz Plattform von IBM kann ein solcher übergreifender Ansatz realisiert werden.

Kerntestaktivitäten wie Testplanung, Testautomatisierung sowie Testausführung können dabei in engem Zusammenspiel mit dem Anforderungsmanagement und der Entwicklung sowie der Projektplanung stattfinden. Dabei stehen Aspekte wie Kollaboration, Automatisierung und Transparenz im Vordergrund, die beispielsweise durch Team dashboards, Benachrichtigungsfunktionen und Zugriff via Internet unterstützt werden (siehe Abbildung 3).

Eine wichtige Anforderung an eine Kollaborationsplattform ist die Nachvollziehbarkeit und Verfolgbarkeit von Änderungen, Aufgaben und Arbeitsergebnissen. Die Nachvollziehbarkeit hat entscheidenden Einfluss auf die Qualität und Wartbarkeit der erstellten Software. Durch neue oder geänderte Anforderungen müssen Spezifikationen wie z. B. Dokumente, Modelle, Quellcode oder Testbeschreibungen ständig angepasst werden. Diese Informationen sind voneinander abhängig und müssen daher aufeinander abgestimmt werden, was ohne Werkzeuge und Integrationen nicht gewährleistet werden kann.

IBM Rational Quality Manager

Eine elementare Komponente der kollaborativen Testplattform bildet der IBM Rational Quality Manager. Eine Web 2.0

basierte Oberfläche ermöglicht die teamübergreifende Durchführung des kompletten Testprozesses von der Planung über die Ausführung bis hin zur Auswertung der Testergebnisse.

Mit Hilfe des IBM Rational Quality Manager werden unter anderen folgende Aufgaben durchgeführt:

- Import von zu testenden fachlichen Anforderungen
- Beschreibung von Testplänen und Testfällen
- Erstellung von manuellen Testskripten
- Verwaltung von Testumgebungen und konkreten Testmaschinen
- Ausführung von manuellen oder automatischen Tests
- Analyse der Testergebnisse
- Beschreibung und Übergabe von gefundenen Fehlern
- Erstellung von Reports auf Basis historischer und aktueller Daten

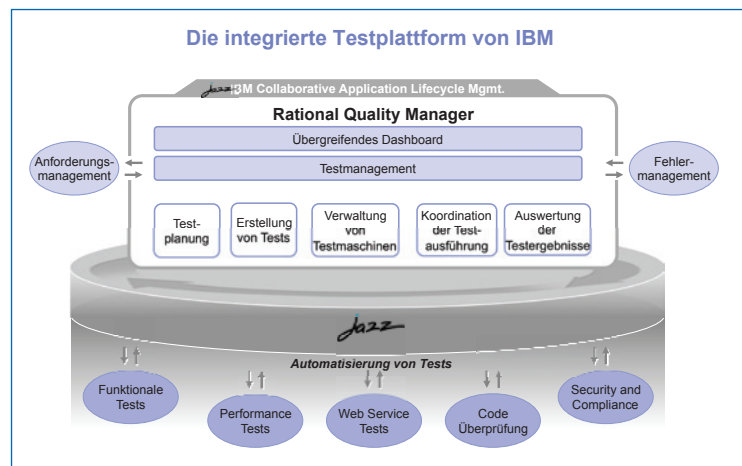


Abb. 4: Funktionen und Integrationsmöglichkeiten IBM Rational Quality Manager

Mit IBM Rational Quality Manager können verschiedenste Testverfahren implementiert werden. Neben dem im Werkzeug selbst vorhandenen manuellen Testeditor, mit dem interaktive Testskripte erstellt und ausgeführt werden können, bietet die Software offene Schnittstellen und Adapter für verschiedene Testautomatisierungswerkzeuge von IBM oder anderen Herstellern. Der Vorteil besteht darin, dass damit Tests für verschiedene Plattformen automatisiert werden können. Diese Offenheit bietet dem Kunden Investitionssicherheit: Bereits vorhandene Testautomatisierungswerkzeuge können in einen vom IBM Rational Quality Manager gesteuerten Gesamtprozess für die Testausführung integriert und weiter genutzt werden.

Für den Zugriff auf Informationen des Rational Quality Manager bietet das Werkzeug „Representational State Transfer (REST)“-basierte Schnittstellen an. Die REST-Services stellen einen plattformunabhängigen und leichtgewichtigen Architekturstil dar, mit dem Ressourcen einfach über HTTP und

XML adressiert und verändert werden können. So lässt sich jede Ressource (wie z. B. ein Testfall) über eine URL (Uniform Resource Locator) eindeutig adressieren.

IBM ist in der Lage, ein umfassendes Systemportfolio für die Planung, Steuerung, Implementierung, Ausführung sowie Auswertung der gesamten Testaktivitäten und -verfahren anzubieten und dabei angrenzende Systembereiche nahtlos zu integrieren.

Wie alle Jazz-basierten Werkzeuge nutzt der IBM Rational Quality Manager ebenfalls die Open Services for Lifecycle Collaboration Spezifikation (OSLC). Darunter versteht man einen herstellerübergreifenden Standard für die domänenübergreifende Kommunikation von Tools unterschiedlicher Hersteller im Application-Lifecycle-Management. Damit besteht für den Anwender die Möglichkeit, Artefakte unterschiedlicher Werkzeuge und Hersteller über Verlinkungen semantisch miteinander zu verbinden. Wird beispielsweise in der Kundenorganisation für die Fehlerverwaltung ein beliebiges Tool mit OSLC-Schnittstelle eingesetzt, kann IBM Rational Quality Manager die Testergebnisse mit den Fehlerbeschreibungen des Werkzeuges verbinden. Der Anwender kann zwischen den verknüpften Informationen von einem Werkzeug in das andere wechseln.

Zusammenfassung

Next Generation Testing beschreibt die Transformation von Testorganisationen. Dabei spielen moderne Verfahren, Technologien und Tools eine entscheidende Rolle, um Kosten

zu senken, die Qualität zu verbessern, Risiken zu managen und die Testdurchlaufzeiten zu optimieren. IBM ist durch die Kombination von IBM Testing Services mit übergreifenden Lösungen von IBM Rational Software für Qualitätssicherung und Testautomatisierung sowie den Erkenntnissen aus IBM Research in der Lage, die neuesten Anforderungen im Bereich Softwarequalität zum Vorteil der Kunden umzusetzen. ■

Thomas Winkeler

seit 2009 als Senior Managing Consultant im Bereich Sales und Solutioning in der Service Line Application Management Services (AMS) der IBM in Deutschland tätig. Seit Januar 2011 mit der Funktion des AMS Sales Leaders für Testing Services betraut. Zuvor in diversen nationalen wie internationalen Großprojekten der Travel und Transportation Industrie als Projekt- bzw. Programmleiter tätig.
(thomas.winkeler@de.ibm.com)

Rene Meyer

ist seit Anfang 2003 im Bereich Rational Software der IBM Software Group als Spezialist für Application Lifecycle Management tätig. In dieser Position verantwortet er im Rahmen zahlreicher Projekte die Einführung und Umsetzung von Lösungen für Anforderungs- und Testmanagement sowie Testautomatisierung für Kunden im Bereich Banken und Versicherungen sowie aus dem Mittelstand.
(rene.meyer@de.ibm.com)

OBJEKTSpektrum

Die Zeitschrift für Software-Engineering und -Management

Online Themenspecials des Jahres 2012



Cloud	26.04.2012
Mobile	24.05.2012
Requirements Engineering	28.06.2012
Testing	27.09.2012
Agility	18.10.2012
Architekturen	15.11.2012

Anmeldung unter: www.sigs-datacom.de/os/themenspecial.htm

Managed-Test-Services: Herausforderung für den Auftraggeber

Mit der Einführung von Managed-Test-Services verfolgen Unternehmen die Strategie, den notwendigen Test der Unternehmensanwendungen professionell und kostengünstig durch externe Dienstleister erbringen zu lassen. Die erfolgreiche Einführung hängt nicht nur von der Wahl des Partners ab, sie stellt auch Anforderungen an das beauftragende Unternehmen. Der folgende Artikel benennt diese Herausforderungen.

Qualitativ hochwertige Software, die die Business-Anforderungen eines Unternehmens optimal unterstützt, ist ein entscheidender Wettbewerbs- und Kostenvorteil in unterschiedlichen Branchen. Bestrebungen der letzten Jahre, die IT-Anwendungslandschaft für aktuelle und zukünftige Herausforderungen im Kerngeschäft eines Unternehmens zu optimieren, hat gesteigerte Anforderungen an einen professionellen Test-Service zur Folge gehabt. Die Analyse der eigenen Fähigkeiten, einen professionellen Test zu managen und durchzuführen, kommt nicht selten zum Schluss, dass professionelles Testen weder zur Kernkompetenz des Unternehmens gehört, noch zu vertretbaren Kosten und mit der notwendigen Qualität durchgeführt wird. Diese Erkenntnisse führen fast zwangsläufig zur Strategie, Test-Services entweder im Hause durch professionelle externe Tester oder, konsequenter, als Managed-Test-Services durch einen externen Dienstleister durchführen zu lassen.

Das Outtasking von Managed-Test-Services scheint bei oberflächlicher Betrachtung auf die Auswahl und die Beauftragung eines erfahrenen und zuverlässigen Partners für diese Dienstleistung beschränkt zu sein. Eine genauere Betrachtung macht aber schnell klar, dass der beauftragte Dienstleister alleine einen effektiven und professionellen Managed-Test-Service nicht sicherstellen kann, sondern dass auch

das beauftragende Unternehmen vor nennenswerten Herausforderungen bei der Einführung steht. Diese können in drei zeitliche Kategorien eingeteilt werden:

- Service-Enablement
- Service-Transition
- Service-Delivery

Service-Enablement

Der Erfolg von Managed-Test-Services wird nicht unwesentlich von den Vorbereitungen des Unternehmens auf das Outtasking der Test-Services bestimmt. Hier gilt es das Unternehmen auf die Anforderungen der Managed-Test-Services nach standardisierten, fabrikmäßigen Abläufen vorzubereiten. Dies betrifft nicht nur Unternehmen, die bisher wenig Erfahrung mit der externen Beauftragung von Testservices haben, sondern auch alle Unternehmen, die bisher über keine standardisierten Softwareentwicklungs- und Wartungsprozesse verfügen. Die folgenden Themengebiete sind diesbezüglich für einen professionellen und effektiven Managed-Test-Service von entscheidender Wichtigkeit:

- Standardisierte Entwicklungsprozesse
- Definierte, qualitativ hochwertige Arbeitsergebnisse im Prozess
- Definierte und planbare Übergabepunkte vom Anforderungsmanagement und der Entwicklung an den Test bzw. vom Test an den Abnahmetest durch die späteren Anwender
- Klares und für alle Beteiligten transparentes Change-Management
- Einheitliches und integriertes Toolset für das Fehler- und Testmanagement
- Externer Zugriff auf die Testumgebungen und -tools

Sollte der zusätzliche Einsatz von globalen Ressourcen im Rahmen der Managed-Test-Services geplant sein, so sind noch die beiden folgenden Punkte zu beachten:

- Durchgängig verfügbare Dokumentation in englischer Sprache (oder einer Sprache, die alle Projektbeteiligten ausreichend beherrschen)
- Schulungen zu kulturellen Unterschieden beim Einsatz von globalen Ressourcen.

Ein für den Managed-Test-Service beauftragter Dienstleister kann bei der Erbringung seiner Services am effizientesten und damit kostengünstigsten arbeiten, wenn auch das beauftragende Unternehmen nach standardisierten Entwicklungs- und Wartungsprozessen arbeitet. Diese müssen über definierte und klar vereinbarte Schnittstellen mit den Prozessstandards des Dienstleisters interagieren. Wichtig sind hier insbesondere klar definierte Verantwortlichkeiten, Inhalt und Qualität der Arbeitsergebnisse, genaue zeitliche Einordnung in den Gesamtprozess und klar definierte Kommunikationsprozesse.

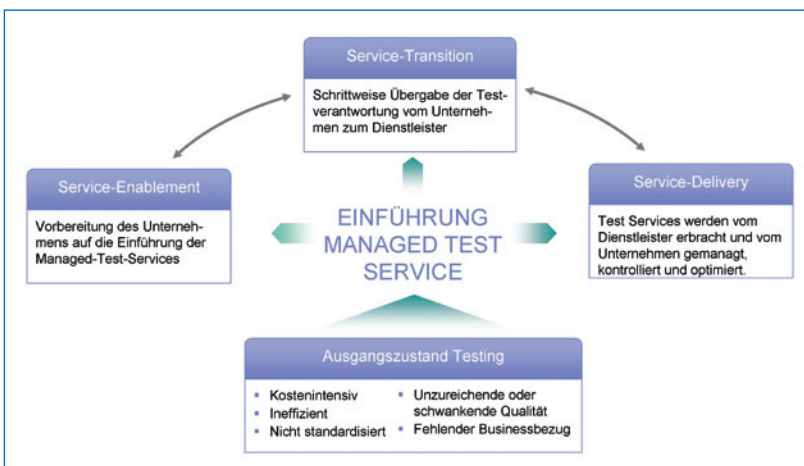


Abb. 1: Übersicht – Einführung von Managed Test Services

Handlungsbedarf seitens des beauftragten Unternehmens ist besonders dann dringend anzuraten, wenn in den Projekten des Unternehmens individuell gestaltete Prozesse zum Einsatz kommen.

Ein Unternehmen, das bereits vor oder während der Ausschreibung der Managed-Test-Services seine Prozesse und Tools standardisiert, erleichtert sowohl die Einführung als auch die Service-Delivery der Managed-Test-Services. Es ist in die Lage, die Qualität der Test-Services beim Wechsel zu einem Managed-Test-Service zu erhalten und die erhofften Qualitäts- und Kostenvorteile schneller zu realisieren. Dies führt für das beauftragende Unternehmen zu insgesamt höheren Kostenvorteilen und erhöht die Akzeptanz für die Auslagerung der Test-Services im Unternehmen.

Service-Transition

Für das beauftragende Unternehmen beginnt die Service-Transition bereits deutlich vor der Beauftragung des externen Dienstleisters. Der Übergang vom bisherigen Modell zum Managed-Test-Service muss geplant werden und notwendige Entscheidungen, wie dieser Übergang erfolgen soll, müssen vorbereitet oder getroffen werden. Ein besonders kritischer Faktor für diesen Übergang stellt die Gewährleistung der vollständigen Kontinuität der notwendigen Test-Services vor, während und nach dem Übergang dar. Hierbei sind vor allem die Zusatzbelastungen der am Übergang beteiligten Mitarbeiter zu berücksichtigen, die durch folgende Aufgaben verursacht werden:

- Aufbau der Governance-Struktur zur Steuerung des Managed-Test-Services hinsichtlich Organisation, Kommunikation, Prozesse, Metriken, Reporting und Service-Level-Agreements
- Fachliche Einweisung des Dienstleisters in die zu testenden Systeme sowie Einführung der durch die Systeme unterstützten Businessprozesse des Auftraggebers
- Übergabe aller für den Test benötigten Projektergebnisse (z. B. Fach- und IT-Konzepte), Testdokumente, Testfälle und offene Fehler
- Übergabe des fachlichen und technischen Know-how für nicht-funktionale Tests wie z. B. Performance- und Sicherheitstests
- Einweisung in bestehende Testautomatisierungen und Übergabe der Testskripte
- Einweisung und ggf. Übergabe von Testarchitekturthemen wie Testkonzeption, Testumgebungen und Testdaten
- Aufbau der notwendigen Infrastruktur wie z. B. externer Zugriff auf die Testsysteme, Bereitstellung der benötigten Test-Clients, Etablierung einer gemeinsamen Tool-Landschaft.
- Einweisung in die bestehenden IT-Prozesse für Softwareentwicklung, Wartung und Betrieb des Auftraggebers

Um die Service-Transition optimal zu gestalten, wird es notwendig sein, die Planung der anstehenden IT-Maßnahmen so anzupassen, dass für die notwendige Service-Transition genügend Raum bleibt und das Risiko für strategisch relevante IT-Maßnahmen beherrschbar bleibt. Insbesondere ein Innovationsstau bei den anstehenden IT-Maßnahmen muss

verhindert werden, um den Aufgaben der IT im Unternehmen auch während des Übergangs gerecht zu werden.

Das beauftragende Unternehmen muss sich auf jeden Fall im Klaren darüber sein, dass die Vorbereitung auf die Einführung der Managed-Test-Services Zeit benötigt, sorgsam geplant werden muss und den verantwortlichen Mitarbeitern und Know-how-Trägern im Unternehmen den notwendigen Freiraum einräumt, diese verantwortungsvolle Aufgabe im Sinne des Unternehmens lösen zu können. Auch bietet sich der Einsatz externer Berater an. Diese sollten die notwendige Erfahrung mit der Einführung und dem Betrieb eines Managed-Test-Services mitbringen, um den Wechsel für das beauftragende Unternehmen professionell zu gestalten.

Auf die eigentliche Ausschreibung, eventuell zweistufig über RFI („Request for Information“) für die Vorauswahl geeigneter Dienstleister und RFP („Request for Proposal“) für das eigentliche Angebot, und die darauf folgende Anbieterauswahl möchte ich in diesem Artikel nicht eingehen. Ich empfehle, diese Phase zu nutzen, um die eigene Vorstellung zur Einführung des Managed-Test-Services zu überprüfen und weiterzuentwickeln. Es ist wichtig, dass bereits in dieser Phase mit dem zukünftigen Dienstleister an einem Entwurf der Service-Transition und der späteren Service-Delivery gearbeitet wird.

Nach der Anbieterauswahl erfolgt dann eine genauere Planung der Service-Transition, auch wird die Ausgestaltung der Service-Delivery im Detail vereinbart. Bedingt durch die Komplexität der Service-Transition ist es notwendig, gemeinsam mit dem Dienstleister die Transition im Detail zu planen und dabei eine gemeinsame und auf die Anforderungen beider Unternehmen abgestimmte Service-Transition zu definieren, zu planen und umzusetzen.

Das beauftragende Unternehmen muss im Rahmen dieser Phase nicht nur die Vorbereitungen des Dienstleisters auf die Übernahme des Managed-Test-Services unterstützen, sondern wird parallel die Governance-Struktur zur Beauftragung und Steuerung des Dienstleisters definieren und aufbauen. Vorhandene Entwicklungsprozesse müssen mit den Test- und Managementprozessen des Dienstleisters integriert werden, Service-Level-Agreements und Metriken zur Steuerung der Testaktivitäten müssen definiert und eingeführt werden. Zusätzlich muss die notwendige Infrastruktur für die zukünftigen Managed-Test-Service zur Verfügung gestellt werden.

Für die eigentliche Übergabe der Testverantwortung an den Dienstleister können verschiedene Methoden Anwendung finden. Dies ist nicht nur durch die bisherige Vorgehensweise des Auftraggebers in der Entwicklung und im Test, sondern auch von der Komplexität und der Phase des Projektlebenszyklus (Einführung, Weiterentwicklung bzw. Wartung eines Systems) abhängig. Es ist daher wahrscheinlich, dass mehrere der folgenden Methoden zum Einsatz kommen:

- Übergabe aller vorhandenen Materialien, technische und fachliche Einführung in das System (für einfache Systeme, die sich in der Wartung befinden oder für neu zu entwickelnde Systeme, siehe Abbildung 2)
- Übergabe der Regressionstests in einer ersten Phase mit anschließender Übergabe der Testaufgaben im Rahmen

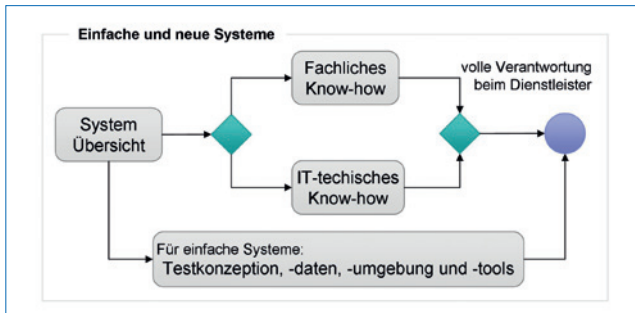


Abb. 2: Übergabe eines einfachen oder neuen Systems.

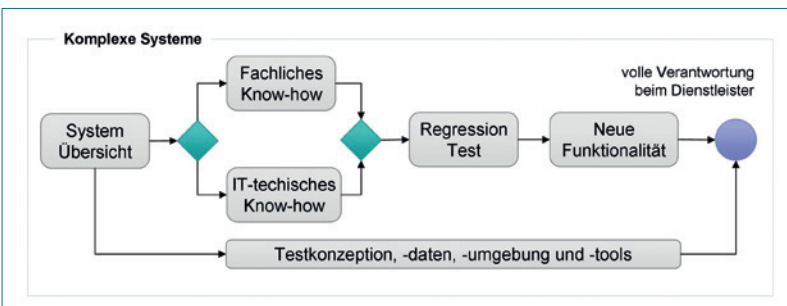


Abb. 3: Übergabe eines komplexen Systems.

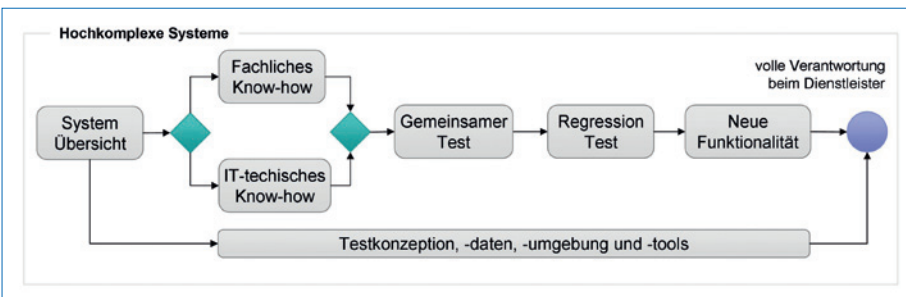


Abb. 4: Übergabe eines hochkomplexen Systems.

Service-Delivery

Der Übergang von der Service-Transition zur Service-Delivery erfolgt in der Regel kontinuierlich, da die Übergabe der Testverantwortung System für System erfolgt. Dieser Umstand erleichtert die Einführung der notwendigen Governance-Strukturen im beauftragten Unternehmen, führt aber zu einer Doppelbelastung hinsichtlich parallel verlaufender Service-Transition- und Service-Delivery-Aufgaben. Auch muss in dieser Phase mit erhöhtem Ressourceneinsatz gerechnet werden, da sich die angestrebte Effizienz der Managed-Test-Services in den Governance- und Test-Prozessen erst mit der Zeit erreichen lässt.

Auch nachdem die Service-Transition abgeschlossen ist und funktionierende Managed-Test-Services eingeführt sind, empfehle ich Maßnahmen zur kontinuierlichen Optimierung der Managed-Test-Services. Die konsequente Nutzung von Methoden wie Audits, „Lessons Learned“, Best Practices und die Einführung neuer innovativer Testmethoden unterstützen die kontinuierliche Verbesserung der Managed-Test-Services.

Zusammenfassung

Die Einführung von Managed-Test-Services kann die Qualität der Test-Services eines Unternehmens auf ein neues Niveau heben. Beim Outsourcing der Test-Services ist neben einem erfahrenen Dienstleister aber auch die professionelle Vorbereitung und Einführung durch das beauftragende Unternehmen von entscheidender Wichtigkeit. Auf das Unternehmen kommen dort Herausforderungen zu, die weit über die vorherigen Aufgaben hinausgehen und damit eine echte Herausforderung darstellen. Um diesen Herausforderungen adäquat zu begegnen, kann der Einsatz erfahrener Berater für Managed-Test-Services den Erfolg gewährleisten. Eine frühzeitige Adressierung wichtiger vorbereitender Tätigkeiten im Rahmen des Service-Enablements und eine professionelle Planung der Service-Transition ermöglichen dem Unternehmen eine erfolgreiche und effiziente Einführung von Managed-Test-Services. ■

der Weiterentwicklung des Systems (für komplexere Systeme, die sich in der Weiterentwicklung oder Wartung befinden, siehe Abbildung 3)

- Übergabe durch eine oder mehrere gemeinsam durchgeführte Testphasen, wobei die Verantwortung Schritt-für-Schritt an den Dienstleister übergeht (für hochkomplexe Systeme in der Weiterentwicklung, siehe Abbildung 4)
- Zusätzliche Varianten, die Abwandlungen der oben beschriebenen Vorgehensweisen darstellen oder spezifische Vorgehensweisen die individuell an die bisherigen Vorgehensweisen in der Entwicklung und im Test angepasst sind.

Bei der Service-Transition muss berücksichtigt werden, dass die Übergabe des fachlichen und IT-technischen Know-hows Zeit benötigt und sorgsam umgesetzt werden muss. Ich empfehle die Service-Transition aktiv zu begleiten und den Erfolg der Übergänge kontinuierlich zu prüfen. Im Rahmen der Service-Transition werden sowohl das beauftragende Unternehmen als auch der Dienstleister eine Lernkurve durchlaufen, die effektiv als Maßnahme zur kontinuierlichen Verbesserung der Übergänge begleitet werden sollte.

Dr. Andreas Richter

leitet die IBM Test Community in Deutschland und ist Experte für die Optimierung von Testprozessen und -organisationen. Er hat Erfahrungen beim Aufbau von Global Delivery Testcentern und deren Integration in die Governance- und SW-Entwicklungsprozesse der Auftraggeber.
(andreas.richter@de.ibm.com)

Testen in Agilen Projekten – Tipps für eine erfolgreiche Umsetzung

Das Testen in agilen Projekten wirft viele Fragen auf: Anforderungen ändern sich, was ist mit den Tests? Welche Tests sind in agilen Projekten überhaupt erforderlich? Mit welchen Strategien werden sie erfolgreich umgesetzt? In Methoden wie z. B. Scrum sucht man vergeblich nach bekannten Testrollen, Testphasen, Teststufen oder Testarten (vom Unit-Test mal abgesehen). Wie lassen sich also Tests in agilen Entwicklungsprozessen erfolgreich einsetzen, ohne das Tempo agiler Entwicklung zu bremsen?

Idealerweise sollte eine geeignete Teststrategie im agilen Umfeld die Qualität erhöhen und auch Ressourcen wie Kosten, Personal oder Zeit sparen.

Kennen Sie das? Sie arbeiten in einem Projekt, das agil sein soll. Es werden selbstverständlich Unit-Tests und Stand-Up-Meetings durchgeführt, und natürlich wird iterativ gearbeitet. Doch wie sieht es mit der Testorganisation und dem Testmanagement aus? Formale Teststufen und Testphasen gibt es eigentlich nicht. Oder Ihr Projekt hat alle Teststufen und Testphasen ordentlich vor und hinter Dingen, die sich Builds und Sprints nennen, geplant und arbeitet deshalb natürlich auch agil? Projekte agil durchzuführen ist nicht neu. Testen auch nicht. Aber in agilen Projektmanagementmethoden wie Scrum gibt es keine Hinweise, wie Tester und Tests eingesetzt werden. Die folgenden Punkte geben Tipps für erfolgreiche agile Teststrategien. Insbesondere im komplexen IT-Umfeld lang laufender oder auch umfangreicher IT-Projekte werden dadurch die Vorteile agiler Entwicklung auch im Test nutzbar. Die Tipps haben sich in mehreren agilen IBM-Projekten als Schlüssel für den erfolgreichen Test herausgestellt.

Tipps für die Testvorbereitung:

TIPP: „Whole Team“ mit „Generalisierten Spezialisten“

Klassische Strukturen mit Entwicklungsteams und anschließend parallel arbeitenden Teams für z. B. Fachtests und IT-Tests funktionieren prinzipiell nicht in einem agilen Umfeld, da schnelle Entscheidungen und Aktionen nicht umgesetzt werden können. Entscheidend ist dabei der „Whole Team“-Gedanke [CRI09], der erfordert, dass jedes Teammitglied seinen speziellen Beitrag zum Gesamterfolg und zur Produktivitätssteigerung des Teams leistet [BEC04]. Nur in einem Team kann Einigkeit zur Erreichung des gemeinsamen Ziels bestehen. Dies erfordert die Einbindung der Testmanager, Testarchitekten und Tester in das selbstbestimmende Team und die täglichen Stand-Up-Meetings. Spezialisierte Tester können z. B. mit den Entwicklern zusammenarbeiten und sollten in Planungs-, Design- und Schätzaktivitäten eingebunden werden. Idealerweise führt auch nicht jeder nur die Aufgaben der ihm zugeordneten Rolle aus, sondern

schaut über den eigenen Tellerrand. Das heißt, Tester dürfen sich auch an der Entwicklung oder Architektur beteiligen. Die Tester sind somit ebenfalls und im besten Fall als sogenannte „Generalisierte Spezialisten“ einsetzbar [AMB11].

TIPP: Tester und Entwickler im Paar

Es empfiehlt sich schon zu Beginn jeder Iteration Entwickler und Tester für einzelne Stories zu „paaren“. Somit wissen Entwickler, wer für den Test zuständig ist und Tester wissen, wen sie im Fall von Problemen direkt kontaktieren können. Folgende Vorteile ergaben sich daraus:

- Besseres Verständnis der Use-Cases bei Testern und Entwicklern durch unterschiedliche Sichtweisen
- Zeitersparnis im Fehlermanagement
- Besseres Vertrauensverhältnis von Entwicklern und Testern und dadurch auch bessere Zusammenarbeit
- Dokumentation konnte weniger detailliert und agiler gestaltet werden
- Beschleunigte Testfallerstellung und Bereitstellung von Testergebnissen

TIPP: Test und Entwicklung sollten geografisch nicht getrennt sein

Geografische Verteilung führt leicht zu „Wir gegen die Anderen“-Spielchen. Ansätze über eine zentralisierte Testorganisation oder sogar eine ausgelagerte Test-Factory funktionieren nur bedingt in einem agilen Umfeld. Das bedeutet jedoch nicht, dass alle Beteiligten im gleichen Zimmer sitzen müssen. Es ist jedoch von großem Vorteil, den „Whole Team“-Gedanken auch geografisch umzusetzen. Ein Team kann besser eine eigene „Persönlichkeit“ entwickeln, wenn Tester und Entwickler für eine Story gepaart am selben Ort arbeiten. Auch arbeitet das Team dann deutlich produktiver. Die Prinzipien des Pair-Programming-Ansatzes [BEC04] können so auch für andere Disziplinen wie Modellierung, Planung, Anforderungsanalyse und eben auch bestens für das Testen genutzt werden [WIL02].

TIPP: Agiles Verhalten von Testern

Tester brauchen in agilen Projekten eine „Just-Do-IT“-Mentalität. Die Wertschätzung der Tester in einem agilen Team entsteht nur, wenn diese helfen, etwas zu verbessern oder zu vervollständigen. Daran müssen sich die Tester auch in jeder Projektphase eigenständig erinnern.

TIPP: Schaffen Sie Ihre eigene agile Methode

Jedes Projekt hat per se einen eigenen Entwicklungsprozess. Auch wenn Projekte scheinbar keine Methoden haben, setzen sie dennoch eine ein, die eventuell nicht formal benannt werden kann oder gar dokumentiert ist. Die eine, erfolgversprechende agile Methode, die für jedes Team und jedes Projekt passt, gibt es jedoch nicht. Deshalb ist es notwendig, aus bewährten Vorgehensweisen wie z. B. Scrum, Test Driven Development, eXtreme Programming und speziell für Test auch aus dem Full-Lifecycle-Testing-Ansatz, die richtigen Ergebnistypen und Aktivitäten auszuwählen und an das eigene Vorhaben anzupassen. Dies geschieht typischerweise in einem Workshop zu Projektbeginn.

- Welche Inspektionen und Reviews werden benötigt? Für Stories, Testfälle, Architektur, Code usw.?
- Welche Standards z. B. für Code-Design sind sinnvoll?

Unterstützung im Softwareprojekt



Jeden Tag sind Sie aktiv in Softwareentwicklungsprojekten involviert und **managen** sie. Wie erläutern Sie Ihrem übergeordneten Projektverantwortlichen die neuesten Trends und belegen die Wirtschaftlichkeit der richtigen Technologie? **OBJEKTSpektrum** unterstützt Sie bei der Argumentation und liefert Ihnen wertvolle **Anregungen für den Alltag des Software-Engineerings**.



JavaSPEKTRUM ist das Praxismagazin für professionelle Softwarearchitekten und Entwickler, die mit der Java-Plattform arbeiten. Das Heft stellt neue Entwicklungen und Konzepte vor, prüft die Relevanz für den täglichen Einsatz. Es werden aktuelle Themen wie **Aspektororientierung, Agilität, Eclipse, Embedded** vertieft. Wie Java mit anderen Plattformen in der Praxis effizient interagieren kann, erfahren Sie in der Rubrik „**Integrationspektrum**“.

Alle Jahresabonnenten erhalten einen **persönlichen Login** zu allen Artikel-pdfs. Wir benötigen hierfür unbedingt **Ihre e-Mail-Adresse**.

Bitte schicken uns diesen Bestellschein ausgefüllt **per Fax (+49 (0)2241/2341-199)** zurück oder gehen Sie online unter **www.sigs.de/publications/aboservice.htm**

+++ Jetzt bestellen +++

Hiermit bestelle ich ein **Mini-Abo mit 3 Ausgaben für nur 15,00 €:**

OBJEKTSpektrum* JavaSPEKTRUM*

*wandelt sich automatisch in Jahresabo um, wenn nicht 6 Wochen vor Ende des Bezugszeitraums gekündigt wird

ein Jahresabonnement:	Deutschland	Europa	Auß. Europa
OBJEKTSpektrum (OS)	<input type="checkbox"/> 50,90 €	<input type="checkbox"/> 58,60 €	<input type="checkbox"/> 63,60 €
JavaSPEKTRUM (JS)	<input type="checkbox"/> 36,60 €	<input type="checkbox"/> 42,00 €	<input type="checkbox"/> 49,00 €
Kombi-Abo (OS+JS)	<input type="checkbox"/> 81,30 €	<input type="checkbox"/> 95,40 €	<input type="checkbox"/> 105,00 €

ein kostenloses Probeheft

OBJEKTSpektrum JavaSPEKTRUM

Firma: _____ Abteilung: _____
 Name: _____ Vorname: _____
 Straße: _____ L/PLZ/Ort: _____
 Telefon: _____ Telefax: _____
 e-Mail: _____ Position: _____

Zahlungswunsch: per Rechnung per Kreditkarte – wir rufen Sie an unter Tel.-Nr.: _____
 per Bankeinzug Kto.-Nr.: _____ BLZ: _____ Bank: _____

Datum: _____ Unterschrift: _____

- Welche Praktiken werden eingesetzt: Scrum, Agile with Discipline (AWD), Test Driven Design (TDD), Pair Programming...?

Hierbei sollten auch die Werte aus dem Manifest für Agile Softwareentwicklung stets für die Auswahl maßgebend sein und nur ein Minimalset an Ergebnistypen festgelegt werden. Das heißt:

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als umfassende Dokumentation
- Reagieren auf Veränderungen ist wichtiger als das Befolgen des Projektplans

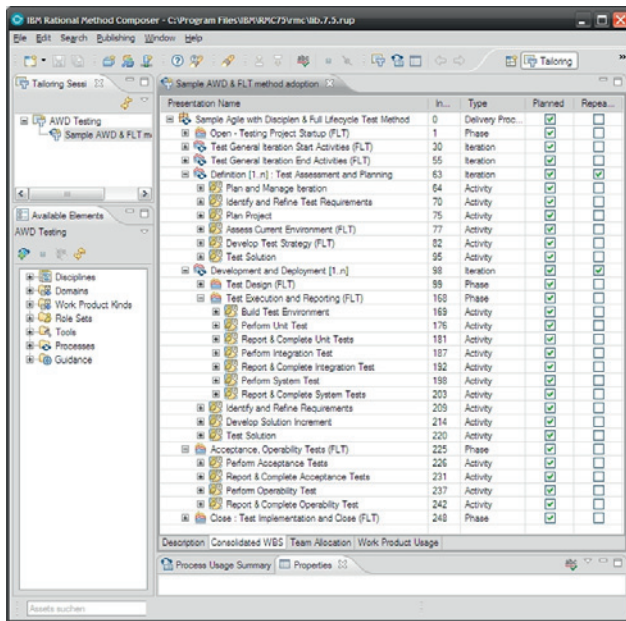


Abb. 1: Beispiel agiler Methoden Anpassung

Die Abbildung 1 „Beispiel agiler Methoden Anpassung“ stellt eine angepasste Kombination der folgenden Methoden dar:

- IBM Agile with Discipline (AWD)
- Full Lifecycle Testing (FLT)
- World Wide Project Management Method (WWP-MM)

Die Anpassung erfolgte mit Hilfe des IBM Rational Method Composers.

TIPP: Gleiche Sprache und gleiches Verständnis

In agilen Methoden sind Rollen wie Testmanager oder auch Tester häufig nicht beschrieben. Jeder ist auch Tester – aber kennt auch jeder die damit verbundenen Tätigkeiten und deren Bedeutung für das Projekt? Nein, die nötigen und zu verwendenden Teststufen, Testarten und Testtypen sind Entwicklern und Projektleitern häufig nicht bekannt. Deshalb hilft es auch im agilen Umfeld eine Teststrategie zu erstellen, die erklärt wie Tests im Projekt durchgeführt werden. Der Hauptfokus liegt dabei aber nicht auf der Prozessbeschreibung, sondern in der Erklärung der testspezifischen Begriffe für das Projekt.

Die Begriffe Testphase, Teststufe und Testart sorgen häufig für Verwirrung. Zum Beispiel sollte der Begriff „Testphase“ nicht für Integrationstests oder Systemtests verwendet werden, sondern wenn überhaupt, dann nur für Testvorbereitung, Testerstellung und Testdurchführung innerhalb einer Iteration. Bei den Teststufen handelt es sich um Ebenen der physikalischen Integration oder der Qualität, die nach wie vor nötig sind, um sicherzustellen, dass sie Software den Anforderungen genügt. Die Stufen werden aber in jeder Iteration ohne zeitliche Abhängigkeit durchlaufen. Dabei verfolgen die Stufen folgende Ziele:

1. Unit-Test: Entwickler testen ihren Code mit dem Ergebnis, dass der Code das tut, was der Entwickler wollte
2. Entwicklerintegrationstest: Entwicklerteams testen, ob alles im Sinne laufender Software zusammenpasst
3. Systemtest: Tester bzw. Fachexperten testen, ob ihre angeforderten Anwendungsfälle damit funktionieren
4. Systemintegrationstest: Tester bzw. Architekten prüfen, ob die Lösung auch im Zusammenspiel mit anderen Systemen noch funktioniert
5. Akzeptanztest: Stakeholder prüfen die Anwendung auf die geforderten Anforderungen
6. Inbetriebnahmetest: Überprüfung, ob die Anwendung so in Betrieb gehen kann. (zum Beispiel Betriebsanleitung vorhanden, funktionierende Installationsanleitung...)

Auch wenn diese Teststufen nicht konsequent geplant werden, so ist doch jeder Fehler einer dieser Stufen zuzuordnen, da es sich um Qualitätsstufen handelt, die durch den Softwareerstellungsprozess zwangsläufig durchlaufen werden (manchmal leider ohne zugehörige Tests). Es ist auch bekannt, dass ein Fehler, je später er entdeckt wird, umso teurer in der Behebung ist. Die Stufen sind de facto vorhanden, egal wie agil ein Softwareentwicklungsprozess gestaltet oder welche Methodik dafür verwendet wird. Erst durch die Einführung aller Stufen auch in frühen Iterationen, können durch früher gefundene Fehler Kosten gespart werden.

Der Begriff Testart dient der Klassifikation konkreter struktureller oder funktionaler Tests wie z. B. Last- und Performance-

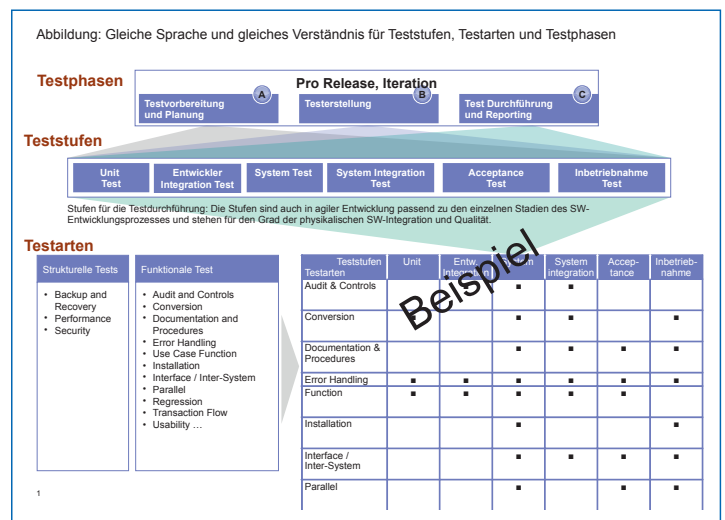


Abb. 2: Gleiche Sprache und gleiches Verständnis

Tests, Backup- und Recovery-Tests, Black-Box-Tests usw. und leitet sich aus dem Testziel ab. Eine Matrix, welche Testarten für welche Teststufen vorgesehen sind, kann helfen, die Tests für eine anstehende Entwicklungsiteration zu planen (siehe Abbildung 2 „Gleiche Sprache und gleiches Verständnis“).

Tipps für die Testerstellung und Planung

TIPP: Produktqualität bestimmt Testumfang und Zeitplan

Über den korrekten Testumfang und den Zeitplan wird der Erfolg eines agilen Projektes sichergestellt. Eine Evaluierung der Produktqualität vor jeder Iteration ist ein guter Weg, um den geeigneten Testumfang zu finden. Agilität ist kein Wundermittel und ändert nichts am „Magischen Dreieck“, das zeigt, dass eine Änderung an einer der Steuergrößen:

- Erwartung der Stakeholder
- Projektdauer bzw. Termine
- Kosten
- Inhalt, Umfang und Qualität des Projekts und seiner Ergebnisse

automatisch zu Änderungen an einer oder mehreren Größen führt [WIK06].

Wenn es früher in Wasserfallprojekten aufgrund von geänderten und zu umfangreichen Anforderungen zu Terminverschiebungen gekommen ist, so kommt es heute zu Einschränkungen in der geforderten Funktionalität oder Erwartungshaltung, aber das Projekt schließt wenigstens pünktlich. Da agile Projekte extrem zeitgesteuert arbeiten und Termine grundsätzlich einhalten, kommt dem Test eine größere Bedeutung zu. Nur über ihn werden auch negative Testergebnisse, wie fehlende oder fehlerhafte Funktionalitäten, wieder in die Umsetzungsliste (Backlog) für kommende Iterationen eingebracht.

TIPP: Plane Regressionstests

In jeder Iteration sollten passend zu neuen Features Regressionstests geplant werden. Gleichzeitig sollten zu gefundenen Fehlern ebenfalls Regressionstests für laufende oder künftige Iterationen definiert werden. Die Regressionstests müssen aktiv für unterschiedliche Ebenen wie Builds, Iteration (Sprint) und Release verwaltet werden. Dabei ist auf eine Maximierung der Abdeckung und Minimierung der Durchlaufzeit zu achten.

TIPP: Investiere in Automatisierung

Testautomatisierung ist zwingend erforderlich, um die nötige Testquantität und Qualität zu erreichen. Nützlich ist sie aber nur, wenn dadurch auch Aufwand oder Zeit gespart wird. Deshalb ist es sinnvoll, klein anzufangen, d. h. nur positiv getestete Fälle zu automatisieren, die auch direkt für Regressionstests sinnvoll verwendet werden können. Es ist darauf zu achten, dass die Geschwindigkeit der Entwicklung nicht gebremst wird oder tägliche Builds nicht gefährdet werden. Allerdings sollte auf Dauer alles, was automatisiert werden kann, auch automatisiert werden.

TIPP: Beschreibe Akzeptanztests und Systemtests statt umfangreiche Anforderungen und Konzepte.

Test First oder Test Driven Development (TDD) lässt sich für alle Teststufen nutzen. Ursprünglich wurde TDD von Kent

Beck [BEC03] nur für Unit-Tests entwickelt. Die Methodik lässt sich aber auch hervorragend für Systemtests, Systemintegrationstests und Akzeptanztests einsetzen, die dann den wesentlichen Teil der Spezifikation ausmachen [AMB10] und sogar andere Formen der Spezifikation überflüssig machen können. Kleiner Nebeneffekt: Entwickler haben so auch erstmalig in der IT-Geschichte eine Motivation, dass die Spezifikationen aktuell sind. Schließlich werden sie ja in Form von Tests direkt zur Überprüfung der Umsetzung genutzt.

Tipps für die Testdurchführung

TIPP: Testdurchführung in derselben Iteration wie Entwicklung

Alle durchzuführenden Testarten sollten in derselben Iteration wie die Entwicklung stattfinden und nicht in der nachfolgenden für vorausgehende Iterationsergebnisse. Hierzu ist es erforderlich, Vorgehen und Zeitplan mit Entwicklern und Testern zu definieren. Die nötigen Testfälle sind Teil der priorisierten Aufgaben (Backlog) und werden für dieselbe Iteration geplant. Das Durchführen der Iterationen oder Sprints als Mini-Wasserfälle (erst Entwicklungsiteration, dann Test, dann Fehlerbehebung) muss unbedingt vermieden werden.

Je nach festgelegtem agilen Entwicklungsmodell können die Teststufen den Entwicklungsstufen zugeordnet werden (siehe Abbildung 3 „Teststufen und Entwicklungsstufen“).

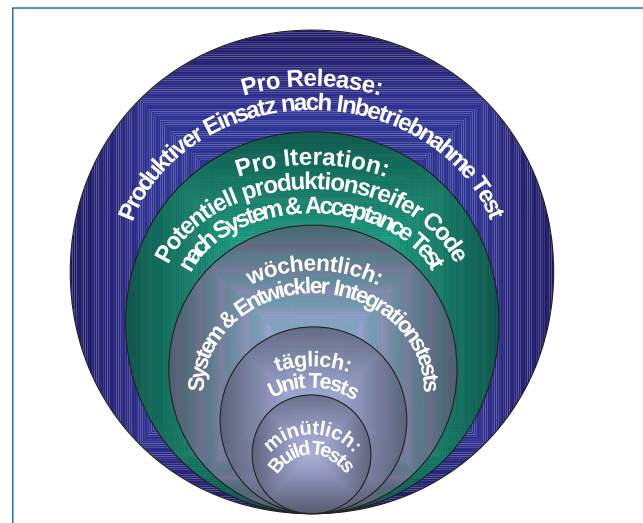


Abb. 3: Teststufen und Entwicklungsstufen

TIPP: Stelle sicher, dass „Done“ auch den Test beinhaltet

Zwischen Entwicklung und Test müssen gemeinsame Kriterien für „Done“ etabliert werden. „Done“ bedeutet auch, dass alle geplanten Tests der Iteration erfolgreich im Sinne vereinbarter Kriterien für Design, Kodierung und Test durchgeführt wurden. Nur so kann das Iterationsergebnis (theoretisch) dem Kunden zur Verfügung gestellt werden.

Im Allgemeinen gelten dabei folgende Kriterien für „Done“:

- Design und Code für geplante Aufgaben sind fertig.
- Keine schwerwiegenden Fehler und keine anderen Fehler, die im Team nicht angenommen wurden. Test ist nicht „Done“, solange bestehende Fehler nicht gelöst sind. Die

Fehler werden dabei priorisiert und wieder in die Aufgabenliste aufgenommen.

- Code ist getestet (Unit, Funktional, System, Performance, End-To-End...).
- Akzeptanz wurde sichergestellt.

TIPP: Nutze parallel unabhängiges Testen

In komplexen Vorhaben ist der „Whole Team“-Ansatz alleine und Test Driven Development (TDD) nicht ausreichend. Durch die agile Arbeitsweise und die damit verbundene Zusammenarbeit aller Beteiligten werden Anforderungen besser verstanden. Aber Anforderungen ändern sich, wurden vergessen, oder lassen sich noch nicht festlegen (wie z. B. bei nichtfunktionalen Anforderungen). Weitere Probleme wie z. B. durch End-to-End-Integration, verwendete Technologien oder Performanz sind noch nicht verstanden. Hier hilft der Einsatz eines unabhängigen Teams, das parallel auf diese Probleme testet [AMB10]. Es versucht diese Fehler zu finden und als Änderungsaufgaben wieder in das Team zu bringen.

Dies scheint ein Widerspruch zum „Whole Team“-Gedanken zu sein. Ist es auch! Es ist erforderlich, dass ein unabhängiges, zentralisiertes Testteam das Team ergänzt, um bestimmte Aufgaben parallel zur laufenden Entwicklung zu übernehmen. Dies betrifft zum Beispiel die folgenden Testarten:

- Integrationstests vor Produktivsetzung
- Last- und Performanztests
- End-Of-Lifecycle-Tests
- Tests in komplexen verteilten Umgebungen
- Tests mit tiefem Spezialwissen wie z. B. korrekte Umsetzung gesetzlicher Anforderungen oder Aufdecken von Sicherheitslücken im Source Code.
- Usability-Tests
- Tests mit besonderen Berechtigungen auf Testdaten
- Test nichtfunktionaler Anforderungen
- und viele andere...

Im Gegensatz zu traditionellen unabhängigen Testteams, ist dieses „unabhängige“ Team in einem agilen Umfeld für spezielle Aufgaben zuständig, die nur einen kleinen Teil des Gesamttests abdecken. Der überwiegende Aufwand in der Testdurchführung verbleibt im „Whole Team“.

TIPP: Änderungsmanagement statt Anforderungsmanagement und Fehlermanagement

Wer kennt das nicht: In vielen Projekten gibt es endlose Diskussionen, ob gefundene Fehler überhaupt echte Fehler sind oder nicht vielleicht doch neue oder bisher falsch spezifizierte Anforderungen. Sind neue Anforderungen aber nicht vielleicht sogar schon Fehler, weil es sich um nicht umgesetzte Funktionalitäten handelt? Wie auch immer, diese politischen Diskussionen haben in der agilen Entwicklung nichts verloren. Es muss Einigkeit im Umgang mit gefundenen Fehlern geben. Einigen wir uns also auf den übergeordneten Begriff „Änderung“, oder auch gleich in agiler Sprache „Change Story“, die – aus welchem Grund auch immer – irgendwann umgesetzt werden muss. Fehler können so als eine Art Anforderung betrachtet werden, die wie andere Anforderun-

gen priorisiert, geschätzt und in den Backlog für die spätere Umsetzung aufgenommen werden. Fehler (oder Änderungen), die direkt behoben werden können, sollten diesen Prozess aber nicht durchlaufen. Das heißt, nur komplexe Fehler kommen in den Backlog. Als Folge daraus werden Anforderungsmanagement und Fehlermanagement in agiler Entwicklung am besten in Form eines guten Änderungsmanagements umgesetzt.

Anmerkung: Die Diskussion, ob es sich um Fehler oder Neuanforderung handelt, ist oft vertraglich begründet, weil Neuanforderungen z. B. nicht über einen vereinbarten Festpreis abgegolten sind. Für den agilen Ansatz bedeutet das aber lediglich, dass die Diskussion dann trotzdem geführt werden muss, und bestimmte Änderungen einen Hinweis über die extra zu bezahlende Umsetzung erhalten [AMB07].

Agile Teststrategien: Tipps für eine erfolgreiche Umsetzung

Testvorbereitung

- „Whole Team“ mit „Generalisierten Spezialisten“
- Tester und Entwickler im Paar
- Test und Entwicklung nicht geografisch getrennt
- Agiles Verhalten von Testern
- Schaffen einer eigenen agile Methode
- Gleiche Sprache und gleiches Verständnis für Tests

Testerstellung und Planung

- Produktqualität bestimmt Testumfang und Zeitplan
- Plane Regressionstests
- Investiere in Automatisierung
- Beschreibe Akzeptanztests und Systemtests statt umfangreiche Anforderungen und Konzepte

Testdurchführung

- Testdurchführung erfolgt in derselben Iteration wie Entwicklung
- Stelle sicher, dass „Done“ auch Test beinhaltet
- Nutze parallel unabhängiges Testen
- Änderungsmanagement statt Anforderungsmanagement und Fehlermanagement

Fazit

Die aufgeführten Tipps sind natürlich nicht der Weisheit letzter Schluss. Sie können aber im Sinne einer Checkliste verwendet werden, um die Ursache für Probleme im Test agiler Anwendungsprojekte zu finden. Auch dienen sie als Denkanstöße, den eigenen Entwicklungsprozess zu verbessern. Tester, Testmanager und Testarchitekten, die in agilen Entwicklungsprozessen eingebunden sind, werden häufig mit einer neuen Arbeitsweise konfrontiert. Diese ist durch Folgendes geprägt:

- intensivere Zusammenarbeit, die mehr Kommunikation anstatt des üblichen Dokumentenaustauschs erfordert
- kürzere Arbeitszyklen
- Reaktion auf sich verändernde Testanforderungen, die mehr Flexibilität erfordert Spezifikationen werden nicht mehr „fertig“ an Tester übergeben, sondern der Test kann sogar die Anforderungsspezifikation darstellen.

- Tester zu sein ist keine ausreichende Qualifikation mehr. Um den „Whole Team“-Gedanken umzusetzen, müssen sie wie auch Entwickler und Architekten zu flexiblen, generalisierten Spezialisten werden, die auch andere Aufgaben übernehmen.

Bestehende Testtechniken können für die agile Entwicklung genutzt werden. Die Herausforderung besteht hauptsächlich darin, sie bewusst und vor allem gewollt an die neue Methodik anzupassen. ■

Literatur

[AGI01] Das agile Manifest:

<http://www.agilemanifesto.org/>, 2001

[AMB07] Ambler, Agile on a Fixed Budget, Dr. Dobb's Journal, September 2007,

[AMB10] Ambler, Agile and Independent Testing,

https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_and_independent_testing25, 2010

[AMB11] Ambler, Agile Software Requirements:

Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series), Addison Wesley, 2011

[BEC03] Beck, Test Driven Development by Example, Addison-Wesley, 2003

[BEC04] Beck, Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley, 2004

[CRI09] Crispin, Gregory, Agile Testing – A Practical Guide for Testers and Agile Teams, Addison-Wesley, 2009

[WIL02] Williams, Pair Programming Illuminated, Addison-Wesley, 2002

[WIK06] Wikipedia, Projektmanagement,

<http://de.wikipedia.org/wiki/Projektmanagement>

Axel Loser

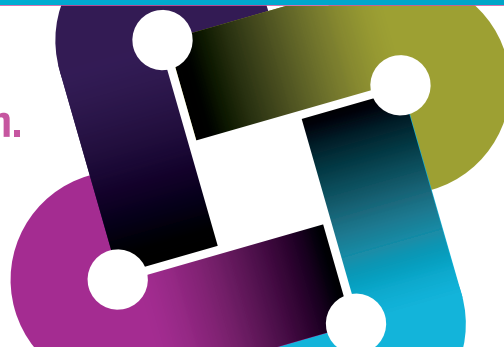
IBM GBS Testing Competence Lead,
Application Innovation Services

(axel.loser@de.ibm.com)

Intelligentes Business braucht intelligente Software.

Für alle, die Software entwickeln wollen. Nicht Probleme.

Entdecken Sie agile Softwareentwicklung mit IBM Rational!



Ganz gleich, ob Ihr Entwicklungsteam über viele Länder verteilt und in verschiedenen Zeitzonen agiert oder lokal in unterschiedlichen Abteilungen und mit abweichenden Projektplänen arbeitet: IBM Rational Team Concert für die agile Softwareentwicklung bietet Ihnen vielfältige Optimierungsmöglichkeiten für die Zusammenarbeit und um Kosten zu sparen.

Denn die integrierte Software vereint vielfältige Tools in einer interaktiven, einfachen und agilen Entwicklungsumgebung.

Finden Sie heraus, wie agile Softwareentwicklung in den Kontext unternehmensweiter IT-Prozesse passt und wie sie in verteilten Teams funktioniert: ibm.com/software/de/rational/agility

Hier erfahren Sie zudem, wie eine agile Lizenzierung auf die Ansprüche von agilen Unternehmen angepasst werden kann. Freuen Sie sich auf spannende Fachbeiträge und lesen Sie, was die Experten zu dem Thema schreiben.

Testen Sie noch oder liefern Sie schon?

Jede Reise beginnt mit dem ersten Schritt. Wenn das Ziel Ihrer Projektreise effektives Qualitätsmanagement einschließt, sind die Anforderungen der erste Schritt auf dem Weg dahin. Die Einbeziehung der Anforderungen in den kompletten Lebenszyklus ist daher zwingende Voraussetzung für einen wirksamen Qualitätssicherungsprozess. Die Beschäftigung mit der Zieldefinition eines Entwicklungsprojektes ist mit der Erhebung und Auswahl der Anforderungen keineswegs abgeschlossen. Erst die Verankerung der Theorie (der Anforderungen) mit der Praxis (den verschiedenen Phasen und Disziplinen des Entwicklungsprojektes) ermöglicht durchgängiges Management und stringente Verfolgung der Anforderungen durch den kompletten Lebenszyklus und sichert den kontinuierlichen Abgleich zwischen den wirtschaftlichen Zielen, operationellen Gegebenheiten, technischen Rahmenbedingungen und Erwartungen des Marktes oder des Kunden.

Entwicklungsprojekte sind heute typischerweise in Szenarien eingebettet, welche die Manifestation von Problemen und potentiellen Risiken geradezu provozieren. Geographisch verteilte Projektteams erschweren die Zusammenarbeit, die Kommunikation und die Transparenz des Projektfortschritts. Heterogene lokale Gebräuche können die sinnvolle Anwendung von Lifecycle übergreifenden Prozessen behindern. Die Dynamik wirtschaftlicher oder technologischer Entwicklungen kann überstürzte Änderungen von Projektvorgaben zur Folge haben und resultiert zudem in schnell zunehmender Komplexität der Anwendungen. In dieser Situation kann sich die zusätzliche Aufgabe, die Einhaltung behördlicher Auflagen wie z. B. Basel II, Sarbanes-Oxley, etc. nachzuweisen, als nahezu unlösbar erweisen.

Die auf den ersten Blick einfache Aufgabe, nämlich das zu entwickeln, was der Kunde erwartet, kann vor diesem Hintergrund zu einer echten Herausforderung werden. Ihre Kunden formulieren eine Anfrage, aber irgendwo unterwegs, in diesem Strudel aus komplizierten System- und Softwarestrukturen, Kommunikationslücken, mangelnder Transparenz über verschiedene Teams und Phasen hinweg, geht die Originalanfrage verloren oder wird verändert oder falsch interpretiert, bevor sie es in das finale Release geschafft hat. Am Ende ist nicht nur der Kunde unzufrieden, das Ergebnis wirft auch ein schlechtes Licht auf den Entwicklungsbereich. Es entsteht der Eindruck, als sei dieser nicht sonderlich gut verzahnt mit den Zielen und den Prozessen des Unternehmens.

Das Problem wurzelt oft im Fehlen eines übergreifenden Anforderungsmanagements – eine Disziplin, die jeden einzelnen Schritt des Application-Development-Lifecycle durchdringen sollte. Jedes Projekt, das durch Probleme bei der Sammlung, Analyse und Priorisierung von Anforderungen beeinträchtigt wird und mit verteilten Anforderungen bei unzulänglicher Nachverfolgbarkeit kämpft, ist a priori zum Scheitern verurteilt.

Anforderungen dirigieren den Entwicklungsprozess, ganz gleich welcher Art das angestrebte Resultat ist. Ob es um ein Produkt, ein System oder eine Anwendung geht, die Verankerung der Anforderungen im Entwicklungsprozess stellt sicher, dass die wirtschaftlichen Vorgaben und der Bedarf Ihrer Kunden in priorisierte Anforderungen einfließen, die beständig jede Phase des Entwicklungsprozesses begleiten und in jeder Phase überprüfbar sind.

Die Verankerung der Anforderungen im Projekt

Das V-Modell illustriert, wie Anforderungen durch den gesamten Prozess fließen (Abbildung 1). Der Anforderungsprozess findet nicht nur ganz am Anfang statt, in dem man die Kunden oder andere Quellen nach ihren Wünschen befragt, und dann hingeht und das System baut. Im realen Leben können sich Anforderungen im Verlauf des Projektes ändern, oft sogar unerwartet, und solche Änderungen wirken sich in jeder Phase des Prozesses aus.

Mit der durchgängigen Verankerung der Anforderungen im Projekt generiert jeder Änderungsauftrag ein Flag, das anzeigt, welche Teile des Projektes von dieser konkreten Änderung betroffen sein werden – von der ursprünglichen Anforderung bis hin zum abschließenden Testfall für dieselbe Anforderung. Tatsächlich können die Konsequenzen jeder Änderung sogar zurück bis zum ursächlichen Geschäftsprozess verfolgt werden, aus dem diese Anforderung resultierte. Die Traceability bleibt nicht beschränkt auf die Softwareentwicklung.

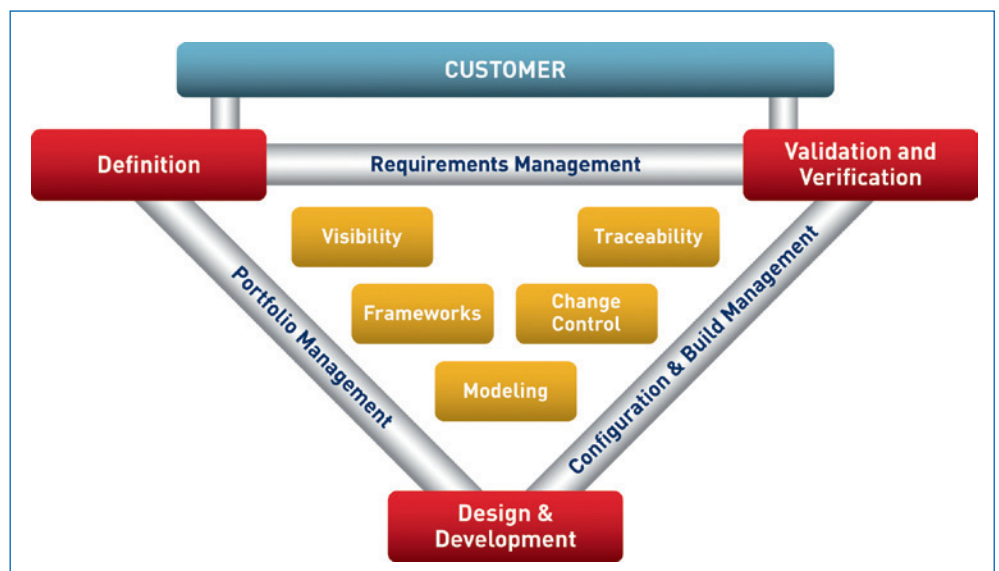


Abb. 1: Die Anforderungen durchdringen jede Phase des Entwicklungsprojektes

Dieser Ansatz zur Verankerung der Anforderungen mit dem ganzen Entwicklungsprojekt leistet:

- Die Erfassung der Wünsche des Kunden
- Die Definition der Kundenanforderungen
- Die Priorisierung der Anforderungen (z. B. Differenzierung in essentielle Anforderungen und solche, die zwar "nice to have" sind, aber nicht von überragender Bedeutung für den Erfolg des Projektes)
- Die Definition und Verfeinerung der Anforderungen bis zu einer Detailebene, die für die Entwicklung handhabbar ist.
- Den Nachweis der Einhaltung beliebiger Regelwerke, wie z. B. behördliche Vorschriften, Unternehmensleitsätze, Prüfbedingungen, Metriken für Prozessverbesserung, etc.
- Die Nutzung der Anforderungen zur Steuerung der Implementierung von Hardware, Software, Elektronik, Embedded-Systemen, etc.
- Die Validierung der Anforderungen durch die Tests
- Das kontrollierte Management von Änderungen

Die Anforderungen werden mit allen Aktivitäten im Application-Development-Lifecycle verknüpft. Gleichzeitig wird die Entwicklung integriert, automatisiert und beschleunigt, weil die verschiedenen in einzelnen Phasen genutzten Prozesse zu einem umfassenden Lifecycle-Prozess verbunden werden. Ein solches Framework liefert Kontrolle und Steuerung über den gesamten Entwicklungsprozess hinweg. Die Erfassung aller Anforderungen in einer zentralen Datenbank optimiert den Austausch von Informationen zwischen allen Beteiligten.

Das gesamte Projektteam kann schneller und präziser auf Änderungswünsche der Kunden reagieren, egal an welcher Stelle im Prozess sie auftreten. Alle Arten von Anforderungen – von Kunden, funktionale Anforderungen, Anforderungen an die Software oder an das (Sub-) System – stehen für die iterative Verfeinerung des Designs und durch alle Phasen der Implementierung und Tests zur Verfügung. Gleichzeitig sorgt die Verknüpfung dafür, dass der ursprüngliche Wunsch des Kunden nie aus dem Blickwinkel gerät. Die Anforderungen durchdringen jeden Abschnitt des Entstehungsprozesses, so dass die möglichen Auswirkungen von Änderungen jederzeit leicht aufgezeigt, und mögliche Alternativen diskutiert werden können. Dabei ist die Impact-Analyse nicht auf den Source Code beschränkt – alle Projekt-Artefakte werden in die Analyse einbezogen, so dass eine

informierte Entscheidung über die Durchführung einer Änderungsanfrage möglich ist.

Anforderungen definieren und managen

Definition und Management von Anforderungen sind entscheidende Voraussetzungen dafür, dass das Projekt den Bedarf der Kunden erfüllt, dem Kontrakt entspricht und im geplanten Zeit- und Kostenrahmen durchgeführt wird. Darüber hinaus ist dies eine Pflichtdisziplin in vielen Standards, Regularien und Prozessverbesserungsinitiativen wie CMMI®. Eine schlecht beschriebene Anforderung kann verheerende Auswirkungen haben. Wie beim Domino-Effekt können zeitaufwendige Nacharbeiten, unzulängliche Lieferungen, Überschreitungen des Budgets und Probleme bei der Konformität die Folge sein.

Die besten Anforderungen sind solche, die technisch möglich, rechtlich zulässig, vollständig beschrieben und präzise formuliert sind. Sie sind konsistent und widersprechen sich nicht. Sie sind in der Lage, ein eindeutiges Kriterium für die Erfüllung dieser Anforderung zu definieren, so dass für jede Anforderung ein Test formuliert werden kann, anhand dessen zweifelsfrei festgestellt werden kann, ob diese Anforderung erfüllt ist oder nicht. Jede Anforderung sollte eindeutig identifiziert und somit über den ganzen Prozess verfolgt werden können. Anforderungen sollten unabhängig vom Design sein; sie sollten die Erwartungshaltung definieren ohne eine konkrete Umsetzung zu präjudizieren.

Verifikation und Validierung

Verifikation und Validierung dienen der Überprüfung und dem Nachweis, dass Entwurf, Entwicklung, Test und Aus-

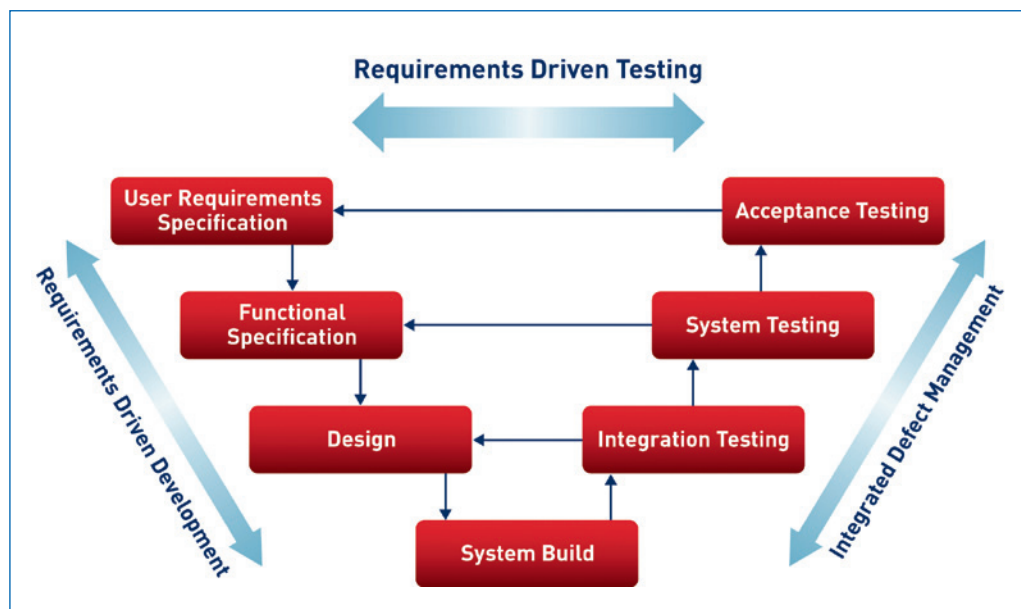


Abb. 2: Verifikation und Validierung

Lieferung des Projektergebnisses in jeder Hinsicht den Anforderungen entsprechen (Abbildung 2). Dafür bietet der hier vorgestellte Ansatz ein Framework für den Einsatz von Werkzeugen, welche die Entwicklungsaktivitäten automatisch zu den Kundenwünschen in Bezug setzen, so dass das Team immer auf Basis der aktuellen Entscheidungen und Prioritäten arbeitet. Das Team ist gefordert mit jedem Build, der getestet wird, eine Verbesserung zu erreichen – die Verankerung der Anforderungen mit dem Entwicklungsprozess ermöglicht dank des Bottom-Up-Ansatzes auch die Validierung des Builds während der Tests. Teamleiter und Tester können bestätigen, dass die für diesen Build vorgesehenen Eigenschaften und Fehlerkorrekturen tatsächlich und im Einklang mit den Prozessvorgaben implementiert worden sind.

Performance und Metriken kontrollieren

Die Aufbereitung von Daten über den Projektfortschritt für das Management ist oft ein Balanceakt zwischen technischer Detailtiefe und übermäßiger Abstraktion. Letztendlich muss das Management in der Lage sein, Fortschritt und Red-Flags auf einen Blick zu erkennen, damit Handlungsbedarf frühzeitig erkannt wird. Die Verankerung der Anforderungen mit dem Entstehungsprozess liefert auch hierfür einen Lösungsvorschlag. Aussagefähige Metriken lassen sich aus den Anforderungen gewinnen und ihren Verknüpfungen mit korrespondierenden Artefakten entlang der verschiedenen Phasen. Die Traceability erlaubt die präzise Messung, wie weit die Implementierung gemessen an der Anzahl der Anforderungen vorangeschritten ist und ob der erreichte Stand mit dem geplanten Status zu diesem Zeitpunkt übereinstimmt. Metriken über Bearbeitung von Anforderungen erlauben Rückschlüsse auf die Qualität der Beschreibung, und Metriken über die Zuordnung von Anforderungen zu Kategorien können Lücken in der Definition offen legen. Solche „Dashboard“-Ansichten sollten für spezifische Projektszenarien und Managementenerwartungen anpassbar sein.

Anforderungen als Maßstab für durchgängiges Qualitätsmanagement

Dieser Ansatz liefert sowohl kurzfristige als auch mittel- und langfristige Vorteile:

- Die Qualität des Projektergebnisses wird verbessert. Es werden insgesamt weniger Fehler gemacht, und Fehler werden früher entdeckt.
- Die Transparenz der Informationen, die Kommunikation und die Zusammenarbeit werden verbessert, auch bei geographisch und funktional verteilten Teams
- Die Produktivität und die Effizienz der Entwicklung werden erhöht.
- Externe und interne Kunden werden zufrieden gestellt.
- Das Entwicklungsprojekt erfüllt auch die wirtschaftlichen Ziele der Unternehmung.

Oft gibt es im Application-Development-Lifecycle Kommunikations- und Informationslücken zwischen Projektplanung, Anforderungsdefinition, Analyse, Design, Implementierung und Verifikation. Nimmt man die Anforderungen als Maßstab für durchgängiges Qualitätsmanagement von der Idee bis zur Lieferung führt dies zwangsläufig zu einer besseren Synchronisation zwischen diesen verschiedenen Disziplinen und zu einem besseren Austausch der relevanten Informationen mit weniger Missverständnissen und damit letztendlich zu einer nachhaltigen Qualitätsverbesserung. ■

Renate Stuecka

Senior Market Manager verantwortlich für Marketing und Kommunikation im Bereich IBM Rational Software. Zudem Erfahrungen im Business Development und Partnermanagement. Mehr als 10 Jahre in verschiedenen Positionen verantwortlich für Marketing, Vertrieb und Entwicklung komplexer Software- und Kommunikationsprodukte.
(renate.stuecka@de.ibm.com)

inserenten

Firma	URL	Seite
Bechtle	www.bechtle.com	40
GFB Softwareentwicklung	www.q-up-data.com	21
IBM Deutschland GmbH	www.ibm.com/software/de/rational/agility	17
IBM Deutschland GmbH	www.ibm.com/software/de/rational	39
OBJEKTSpektrum/JavaSPEKTRUM	www.sigs.de/publications/aboservice.htm	13
Online Themenspecials 2012	www.sigs-datacom.de/os/themenspecial.htm	8
PKS Software GmbH	www.pks.de	2
SVA System Vertrieb Alexander GmbH	www.sva.de	25

Software testen mit „rechtskonformen“ Daten!

Die Rechtsauslegungen der Novellierung zum europäischen Datenschutzrecht sind in den Unternehmen noch nicht ganz angekommen, da tragen sich einige Telekommunikationsanbieter bereits mit dem Gedanken, die Bundesregierung derart zu beeinflussen, dass weiter mit nicht anonymisierten und vollständig personenbezogenen Daten getestet werden dürfe. Die Aufgaben seien derart komplex, dass man sie nur mit erheblichen Geldmitteln stemmen, was wiederum zu Wettbewerbsverzerrung führen könne.

Softwaretesten ist wichtig. Wir haben gerade bei ELENA gesehen, wie viel Geld und Zeit in ein Projekt investiert wird, dessen Anforderungen nicht ausreichend definiert wurden und bei dem die Zeit nicht ausreicht, die Anwendung ausreifend zu testen. Die Softwaretestbranche entwickelt sich derzeit zu einer Industrie, deren Bedeutung immer größer wird. Gut qualitätsgesicherte Software ist eine Voraussetzung für Unternehmen, um im Wettbewerb der Besten bestehen zu können.

Die Anbieter von Softwaretest-Lösungen, gestern noch Nischenanbieter, heute in großen Unternehmen integriert, reifen Ihre Produkte mit immer mehr möglicher Automatisierung aus. Softwaretester, die sich in den verschiedenen Zertifizierungshierarchien prüfen lassen und ständig neuen Anforderungen gegenüberstehen, sind heute gesuchte Spezialisten und Allrounder.

Damit die Kommunikation zwischen Entwicklung und Test funktioniert, setzen diese Teams auf Lösungen namhafter Hersteller und Dienstleister. Mit Rational & InfoSphere Optim Test Data Management Solution hat auch IBM verschiedene Werkzeuge im Portfolio und hat zuletzt dafür gesorgt, dass mit der Einbindung der Plattform JAZZ, ein kollaboriertes Arbeiten möglich ist.

Ein Baustein, der vielen Unternehmen immer noch Kopfschmerzen bereitet und auf den auch große Softwarehersteller bisher nur Teilantworten liefern, ist das Arbeiten mit synthetischen Daten, dem Verwalten von großen Testdatenbeständen und validen Anonymisierungsfunktionalitäten aus den Produktionsdaten heraus. IBM hat hier mit einem Partnerunternehmen eine Lösung gefunden, mit der nahezu alle Anforderungen an ein Testdatenmanagement erfüllt werden können.

Q-up rundet hier das Portfolio in Richtung ETL, Datenmodellierung und Aufspüren von Daten mit InfoSphere/Optim ab und ermöglicht die Verbindung von anonymisier-



Der Testdatengenerator mit IQ

Compliance & Datenschutz

- „live“ Anonymisierung von Produktionsdaten
- Sicherheit für externe Testteams & Auftraggeber
- Sichere Datenmigration in neue Systeme
- Integration in alle gängigen Testsuiten

Sie wünschen eine ausführliche Beratung oder Präsentation? Nutzen Sie unsere kostenlose Service-Hotline*:

0800 7873282

*aus dem deutschen Festnetz, Mo.-Fr. 10-13 Uhr und 14-17 Uhr

www.q-up-data.com



Q-UP FUNKTIONEN

- ✓ Erzeugen synthetischer Daten, Files und Datenpools
- ✓ Teil und „Live“ Anonymisierung von Produktionsdaten für Softwaretests
- ✓ Simulieren der fachlichen Logik in Testdaten
- ✓ Dynamisierung von Testdaten und Simulation historisch gewachsener Daten
- ✓ Berücksichtigung der referentiellen Integrität
- ✓ Kostenreduktion bei Maintenance und Testdatenerstellung
- ✓ Einfache Integration in alle gängigen Testsuiten

Ein Produkt der



Obere Zeil 2
61440 Oberursel
Tel.: +49 (0) 6171/69410-0

DER TESTDATENGENERATOR

ten Produktionsdaten hin zu vollsynthetischen, automatisierten Testdaten für Softwaretests, auch in hochkomplexen Umfeldern. Mit der Integration von Q-up, dem Testdatenmanagementwerkzeug, der GFB-Softwareentwicklung aus Oberursel bei Frankfurt, haben Softwaretester die Möglichkeit, kollaboriert Testdatenbestände zu erzeugen und auf Knopfdruck zur Verfügung zu stellen. Hier wird ein Produkt direkt aus einer Anforderung aus dem Softwaretest bereitgestellt, das den Bedürfnissen in der täglichen Arbeit in Preis und Leistung gerecht wird.

Hochkomplexe Testdaten können aus dem Rational Quality Manager oder anderen Testsuiten beschrieben und direkt in den Test geliefert werden, ob Einfach- oder Massendaten. Mit dieser Lösung hat es die GFB geschafft, dass Unternehmen sich mit der Thematik auseinandersetzen können und den Weg der Anonymisierung von Produktivdaten so erreichen, dass die Daten lebend verwaltet werden. Es müssen nicht mehr ständig neue Produktionsdatenabzüge vorgehalten werden, die veralten und verbraucht werden. Neben Zeitreisen und der wartungskostenfreundlichen Verwaltung der Testdaten, haben Unternehmen hiermit die Möglichkeit, Testdatenprojekte für Softwaretests IT-compliant und datenschutzrechtlich konsequent richtig umzusetzen. Mit der Einbindung in die Jazz Plattform erhalten Testteams mannigfaltige Möglichkeiten der Automatisierung und sind als Mitarbeiter datenschutzrechtlich abgesichert.

Mit der AutoLoader-Funktion hat die GFB bei namhaften Kunden derzeit Testinstallationen im Betrieb, die zeigen, dass valide Anonymisierungsprojekte für Testdaten möglich sind und den Weg ebnen, um in der Zukunft vollständig auf Produktionsdaten verzichten zu können. Das Umsetzen der fachlichen Inhalte in Produktionsdaten ist mit Q-up möglich geworden.

Gesetzliche Anforderung Wahrnehmung und Wirklichkeit

Was muss datenschutzrechtlich beachtet werden, wenn mit Daten von Kunden, Auftraggebern, Lieferanten, Partnern und Personal Software getestet wird? Für die meisten Softwaretests werden keine personenbezogenen Daten benötigt, aber eingesetzt! Wie verhält sich die Software mit diesem oder Millionen von Datensätzen? Ein Datensatz entwickelt und verändert sich über seinen Lebenszyklus derart, dass Testmanager davon ausgehen, dass Softwaretests mit produktiven Daten aus verschiedenen Zyklen relevantere Ergebnisse liefern als solche, die von Hand oder so genannten Testdatengeneratoren erzeugt werden.

Dass Abzüge von Produktionsdaten in der Realität verfremdet werden müssen, also die personenbezogene Herkunft eines Datensatzes keine Rückschlüsse auf den Eigentümer ableiten lassen darf, ist bekannt. Nicht bekannt ist, dass der Eigentümer seine Zustimmung zur Verwendung, der Verfremdung, einer Ausfuhr oder Übergabe an Dritte geben muss. In einem Grundsatzpapier der EU steht: „Die derzeit geltende Datenschutzregelung der EU zielt darauf ab, die Achtung der Grundrechte natürlicher Personen, insbesondere des Grundrechts auf den Schutz personenbezogener Daten zu garantieren, wie es die EU-Charta der Grundrechte vorsieht.“

Noch gibt es keine Bestimmungen, wie mit Daten in der Qualitätssicherung umgegangen werden muss. Liest man das Bundesdatenschutzgesetz mit dem Hintergrund Softwaretest, so ist abzuleiten, dass die bisher getroffenen Maßnahmen und derzeitigen Standards nicht mehr ausreichen, um sich vor Missbrauch, Diebstahl und daraus entstehenden zivilrechtlichen Folgen zu schützen. Unternehmen, die hier vorbeugen wollen, müssen jetzt reagieren, um für die Zukunft gerüstet zu sein. Kein Unternehmen kann es sich leisten, in den Fokus datenschutzrechtlicher Ermittlungen zu kommen.

Besonders das Aufspüren und der Nachweis, welche Daten wo und von wem verwendet wurden und wann diese zurückgeführt wurden, ist ein mühsamer und langer Prozess, der Ressourcen unnötig bindet. Einmal erzeugte Produktionsdatenabzüge dürften ohne die Einwilligung der betroffenen Personen nicht gelöscht werden, zieht man das Bundesdatenschutzgesetz zu Rate, hier geregelt in §11. Es gilt der Merksatz: „Es sind genaue Prozesse zu definieren und einzuhalten, wenn Daten übergeben und genutzt werden sollen“.

Quick Check – Fragen an Ihr Unternehmen

1. Nutzt Ihr Unternehmen Produktionsdaten im Softwaretest?
2. Werden diese anonymisiert, also verfremdet, bevor sie für Testzwecke eingesetzt werden?
3. Ist sichergestellt, dass Testmanager keine aussagefähigen Produktionsdaten erhalten?
4. Hat Ihr Unternehmen Testdaten schon einmal in ein Land außerhalb der EU gesendet?
5. Hat Ihr Unternehmen eine Einverständniserklärung des Dateneigentümers zur Verwendung personenbezogener Daten im Softwaretest?
6. Hat Ihr Unternehmen einen Prozess definiert, wie mit Testdaten zu verfahren ist, wenn der Testprozess beendet wurde?
7. Werden die Testmanager in Ihrem Unternehmen oder die beauftragten Unternehmen datenschutzrechtlich aufgeklärt? ■

Stephan Oswald

ist zuständig für Vertrieb & Marketing von Q-up bei der GFB Softwareentwicklungsgesellschaft mbH in Oberursel.
(Stephan.Oswald@GFB-Softwareentwicklung.de)

Source-Code-Richtlinien leichter einhalten

Seit geraumer Zeit gibt es Studien zur Korrelation zwischen der Qualität beziehungsweise Komplexität des Source Codes und der daraus resultierenden statistischen Fehlerhäufigkeit je Source-Code-Abschnitt. Da Wartungs- und Pflegekosten von Softwareprojekten seit Jahren eine immer wichtigere Rolle spielen, nehmen die Einhaltung und Sicherstellung der Source-Code-Qualität basierend auf zugehörigen Richtlinien eine zentrale Rolle im Softwareentwicklungsprozess ein. Dies auch vor dem Hintergrund, Qualitätskriterien als wichtigen Bestandteil von Abnahmekriterien gegenüber Zulieferfirmen zu definieren. Hinzu kommt die – je nach Industriezweig – teilweise zwingend notwendige Einhaltung zugehöriger Standards. Dieser Artikel beschreibt die Wichtigkeit von Standards zur Source-Code-Entwicklung, zeigt mögliche Vorgehensweisen zur Etablierung oder auch Einhaltung der Standards auf und nennt einige Tools zur automatisierten Analyse des Source Codes sowie zur Überprüfung der Einhaltung von Richtlinien.

Notwendigkeit von Richtlinien und Standards

In der heutigen Softwareentwicklung spielen Richtlinien und Standards eine sehr wichtige Rolle. Sowohl für das Design als auch für die Implementierung existieren bereits eine Fülle an zugehörigen Richtlinien und Standards. Zu Beginn eines Softwareprojektes passen die beteiligten Entwicklerteams meist existierende Richtlinien an die eigenen Bedürfnisse ihrer Entwicklungsorganisation oder aber an projektspezifische Gegebenheiten an („tailoring“). In einigen Fällen übernehmen Entwicklungsorganisationen sogar bereits existierende Standards/ Richtlinien vollständig als Basis für die geplante Softwareentwicklung. Existierende Frameworks für Quellcode-Richtlinien – wie beispielsweise die Findbugs-Bibliothek zur Identifikation von Fehlermustern in Java-Quellcode – stellen schon eine sehr gute Basis solcher Richtlinien dar.

Ziele von Quellcode- bzw. Codier-Richtlinien sind:

1. Die Lesbarkeit des Source Codes zu optimieren:

Die Lesbarkeit des Quellcodes ist heutzutage eine implizite Anforderung der Softwareentwicklung. Für die unterschiedlichsten Programmiersprachen existieren Guidelines beispielsweise zur Namensgebung von Variablen, Methoden, Konstantenbezeichnern usw. Bezeichner sollen im Allgemeinen lesbar, d. h. zu einem gewissen Grad bereits selbsterklärend sein. Da diese Regeln heutzutage eine de-facto-Anforderung in der Softwareentwicklung darstellen und bereits weitläufig bekannt sein sollten, wird hier nicht näher darauf eingegangen. Nichtsdestotrotz gibt es durchaus immer wieder Fälle, in denen ein Entwickler diese grundlegende Anforderung nicht einhält. Sinn und Zweck einer solchen Standardisierung im Rahmen der Quellcode-Lesbarkeit ist zum einen die Anforderung, dass unterschiedliche Entwickler möglichst ein-

fach an fremdem Source Code arbeiten können. „Einfach“ bedeutet hier, ohne größeren Einarbeitungsaufwand aufgrund beispielsweise kryptischer Codierweise. Ein weiterer positiver Aspekt einer guten Lesbarkeit von Source Code ist, dass er generell besser verständlich ist und sich somit Fehler leichter finden lassen. Es existiert eine Fülle an Produkten zur Analyse und Bewertung von Source Code auf der Basis von Style-Guidelines. (siehe auch Referenzen)

2. Fehler möglichst frühzeitig im Softwareentwicklungszyklus erkennen und vermeiden:

Heutige Softwareentwicklungsprozesse haben sich bereits in vielen Bereichen dem Aspekt Qualität angenommen. Sowohl auf Design- als auch auf Source-Code-Ebene existieren standardisierte Pattern als auch Anti-Pattern. Für Webanwendungen gibt es eine Reihe von Security-Richtlinien. Diese können beispielsweise mit Tools wie Rational AppScan verifiziert werden. Darüber hinaus werden Softwarequalitätsmetriken zunehmend beliebter. Solche Metriken stellen eine Korrelation zwischen dem Vorhandensein einer Metrik oder einer bestimmten Kombination mehrerer spezifischer Metriken dar, und leiten daraus eine Aussage über die Qualität der untersuchten Softwarekomponente ab. Die Interpretation solcher Qualitätsbewertungen sollte jedoch nicht alleine auf den reinen Werten der ermittelten Metriken erfolgen, sondern zusätzliche Aspekte wie Design, Style-Guidelines o. ä. bewerten. Beispiele für Metriken sind: NCSS (non commented source code statements), DOIT (depth of inheritance tree), CCN (cyclomatic complexity number). Eine Erläuterung der einzelnen Metriken erfolgt hier nicht. In den Referenzen befindet sich unter anderem ein Link zu weiterführenden Informationen zum Thema. Zur Ermittlung von Softwarequalitätsmetriken existieren bereits eine Fülle am Markt erhältlicher Produkte, teils kommerziell, teils Open Source.

Für die Auswahl beziehungsweise Festlegung der für ein Projekt gültigen Source-Code-Richtlinien ist zumeist der leitende IT-Architekt oder IT-Spezialist zuständig. In kleineren Projekten kann diese Verantwortung auch der Projektleiter übernehmen, wenn er über ausreichendes technisches Verständnis verfügt. Die Entscheidung erfolgt meist unter Einbeziehung aller beteiligten Entwickler. Für die Akzeptanz der Richtlinien unter allen Entwicklern ist es notwendig, zum einen ein generelles Verständnis über den Sinn und die Notwendigkeit dieser Richtlinien zu schaffen. Zum anderen sollte eine bestmögliche Integration der für die Einhaltung der Richtlinien meist notwendigen Tools in den Entwicklungsprozess beziehungsweise die Entwicklungsinfrastruktur des Projektteams sichergestellt werden. Dies sollte bereits vor Projektstart etabliert werden, um unnötige Verzögerungen bei der Einführung während eines laufenden Projektes zu vermeiden.

Die Sicherstellung der Einhaltung der ausgewählten Source-Code-Richtlinien kann in verschiedenen Abschnitten des Entwicklungszyklus erfolgen. Die generelle Regel lautet: Je konsequenter und konsistenter die Richtlinien in jeder der aufgeführten Bereiche verifiziert und sichergestellt werden, desto besser ist das Gesamtergebnis beziehungsweise desto geringer ist der reale Aufwand zur Sicherstellung der Einhaltung der Richtlinien.

Tool-Integration zur Validierung für den Entwickler

Zur Einhaltung von Source-Code-Richtlinien ist es aus Sicht des Entwicklers wünschenswert, die entsprechenden Tools möglichst direkt in der eigenen Entwicklungsumgebung (IDE) integriert zu haben. Die am Markt erhältlichen Produkte zur Validierung von Source-Code-Richtlinien erfüllen bereits weitgehend diese Anforderung und stellen entsprechende Plugins für gängige Entwicklungsumgebungen – wie beispielsweise Eclipse oder Microsoft Visual Studio – bereit. Vorteil einer direkten Integration eines solchen Tools ist, dass die Entwickler bereits während der Erstellung des Source Codes beziehungsweise während der Ausführung von Unit-Tests die Einhaltung von Source-Code-Richtlinien verifizieren und gegebenenfalls korrigieren können. Je früher dies geschieht, desto schneller stellen sich die Entwickler auf die Richtlinien ein. Darüber hinaus ist die Akzeptanz der Richtlinien in der Regel höher, wenn ein Entwickler diese reibungslos in seinem Entwicklungsalltag verifizieren und die Einhaltung sicherstellen kann. Abbildung 1 zeigt exemplarisch die Integration des für die Java-Sprache frei erhältlichen Tools Findbugs. Dieses Tool überprüft Source Code auf gängige Pattern und Anti-Pattern und zeigt dem Entwickler die betroffenen Source-Code-Stellen direkt an.

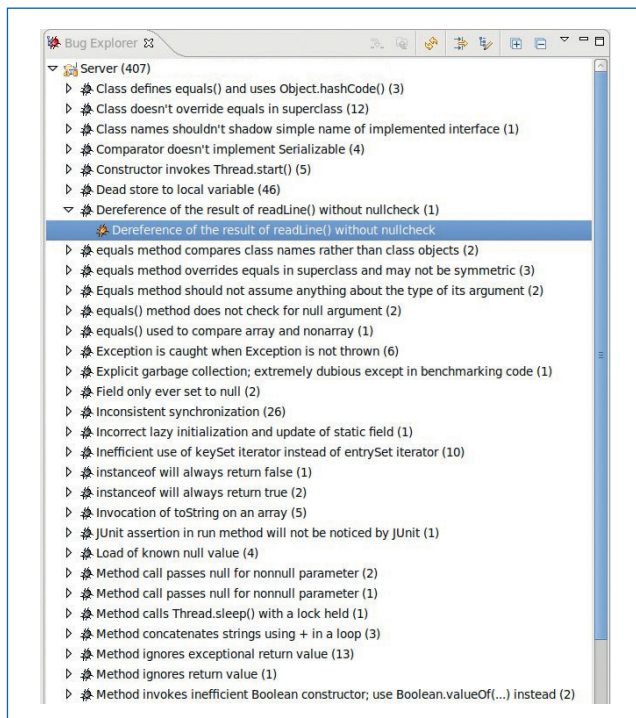


Abb. 1: Exemplarische Tool-Integration in der Eclipse IDE

Sollte die Integration eines Tools in die IDE nicht möglich sein – beispielsweise weil keine zugehörigen Plugins verfügbar sind – ist eine Verifikation der Richtlinien bereits während der Entwicklung trotzdem zu empfehlen. Bei nicht integrierten Tools erfolgt dies meist in mehreren sequentiellen Schritten:

1. Das betreffende Source-Code-Artefakt wird erstellt oder ein existierendes entsprechend überarbeitet/ modifiziert.
2. Der Code wird übersetzt.

3. Die zugehörigen Unit-Tests werden gestartet.

4. Nach erfolgreichem Abschluss der Unit-Tests muss der Entwickler in diesem Fall die IDE verlassen und zum separaten Tool wechseln, das die Einhaltung der Source-Code-Richtlinien validiert. Sofern Verstöße gegen die Richtlinien identifiziert wurden, hat der Entwickler diese entsprechend zu korrigieren und – je nach Umfang der Änderungen – die Schritte 1 bis 4 zu wiederholen, bis sämtliche Source-Code-Richtlinien eingehalten wurden.

Kontrollmechanismus im Rahmen von Inspektionen

Zusätzlich zur direkten Verifikation der Source-Code-Richtlinien kann es sinnvoll sein, weitere Qualitätssicherungsmaßnahmen zu etablieren. Eine sehr effiziente und weit verbreitete Möglichkeit hierfür sind Source-Code-Inspektionen. Inspektionen sind insbesondere dann sinnvoll, wenn ein anderer Entwicklerbereich oder ein anderes Unternehmen den Source Code entwickelt hat. Die Einhaltung von Source-Code-Richtlinien kann hier beispielsweise Teil der Abnahme- und Akzeptanzkriterien werden. Unabhängig davon, welches Team den Source Code entwickelt hat, sollten Inspektionen in einem regelmäßigen Zeitraum angesetzt werden. Je nach Gesamtentwicklungsdauer und Projektorganisation empfiehlt es sich, sie in regelmäßigen Abständen durchzuführen, beispielsweise alle zwei Wochen, oder aber jeweils bereits fertige Komponenten zu inspizieren.

Die Validierung erfolgt in der Regel nach dem im Abschnitt „Tool-Integration zur Validierung für den Entwickler“ dargestellten Verfahren. Je nach Anforderungen an den Inspektionsprozess validieren ein oder mehrere Teilnehmer zunächst die Einhaltung der Richtlinien und präsentieren dann ihre Ergebnisse in einem Inspektion-Meeting. Gibt es Nachbesserungsbedarf, ist dies Aufgabe des Autors der entsprechenden Softwarekomponente. Wurde der Source Code zugeliefert, dann ist der Autor des Lieferanten dafür zuständig.

Integration in den (automatisierten) Build-Prozess

Die beiden vorgestellten Mechanismen sind zwar bereits effizient, können jedoch keine vollständige Einhaltung der Richtlinien sicherstellen. Die erste Methode obliegt mehr oder weniger dem Entwickler und es ist schwierig sicherzustellen, dass alle Entwickler sich an diese Vorgehensweise halten. Die zweite Methode – Inspektionen – kann lediglich stichprobenartige Analyseergebnisse liefern und ist somit ebenfalls nicht zur vollständigen Sicherstellung der Einhaltung von Source-Code-Richtlinien geeignet.

Eine bessere Abdeckung erreicht man durch eine automatisierte Überprüfung im Rahmen des Build-Prozesses. Je nach Anforderung bietet es sich an, vor oder nach dem Übersetzen der Komponenten einen automatischen und vollständigen Validierungslauf zu implementieren. Um dies zu erreichen, muss der leitende IT-Architekt zunächst wieder ein geeignetes Tool definieren. Nach erfolgter Definition des Tools und Konfiguration der gewünschten Source-Code-Richtlinien kann der für den Build verantwortliche Entwickler den

automatischen Start dieses Tools in den projektspezifischen Build-Prozess integrieren. Richtlinien-Verstöße können anhand eines Analyseberichts dargestellt werden, der nach jedem erfolgten Analyselauf ausgewertet wird. Bei Verwendung eines Defect-Management-Systems können basierend auf dem Report direkt Richtlinien-Defects geöffnet und dem entsprechenden Entwickler zugewiesen werden. Diese automatisierte, vollständige Verifikation stellt sicher, dass keine Verstöße gegen Source-Code-Richtlinien in das finale Release und damit an Kunden gehen.

Die dargestellten Methoden zur Sicherstellung der Einhaltung von Source-Code-Richtlinien haben sich bereits in mehreren Projekten bewährt. Sie stellen ein effektives Mittel zur Qualitätssicherung in der Softwareentwicklung dar und können insbesondere auch für die Definition und Validierung von Abnahme- und Akzeptanzkriterien bei der Erstellung von Software durch Zulieferer angewendet werden. ■

[Findbugs] Findbugs Home Page
<http://findbugs.sourceforge.net/>

[Rational] AppScan und Software Analyzer
<http://www-01.ibm.com/software/rational/products/appscan/source/>
<http://www-01.ibm.com/software/awdtools/swanalyzer/enterprise/index.html>

[SonarSource] Sonar Source Home Page
<http://www.sonarsource.org/>

Softwareprüfstelle der Physikalisch-Technischen Bundesanstalt
http://ib.ptb.de/portal/page/portal/IB_PTB/8/85/851/sps/swq/main.html

Ein Einstieg in Softwarequalitätsmetriken
http://en.wikipedia.org/wiki/Software_metric

Software Inspektionen
http://en.wikipedia.org/wiki/Software_inspection

Literatur und Links

[Pressman01] Roger S. Pressman, „Software Engineering – A Practitioner’s Approach“, 5th Edition, 2001,
<http://catalogs.mhhe.com/mhhe/home.do>

[Ligge09] Peter Liggesmeyer, „Software-Qualität: Testen, Analysieren und Verifizieren von Software“, 2. Auflage

[Jones2000] Jones, C., „Software Assessments, Benchmarks, and Best Practices“, Boston: Addison-Wesley, 2000

Oliver Röhrsheim

Zertifizierter Senior IT Architect IBM Softwaregroup, PMP®.
 Aufgabenschwerpunkte in den Bereichen IT-Architektur, Softwareentwicklungsprozesse, Agiles Project Management.
 (o.roehrsheim@de.ibm.com)



TECHNOLOGIE-INTEGRATION FÜR DATACENTER

Lifecycle Management für Systeme mit Embedded Software

- Anforderungsbezogenes Qualitätsmanagement
- Automatisierte & reversionssichere Validierung
- Blackbox Test für einzelne oder vernetzte Systeme
- IBM Premier Partner mit Demo Rechenzentrum: Individuelle Test-Szenarien mit Rational Software

KONTAKTIEREN SIE UNS – DEUTSCHLANDWEIT IN 11 LOKATIONEN!

SVA System Vertrieb Alexander GmbH | Borsigstraße 14 | 65205 Wiesbaden | Telefon 06122-536-0
mail@sva.de | www.sva.de



Testlabor – effizient verwaltet

Heute spricht die ganze Welt von Automatisierung und Virtualisierung, doch was bedeuten diese Buzzwords für die Qualitätssicherung in der Softwareentwicklung?

Es bedeutet, dass die Testausführung vollkommen automatisiert und distribuiert im Testlabor durch Virtualisierung sämtlicher Konfigurationen abläuft. Aber wie integriert man nahtlos und mit geringem Aufwand Qualitätsmanagement mit Testlabor?

Qualitätsmanagement ist heute nicht mehr nur ein Steckenpferd oder gar ein Stiefkind im Softwareentwicklungsprozess, sondern ein fester Bestandteil davon. Es befasst sich nicht nur mit Testfällen und deren Verwaltung, der Traceability zu den Anforderungen und zum Change- und Defect-Management mit Versionierung etc., sondern auch mit der Planung von Ressourcen, insbesondere des Testlabors. Entsprechend hoch sind somit auch die Anforderungen in dieser Disziplin.

Durch komplexe Anforderungen und zunehmendes Qualitätsbewusstsein ist es notwendig geworden, die Testausführung parallel durchzuführen. Der Begriff „Testlabor“ kann hier weiter gefasst werden und z. B. auch Rechnerressourcen beinhalten, die nicht in einem dafür vorgesehenen Raum stehen, wie bei den Rechnern in den Fachabteilungen, die sich nachts „langweilen“. Optimal sind natürlich speziell dafür vorgesehene Rechnerkapazitäten. Diese stehen dann unabhängig von Tageszeit und Projektstress zur Verfügung.

Die Ausführung komplexer, auf mehrere Laborressourcen verteilter Aufgaben sowie deren Überwachung, Steuerung und korrekte Dokumentation, stellt hier natürlich eine besondere Herausforderung dar. Noch schwieriger wird es, wenn man eine Testlab-Manager-Lösung selbst entwickeln und implementieren will. Diesen Aufwand scheuen viele, weil sie der Meinung sind, dass er in keinem Verhältnis zum Nutzen steht, es also kein Return-On-Investment gibt. Die Verwendung einer vordefinierten Tool-Landschaft reduziert diese Arbeit jedoch um ein Vielfaches und kann sie schnell rentabel machen.

Versuch mit einer Out-of-the-Box-Lösung

Im Zuge eines Projektes wurde ich vor die Aufgabe gestellt, so ein Szenario zu implementieren. Die gewählte Out-of-the-Box-Lösung ermöglicht einen hohen Automatisierungsgrad, indem es per Knopfdruck eine komplette Testumgebung erstellt und direkt nach der Initialisierung der Umgebung die Testausführung startet. Wenn die Tests durchgeführt worden sind, startet das System automatisch mit der nächsten Konfiguration - wenn gewünscht auch über Nacht und automatisch.

Doch wie schnell lässt sich so eine Landschaft erstellen? Vorab muss ich gestehen, ich habe mich für eine Standardlösung entschieden, weil hier der Fokus nicht auf der Erstellungsarbeit einer solchen Lösung liegt, sondern auf dem schnellen Produktivwerden.

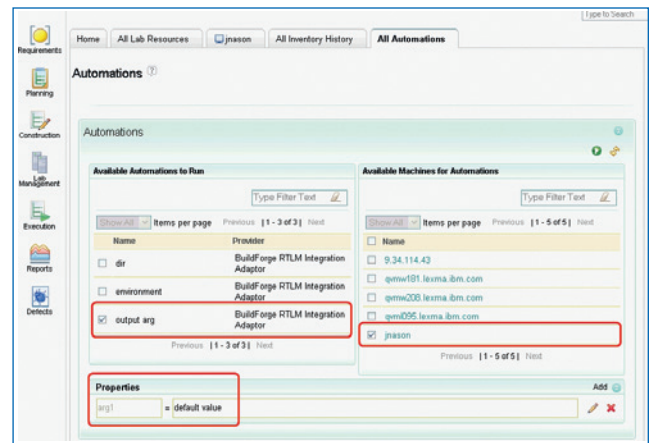
Nach der Installation werden die Werkzeuge miteinander integriert. Dies erfolgt durch die Eingabe von User, Passwort und Rechnerdaten. Danach startet man einen Adapter, der die Kommunikation zwischen Testmanagement-Tool und Deployment sicherstellt. Nach diesen wenigen Schritten ist man in der Lage auf vordefinierte Betriebssystemkonfigurationen zuzugreifen und diese auf die Rechner des Testlabors zu verteilen. Eine weitere Anbindung an ein Testautomatisierungswerkzeug macht es nun möglich, direkt auf den gerade aufgesetzten Konfigurationen automatisierte Tests durchzuführen.

Fazit

Fazit für mich war die Erkenntnis, dass das Aufsetzen und die Konfiguration einer solchen komplexen Lösung sehr einfach waren. Es umfasst Traceability, Dokumentation und Wiederverwertbarkeit – und das out-of-the-box.

Die von mir gewählte Lösung kommt von IBM Rational und stellt eine durchgängige Lösung dar: Die Grundlage bildet der IBM Rational Quality Manager. Dieses Werkzeug basiert auf Jazz, der Technologie-Plattform von IBM Rational. Mit dieser Lösung ist die Qualitätsabteilung optimal aufgestellt, um sich professionell mit der Entwicklung zu integrieren und um die hohen Bedürfnisse einer modernen Qualitätssicherung abzudecken.

Folgende Komponenten kamen in dem geschilderten Szenarium zum Einsatz: Rational Quality Manager, Rational Build Forge und Rational Functional Tester. Damit deckt man z. B. die Testautomatisierung von verschiedenen Konfigurationen einer GUI-Anwendung ab. Durch die Offenheit der Jazz-Plattform sind hier natürlich unzählige andere Werkzeug-Konstellationen denkbar.



In dieser Abbildung ist zu erkennen, wie eine Automation auf einem Testlaborrechner gestartet und mit Parametern versehen werden kann. Mit diesem PowerPack ist es einfach, mehr Qualität und Effektivität zu erreichen, sowie Kosten und Zeit einzusparen. Die Frage nach dem Aufwand-Nutzen-Verhältnis erübrigt sich damit.

Edgar Boehm

IBM Software Group Rational
Senior Technical Sales Professional
Channel Technical Focal Point Rational
(edgar.boehm@de.ibm.com)

Durchgängige Web-Application-Security

Webanwendungen werden immer mehr zum Aushängeschild und zur zentralen Geschäftsplattform für Unternehmen. Doch bestehende Sicherheitsmaßnahmen auf Netzwerkebene schützen nur unzureichend gegen gezielte Angriffe auf Schwachstellen in Webanwendungen selbst. Wöchentlich berichten die Medien über erfolgreiche Hackerattacken. Aber so einfach Sicherheitslücken ausgenutzt werden können, so einfach ist es für die betroffenen Unternehmen in der Regel auch, diese Lücken rechtzeitig zu erkennen und zu schließen.

Schwachstellen in Webanwendungen und deren Auswirkungen

Moderne Webanwendungen sind nicht mehr mit den ersten Webseiten der frühen Internetzeit vergleichbar. Wurden früher häufig nur statische Inhalte mit wenigen Interaktionsmöglichkeiten präsentiert, so sind die aktuellen Webanwendungen oft sehr interaktiv und durch diverse Technologien angereichert. So findet man z. B. Java, Javascript und AJAX oder Flash-basierte Inhalte. Diese Technologien bieten den Vorteil, sehr dynamische Inhalte interessant darstellen zu können. Über das Web-Frontend können Benutzern und Kunden Informationen aus unterschiedlichen Quellen und Bereichen zugänglich gemacht werden.

Bedarf und Interesse an Webanwendungen nehmen stetig zu. Es gibt inzwischen eine Vielzahl unterschiedlicher Plattformen und Frameworks für deren Erstellung. Auch die Anforderungen der Unternehmen an Inhalt und Erstellungstempo der Webauftritte steigen. Das erhöht den Druck auf die Entwickler, mehr Funktionalität und Performance in immer kürzeren Release-Zyklen unterzubringen.

Die Forderung nach immer mehr Funktionalität in immer kürzerer Zeit führt zu Vernachlässigung des Sicherheitsaspekts. Zudem sind klassische Entwickler keine Security-Experten. Die Webtechnologien werden immer komplexer, Unternehmen können sie immer schwerer hinsichtlich ihrer Sicherheit testen. Und je komplexer die Webanwendungen werden, desto mehr Sicherheitslücken enthalten sie. Das macht sie zu einem sehr attraktiven Ziel für Hackerangriffe.

Auch schwerwiegende Sicherheitslücken können recht einfach gefunden und ausgenutzt werden. Ende 2010 war es einem 16-jährigen Schüler gelungen, in 17 Online-Auftritten von Banken Sicherheitslücken zu finden. Der Schüler war in der Lage die Schwachstelle „Cross-Site-Scripting“ (XSS) in den Seiten aufzudecken. Bei dieser Art Angriff schleust man manipulierten Script-Code auf Seiten ein und kann so zum Beispiel an sensitive Nutzerdaten gelangen. Auch die häufig ausgenutzten Phishing-Attacken basieren auf dem Ansatz des Cross-Site-Scriptings.

Wenn Sicherheitslücken an die Öffentlichkeit gelangen, dann bringt das neben dem Datenverlust schlechte Presse und Imageschäden mit sich. Hinzu kommen häufig auch

rechtliche und regulatorische Konsequenzen für die Unternehmen. Denn wenn Hacker auf sensitive Kundendaten zugreifen können, liegt in der Regel eine Compliance-Verletzung vor, wie beispielsweise die Nicht-Einhaltung des Payment Card Industry Data Security Standards (PCI DSS) im Zusammenhang mit Kreditkartendaten. Diese Compliance-Verstöße können für Unternehmen schnell Strafen in Höhe von mehreren hunderttausend Euro nach sich ziehen.

Aktuelle Untersuchungen des IBM X-Force Teams, das sich auf das Auffinden und Dokumentieren von Sicherheitslücken spezialisiert hat, belegen, dass etwa die Hälfte der derzeitigen Schwachstellen in Webanwendungen stecken. Diese Anfälligkeit macht Webauftritte für Kriminelle besonders interessant, da es für sie meist auch sehr leicht ist, diese Schwachstellen auszunutzen.

Sicherheitslücken identifizieren

Grundsätzlich gibt es zwei Vorgehensweisen, um Sicherheitslücken in Webanwendungen zu identifizieren: die statische Analyse des Quellcodes (White-Box-Tests) und die dynamische Analyse der Anwendung selbst in dem Zustand, wie diese auch letztendlich genutzt wird (Black-Box-Tests). Manche Sicherheitslücken lassen sich durch beide Vorgehensweise finden, andere nur mit der jeweiligen Vorgehensweise.

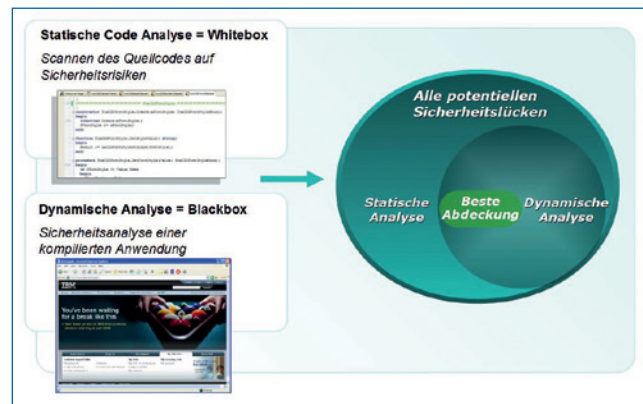


Abb. 1: Sicherheitslücken mit statischer oder dynamischer Analyse aufdecken

Black-Box-Tests (Web)

Black-Box-Test bezeichnet eine Methode, bei der die Tests ohne Kenntnisse über die innere Funktionsweise und der verwendeten Technologie des zu testenden Systems entwickelt werden. Die Tests werden lediglich auf Basis der Interfaces durchgeführt, die auch einem Nutzer der Anwendung zur Verfügung stehen. Im Kontext von Webanwendungen ist dies das HTTP-/HTTPS-Interface.

Das Aufspüren von Sicherheitslücken mittels Black-Box-Tests lässt sich am besten als cleveres Erraten über manipulierte HTTP-Anfragen beschreiben. Dieses Erraten bzw. Herantasten an eine Schwachstelle macht unter anderem auch den Reiz für Hacker aus, die auf ähnliche Art vorgehen. Der große Vorteil einer solchen Untersuchung liegt darin, dass man nahezu unabhängig von der verwendeten Technologie jede auf HTTP(S) basierende Anwendung mit

den gleichen Tests prüfen kann. Die Tests beziehen beim Black-Box-Test üblicherweise das gesamte System ein, d. h. Server (Applikationsserver, Webserver, DB Server, etc.), externe Schnittstellen, Netzwerk, Firewalls oder Drittanbieterkomponenten.

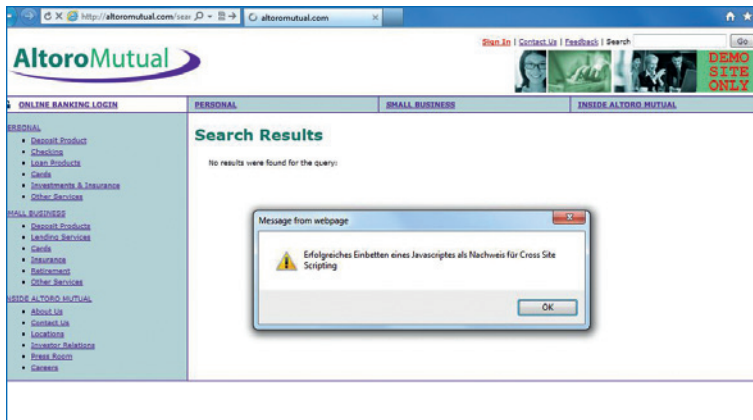


Abb. 2.: Black-Box-Test-Nachweis für Cross-Site-Scripting

erlaubt jedes mögliche Verhalten der Anwendung in jedem Winkel des Quellcodes nachzuvollziehen.

Manuell oder automatisiert

Sowohl Black-Box-Tests als auch White-Box-Tests lassen sich auf manuelle und automatisierte Art und Weise durchführen. Die Automatisierung ersetzt dabei selbstverständlich nicht den menschlichen Verstand. Der erlaubt es schließlich, Sicherheitslücken in der Logik von Abläufen innerhalb von Webanwendungen zu erkennen, wozu Algorithmen in Form von Programmen nur eingeschränkt fähig sind. Andererseits können durch automatisierte Tools sehr viele Tests sehr schnell und mit maschineller Sorgfalt durchgeführt werden. Für einen Mensch würde das tage- beziehungsweise wochenlange Routinarbeit bedeuten. Und das automatische Generieren von Reports im toolbasierten Ansatz nimmt dem manuellen Tester die wohl mühevollste Arbeit ab.

White-Box-Tests (Web)

White-Box-Test bezeichnet eine Methode, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems durchgeführt werden. Im Gegensatz zum Black-Box-Test wird direkt am Quellcode geprüft. Kontrollfluss- oder Datenflussanalysen sind typische Beispiele für White-Box-Tests.

Das Aufspüren von Sicherheitslücken mittels White-Box-Tests lässt sich als Untersuchung einer unendlichen Anzahl von Verhalten in einem endlichen Ansatz, basierend auf dem Quellcode einer Anwendung, bezeichnen, was einem Code-Audit gleicht. Um eine solche Untersuchung durchzuführen, ist eine sehr gute Kenntnis der verwendeten Programmiersprachen und der eingesetzten Frameworks notwendig. Mit dieser Untersuchung lassen sich Schwachstellen allerdings unabhängig von der Konfiguration finden. Eine große Herausforderung besteht darin, theoretische Schwachstellen, d. h. real nicht ausbeutbare Schwachstellen, zu eliminieren. Der große Vorteil dieser Vorgehensweise ist offensichtlich: Der Quellcode steht zur Verfügung und

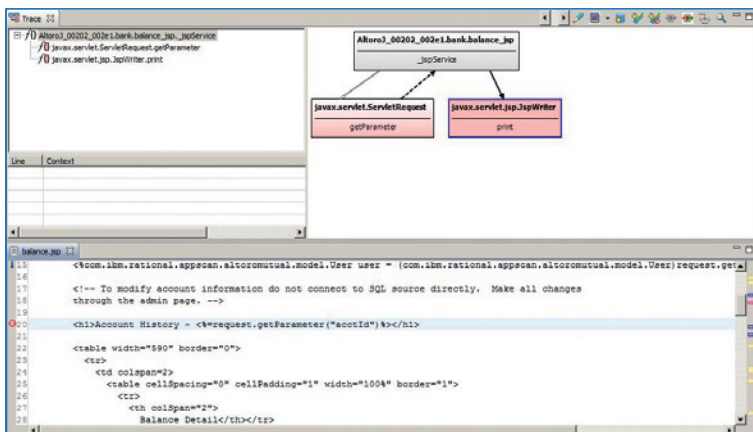


Abb. 3: White-Box-Test-Nachweis für Cross-Site-Scripting

Automatisierung von Black-Box-Tests

Ein Tool zur Automatisierung von Black-Box-Tests senkt den mit manuellen Überprüfungen verbundenen Zeit- und Kostenaufwand und hilft beim Schutz gegen Cyber-Angriffe.

Bei der Auswahl eines Tools für die Automatisierung von Black-Box-Tests ist zu beachten:

- Test-Genauigkeit: Das wichtigste Kriterium eines Black-Box-Testing-Tools ist die Genauigkeit des Scanners, d. h. die Abstimmung des Tests auf die zu testende Anwendung. Dies wird beispielsweise durch eine genaue Analyse der Anwendung, der verwendeten Komponenten und Parameter und einer darauf basierenden Anpassung der Standardtests erreicht.
- Unterstützung von modernen und komplexen Technologien wie JavaScript Frameworks, URL-rewriting, Flash etc.
- Eine geringe False-Positive-Rate verringert den Aufwand für die Nachbearbeitung der Ergebnisse.
- Weitergehende Informationen über Schwachstellen und Integrationen. Dies betrifft insbesondere Erklärungen zu den durchgeführten Tests und den gefundenen Schwachstellen, detaillierte Korrekturhinweise für verschiedene Programmiersprachen und die Anbindung an Bugtracking-Systeme, um Fehler koordiniert zu adressieren.
- Berichte zur Einhaltung von Bestimmungen und Regularien wie: PCI Data Security Standard, Payment Applications Data Security (PA-DSS), ISO 27001 und ISO 27002 sowie Basel II.

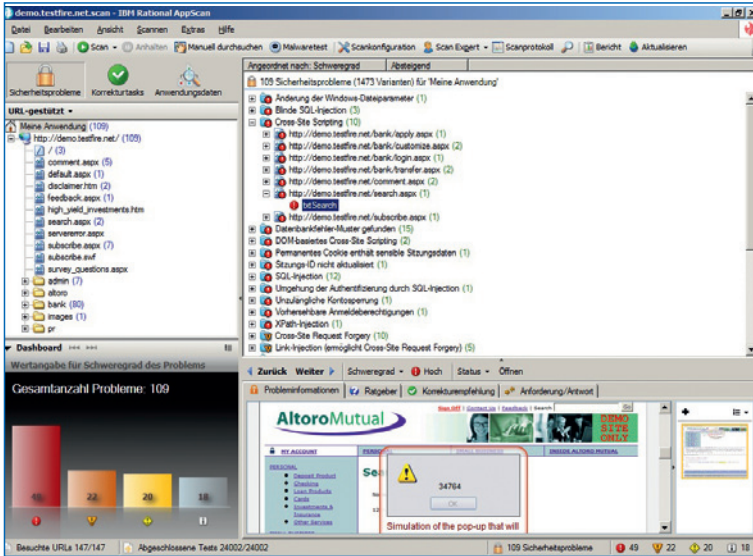


Abb. 4: Black-Box-Testing-Tool: AppScan Standard Edition

Automatisierung von White-Box-Tests

Ein Tool zur Automatisierung von White-Box-Tests hilft Sicherheitsteams dabei, die Anwendungssicherheit zu verstärken, vertrauliche Daten zu schützen und die Einhaltung von Vorschriften zu verbessern. Die Kombination dieses Quellcode-Test-Tools mit der Sicherheitsüberprüfung von Webanwendungen bietet eine umfassende Abdeckung möglicher Sicherheitslücken.

Bei der Auswahl eines Tools für die Automatisierung von White-Box-Tests ist zu beachten:

- Unterstützung einer breiten Auswahl der größten und komplexesten Anwendungen in zahlreichen Programmiersprachen und deren Frameworks.
- Vereinfachte Kooperation zwischen Sicherheit und Entwicklung durch eine flexible Sichtung- und Korrektur

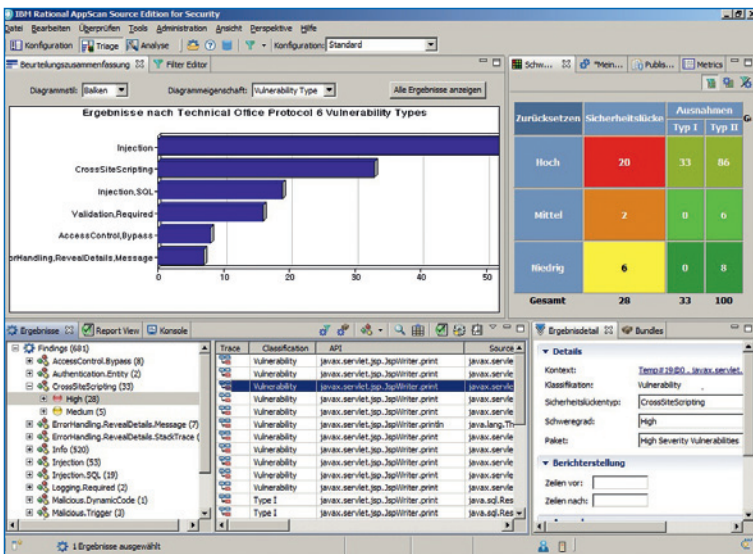


Abb. 5: White-Box-Testing-Tool: AppScan Source Edition

funktion, die den Informationsfluss zwischen diesen Teams automatisiert.

- Einbindung automatisierter Sicherheit in die Entwicklungsprozesse durch die nahtlose Integration der Sicherheitscode-Analyse in Form einer automatischen Überprüfung während des Prozesses der Build-Erstellung.
- Erstellen, Verteilen und Durchsetzen konsistenter Richtlinien und Unterstützung unternehmensweiter Metriken und Berichterstellungsfunktionen in einer zentralisierten Richtlinien- und Analysedatenbank.
- Identifizierung und Korrektur von Sicherheitsfehlern im Quellcode in frühen Stadien des Anwendungslebenszyklus durch die Integration in die IDEs der Entwickler.

Gesamtheitliche Betrachtung von Security

Da sowohl das Black-Box-Testing als auch das White-Box-Testing ihre Vor- und Nachteile haben, ist die Nutzung beider Technologien zu empfehlen. Noch mehr Nutzen bringt die Integration der Prozesse beider Vorgehensweisen. So bringt z. B. ein übergeordnetes Reporting den Vorteil, Schwachstellen sowohl aus der dynamischen als auch der statischen Analyse darstellen zu können. Die sogenannte Korrelation der Schwachstellen – wie sie die AppScan Enterprise Edition oder die Reporting Console unterstützen – erlauben es, besser auf die gefundenen Sicherheitslücken zu reagieren. So sollte man z. B. die Lücken priorisieren, die in beiden Analysetypen gefunden wurden und somit auch sehr wahrscheinlich eine reale Bedrohung für die Webseite darstellen.

Aber auch die Testautomatisierung und die Integration in den Build-Prozess sind sinnvolle Ansätze. Hat sich erst einmal die Nutzung der Security-Tools etabliert, kann man sich wiederholende Aktionen automatisieren. So zum Beispiel die Scan-Konfiguration und den Start des Scan-Prozesses. Wenn der Scan nach dem erfolgreichen Build initiiert wird, kann danach der Security-Experte wie gewohnt die Scan-Ergebnisse und Reports sichten und bewerten. Integriert man als weiteren Baustein die Security-Werkzeuge in die Entwicklungsumgebung, entsteht ein rundes Gesamtbild. Denn wenn Tools in bestehende IDEs der Entwickler eingebunden werden können, dann erhöht dies die Akzeptanz in der Entwicklungsabteilung, da man in der gewohnten Umgebung arbeiten kann.

Werden die genannten Maßnahmen effizient umgesetzt, amortisieren sich die Beschaffungskosten sehr oft innerhalb weniger Quartale. Doch die Nutzung einer Produktsuite von IBM hat noch weitere Vorteile. So arbeiten die AppScan-Produkte auch mit anderen Security-Abteilungen des Unternehmens zusammen. So wird unter anderem das sehr tiefgehende Security-Know-how des X-Force-Teams genutzt.

Sobald die X-Force neue Sicherheitsbedrohungen für Webanwendungen identifiziert hat, werden diese an die AppScan-Entwicklungsteams weitergegeben und Tests schnellstmöglich implementiert. So ist es möglich, Kunden bereits kurz nach dem Auffinden der Schwachstelle ein Update für das Regelwerk zur Verfügung zu stellen.

Aber es gibt weitere Vorteile speziell bei der dynamischen Analyse mit AppScan Standard. So wird beim Testen von Webservices zur Generierung der Oberfläche aus der WSDL sowohl bei AppScan als auch beim Rational Performance Tester der Generic-Service-Client (GSC) genutzt. Dies erlaubt die Testabläufe für Webservices für Security, Funktionalität und Performance in einem einheitlichen Client zu erstellen. Auch bietet AppScan Standard die Möglichkeit, Scan-Ergebnisse in den ISS SiteProtector zu exportieren. Der SiteProtector als Intrusion-Prevention-System kann die Scan-Ergebnisse von AppScan nutzen und gezielt Angriffe auf Sicherheitslücken blockieren, die AppScan als anfällig identifiziert hat. Diesen Blockiermechanismus von Proventia nutzt man solange, bis man anhand der Fehlerbehebungsbeispiele von AppScan die Lücke im Applikationsquellcode behoben hat.

Fazit

Um wertvolle Informationen in Webanwendungen zu schützen, ist es wichtig, die Applikationen durch deren kompletten Lebenszyklus zu testen - von der Entwicklung bis zum Betrieb. Sinnvoll ist es, sowohl dynamische Analysen, als auch Source-Code-Security-Reviews durchzuführen, da beide unterschiedliche Typen von Lücken aufdecken können. Aber auch der gesamtheitliche Ansatz sollte berücksichtigt werden: die Integration für Entwickler, übergeordnetes Reporting, sowie Automatisierung und Integration in den Build-Prozess. Die effektivste Schutzmaßnahme gegen Sicherheitslücken in Anwendungen ist es, diese durch gut durchdachte Prozesse und gezielten Tool-Einsatz gar nicht erst in Produktion gelangen zu lassen. Das Rational AppScan Portfolio bietet alles dies aus einer Hand. ■

Mehr dazu erfahren Sie unter <http://www.ibm.com/software/awdtools/appscan/>.

Tobias Kutzer

seit 2007 im IBM Rational Technical Sales Team in Deutschland. Sein Schwerpunkt innerhalb des Rational Portfolios ist das Quality Management, welches neben dem Testmanagement auch die Aspekte Functional, Performance und Security Testing umfasst. (tobias.kutzer@de.ibm.com)

Michael Kristen

IT Spezialist und seit 2006 bei IBM Rational im Bereich Qualitätsmanagement tätig mit dem Fokus auf Webanwendungssicherheit. (michael.kristen@de.ibm.com)

Impressum

Kontaktadresse für die Beiträge S. 3 bis 38:

IBM Deutschland GmbH
IBM-Allee 1
71139 Ehningen
URL: www.ibm.de

Herausgeber

SIGS DATACOM GmbH

Verlag

SIGS DATACOM GmbH
Lindlaustraße 2c, D-53842 Troisdorf
Tel.: +49 (0) 22 41/23 41-1 00
Fax: +49 (0) 22 41/23 41-1 99
www.sigs-datacom.de
E-Mail: info@sigs-datacom.de

Verlagsleitung

Günter Fuhrmeister

Redaktions- und Herstellungsleitung Zeitschriften

Susanne Herl, Tel.: +49 (0) 22 41/23 41-5 50
E-Mail: Susanne.Herl@sigs-datacom.de

Schlussredaktion:

Heike Weidner

Bildmaterial Cover:

Fotolia.de

Druck

F&W Mediacenter GmbH
Holzhauser Feld 2, D-83361 Kienberg

Abonnenten-Service

IPS Datenservice GmbH, Postfach 13 31
D-53335 Meckenheim,
Tel.: +49 (0) 22 25/70 85-3 74
Fax: +49 (0) 22 25/70 85-3 76
Patrick König, Markus Preis
E-Mail: aboservice@sigs-datacom.de

Erscheinungsweise OBJEKTSpektrum

zweimonatlich

Bezugspreis Sonderheft

Deutschland € 3,80, Europa € 4,50

Bezugspreise OBJEKTSpektrum

Einzelverkaufspreis: D: € 8,90, A: € 9,90, CH: sfr 16,50
Jahresabonnement Deutschland: € 50,90 inkl. Versand
Jahresabonnement Europa: € 58,60 inkl. Versand
Studentenabo: € 43,20 inkl. Versandkosten

Lieferung an Handel

Verlagsunion KG, Postfach 57 07
65047 Wiesbaden, Tel.: +49 (0) 61 23/6 20-0

© 2011 SIGS DATACOM GmbH

4CS-Blackbox-Gerätetest komplettiert die IBM Rational Toolkette

Revisionssicherer Qualitätsnachweis für aktive, sicherheitsgerichtete Softwaresysteme mittels einer technologiedurchgängigen und automatisierten ALM-Toolkette am Beispiel eines Medizinprodukts.

In innovativen Produkten wird ein immer größerer Funktionsumfang softwarebasiert realisiert. Sei es in Geräten der Medizintechnik oder in Kraftfahrzeugen, überall verrichtet komplexe Software größtenteils unsichtbar für den Anwender die Arbeit. Ein bedeutender Bestandteil des Entwicklungsprozesses dieser eingebetteten (embedded) Systeme ist der Nachweis, dass das Risiko, welches von solchen Systemen ausgeht sowie die Fehlerquote tolerierbar sind. So existieren zahlreiche Normen und Richtlinien für sicherheitsbezogene Systeme, die elektrische, elektronische oder Softwarekomponenten enthalten und deren Ausfall ein maßgebliches Risiko für Mensch oder Umwelt bedeutet.

Die internationale Norm IEC 61508 ist eine branchenübergreifende, generische Richtlinie für alle sicherheitsgerichteten, programmierbaren elektronischen Systeme. Darauf basierend existieren für verschiedene Branchen entsprechende Detaillierungen und Spezifika, z. B. ISO 26262 für Automotive oder IEC 62304 für Medizinprodukte. Gemäß der ISO/IEC 62304 „Medical device software – Software life cycle processes“ und MDD 93/42/EWG „Medical Device Directive“ gilt Software als aktives Medizinprodukt, wenn diese dazu bestimmt ist, ein Medizinprodukt zu steuern, zu kontrollieren oder die Funktionen des Gerätes zu beeinflussen. In diesem Fall muss unter Beachtung der geltenden Grundsätze des Software-Lebenszyklus, des Risikomanagements sowie der Validierung und Verifizierung für das Medizinprodukt und somit auch für die Software der betreffende Nachweis geführt werden.

Mit der Rational Workbench für Software und Systems Engineering des Herstellers IBM stehen bereits Werkzeuge für betreffende Phasen des Entwicklungsprozesses zur Verfügung, u. a.

- DOORS für die Anforderungsdefinition,
- Rhapsody für modellbasierte Softwareentwicklung,
- Quality Manager für den Qualitäts- und Testmanagementprozess sowie
- Publishing Engine für das prozessübergreifende Reporting.

Durch die Integration der Rational Werkzeuge in die offene, webbasierte Jazz-Technologieplattform (siehe Exkurs – Jazz) wird die durchgängige Unterstützung des Software-Lebenszyklus eines Produktes ermöglicht.

Die von der Böblinger Firma GADV (Gesellschaft für Automatisierung mit Datenverarbeitungsanlagen mbH) entwickelte Testsuite 4CS (Commandable Component Control & Check System) integriert den Blackbox-Gerätetest in die ALM (Application-Lifecycle-Management)-Toolkette. 4CS komplettiert die IBM Rational Workbench durch eine Anbindung an die Jazz-Plattform, u. a. über den Rational Quality Manager (Abbildung 1).

4CS, zertifizierbar gemäß ISO/IEC 90003, ist eine High-Level-Softwaresuite für das automatisierte, reproduzierbare und revisionssichere Prüfen und Nachweisen des Kommunikations- und Funktionsverhaltens autonom arbeitender oder auch miteinander vernetzter Geräte und Systeme. 4CS unterstützt das aktive Prüfen/Stimulieren der Geräte sowie das Analysieren und Bewerten der erhaltenen Reaktionen und ermöglicht so die Nachweisführung über

- Geräteverhalten,
- bestimmungsgemäße Verwendbarkeit,
- Gebrauchstauglichkeit sowie
- Robustheit.

Darüber hinaus kann 4CS auch nur als Beobachter des Kommunikationsprozesses in einem Netzwerk fungieren (Spy-Funktionalität). Das 4CS-Laufzeitsystem generiert für die zu prüfenden Geräte und Systeme die notwendigen Abläufe, Regeln sowie gerätespezifischen Kommandos. Zur Durchführung des Blackbox-Tests stellt 4CS entsprechende Systemfunktionen (Tabelle 1) sowie die erforderlichen Kommunikationsinterfaces bereit. Der Anwenderfokus liegt auf der durchzuführenden Testaufgabe und nicht mehr auf dem Zusammenstellen des richtigen Prüf-Equipments.

Die Ankopplung an die Jazz-Plattform erfolgt mit der 4CS-offenen Kernel-/Applikationsschnittstelle über den Rational Quality Manager oder im Falle der Test-Case-Modellierung in Kombination mit Rational Rhapsody und den Rhapsody TestConductor.

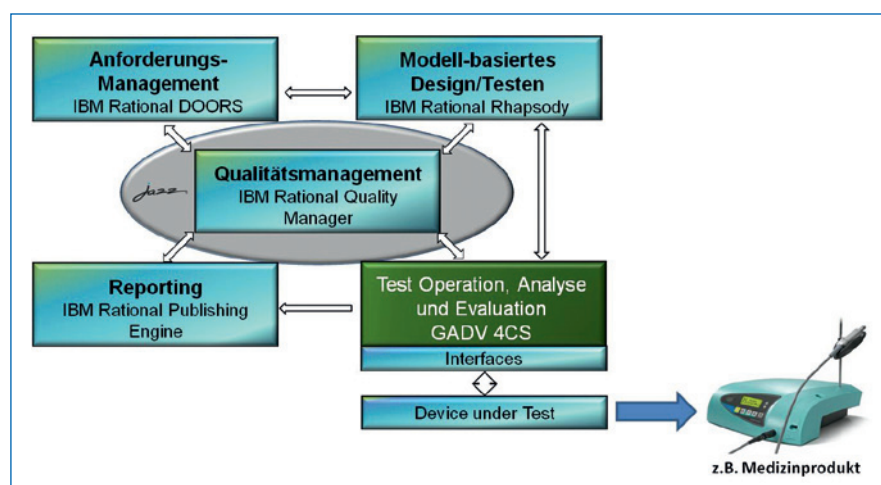


Abb. 1: Technologiedurchgängiges Lifecycle-Management durch 4CS-Integration in die Rational Workbench für Software & Systems Engineering

4CS-Laufzeitsystem		
Interface-Services	Daten-Services	Nachrichten-Services
- funktionsorientiert	- Erkennung	- Senden
- Konfiguration (statisch, dynamisch, programmiert)	- Filterung	- Empfangen (Kognitivmethode)
- Multi-Kanal (Typengleich/-mix)	- Logging	- Gruppenbildung
-hochgenaue Timer	- Antwort-Management	- Alternativen
	- Verteilung	- eindeutige Zeitstempel (adaptive Zeitkorrektur mit Systemuhr)
	- Ausgabe	- Message Queues mit Überlauf-erkennung
	- Protokollierung	- Ereignisverarbeitung (eintreffende Nachricht, Timeout)

Tabelle 1: 4CS-Systemfunktionen

Der Qualitätsprozess und -nachweis beginnt schon im Anforderungsmanagement, in dem jeder Anforderung ein Testfall in einem definierten Testplan zugeordnet wird. Es wird daher sichergestellt, dass während der Entwicklung keine Lücken in der Testabdeckung z. B. durch Kommunikationsbrüche oder Planungsfehler entstehen.

Im Gegenzug ist automatisch ein Informationsfluss aus der Entwicklung oder dem Qualitätsmanagement zurück in das Anforderungsmanagement gegeben. IBM Rational Quality Manager unterstützt und sichert den Qualitäts- und Testmanagementprozess dadurch, dass ein vordefinierter Workflow bei der Umsetzung der Testanforderungen eingehalten wird. Die aus den Anforderungen hergeleiteten Testfälle werden im Rahmen eines Testplanes abgearbeitet. Dazu werden im Testplan (Abbildung 2) nicht nur die erforderli-

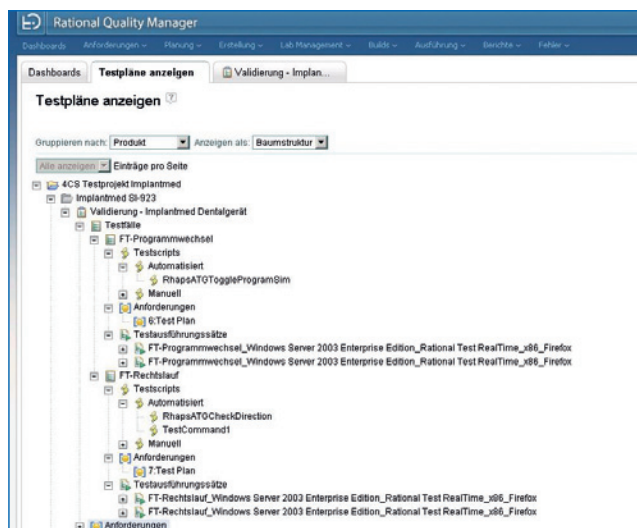


Abb. 2: Testplanübersicht für ein Dentalgerät mit zugeordneten Testfällen, Testskripten und den Anforderungen gegen die getestet werden soll

chen Testfälle dokumentiert, sondern es wird beispielsweise auch definiert, welche Ressourcen, Zeitaufwände, Testressourcen benötigt werden und welche Risiken für die Testausführung bestehen, wenn gewisse Rahmenbedingungen nicht eingehalten werden können.

IBM Rational Rhapsody stellt im einfachsten Fall nur alle ansteuerbaren und zu testenden „Device Under Test-Funktionen“ (DUT) an 4CS für den Blackbox-Test zur Verfügung. Liegt in IBM Rational Rhapsody ein vollständiges Blackbox UML-Modell des zu prüfenden Gerätes vor (Abbildung 3), können mit Hilfe dieses formalen Modells nicht nur die Testfälle vorab validiert werden. Es ist auch möglich, die bei der Ausführung der funktionalen Tests und die im Modell er-

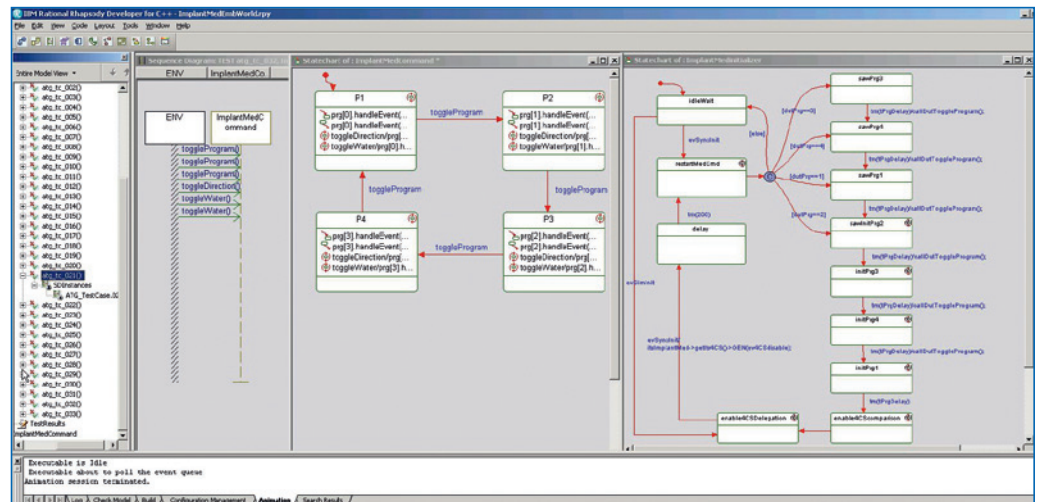


Abb. 3: Modellbasiertes Testen und Simulation eines Dentalgerätes mit Rational Rhapsody und 4CS

reiche Pfadabdeckung zu ermitteln. Optional können unter Verwendung von Rational Rhapsody ATG (Automatische Testfall Generierung) zusätzliche Testfälle generiert werden. Damit lässt sich eine vollständige Pfadabdeckung innerhalb des getesteten Systems erzielen.

Zur Testausführung werden den Testfällen die entsprechenden 4CS „DUT-Testfunktionen“ zugeordnet. Hierbei kann es sich generell um Kommandos, Batchabläufe, automatische oder manuelle Testskripte handeln. 4CS fungiert als Gateway zwischen dem aus Rhapsody generierten generischen Code und dem für das reale Gerät erforderlichen geräte-/prüflingsspezifischen Code. Ergebnisse, z. B. Return-Codes, Console-Outputs oder Log-Files werden in das Qualitätsmanagement zurückgegeben und können dort mittels definierter Kriterien analysiert werden. Ein von der erwarteten Prüflingsreaktion abweichendes Verhalten führt zur Benachrichtigung über ein „unerwartetes Ereignis“ oder zu einer Fehlermeldung und kann in ein Change-Management überführt werden. Die Ausführungsergebnisse können wahlweise im Qualitätsmanagement mittels Berichten oder via Dashboards dokumentiert (siehe Abbildung 4) und analysiert oder automatisch in das Anforderungsmanagement zurückgegeben werden. Damit schließt sich der ALM-Prozess-Kreis und es wird nachvollziehbar, welches Gerät basierend auf welcher Anforderung umgesetzt und auch getestet wurde. Diese Traceability (Rückverfolgbarkeit) ermöglicht die Sicherstellung, dass Anforderungen des Auftraggebers oder aus Normen und Richtlinien nachweislich umgesetzt und beachtet wurden.

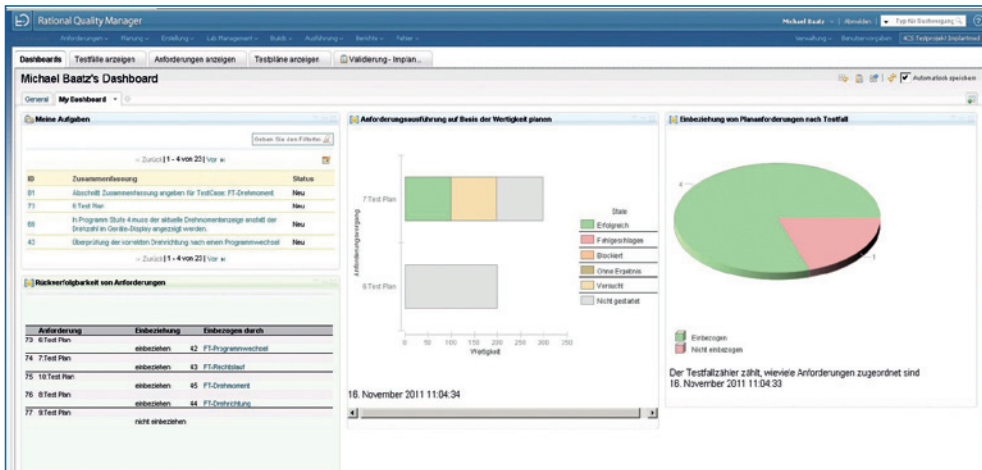


Abb. 4: Quality Dashboard für den Testplan zur Validierung eines Dentalgerätes

Eine Dokumentation im Sinne der Nachweispflicht lässt sich effektiv und schnell erstellen. Der Einsatz ist nicht beschränkt auf die Medizintechnik, sondern kann auch in anderen Branchen (z. B. Automotive, Telekommunikation, Hausleittechnik) erfolgen.

Fazit und Nutzen

Mit der GADV Testsuite 4CS für den Blackbox-Gerätetest wird die technologiedurchgängige IBM Rational ALM-Toolkette auf dem Gebiet Test und Validierung komplettiert. Durch die über die Jazz-Technologieplattform bereitgestellten Standardschnittstellen und die betreffenden Services erfolgt zwischen den prozessspezifischen Tools der Datenaustausch automatisiert und reduziert so die Anzahl möglicher Fehler, da manuelle Eingriffe entfallen. Änderungen, Testablauf, Durchführung und Ergebnisse sind jederzeit reproduzierbar und revisionssicher. Testfälle lassen sich vorab auf Richtigkeit, Konsistenz und Abdeckungsgrad überprüfen. Die Produkte weisen je Produktentwicklungsphase eine deutlich höhere Produktreife auf, das Risiko eines Qualitätsmangels im Einsatz wird minimiert. Sowohl für Einzelgeräte als auch für vernetzte Geräte kann ein Fehlverhalten bei gleichzeitig höherem Testabdeckungsgrad in kürzester Zeit entdeckt werden. Der Test- und Nachweisaufwand kann gegenüber manuellen Methoden und fehlender Tech-

nologiedurchgängigkeit der Toolkette um mehr als 50% reduziert werden. Weitere Informationen, sowie die Aktivitäten der Jazz Community sind im Web unter der Adresse jazz.net zu finden. ■

Quellen:

Beitrag zum Symposium „Die Qualität von eingebetteten Systemen sicherstellen“, Firma GADV und Reutlingen Research Institute (RRI)

www.ibm.com/software/de/rational/jazz

Exkurs – Jazz

Jazz ist die IBM Rational Technologie- und Integrationsplattform basierend auf Web 2.0 Technologien, die dazu konzipiert ist, die Softwareentwicklung interaktiver, produktiver und transparenter zu gestalten, so dass z. B. verteilte Teams, die zeitlich, räumlich oder organisatorisch getrennt sind, enger und effektiver zusammenarbeiten können.

Die Jazz-Technologie basiert auf offenen Web- und OSGi-Standards. Ihre offene und erweiterbare Architektur ermöglicht die Integration von Tools diverser Hersteller sowie von kundenspezifischen Lösungen.

Martin Stockl

ist seit 20 Jahren in der Embedded Real-Time Softwareentwicklung tätig. In seinen ersten sechs Berufsjahren hat er im industriellen Bereich Applikationen mit dem Schwerpunkt „Echtzeit-Bildverarbeitung“ entwickelt, bevor er dann einige Jahre beim Hersteller eines Echtzeitbetriebssystems in Support und technischem Vertrieb gearbeitet hat. Seit 11 Jahren arbeitet er im Bereich Schulung/Consulting/technischer Vertrieb für das Produkt IBM Rational Rhapsody (I-Logix, Telelogic, IBM). (martin.stockl@de.ibm.com)

Gregor Diehl

studierte Informations- und Regelungstechnik an der Technischen Universität Clausthal. Seine Promotion am Institut für Steuerungstechnik der Werkzeugmaschinen (Universität Stuttgart) zum Thema „Steuerungsperipheres Diagnosesystem für Fertigungseinrichtungen auf Basis überwachungsrechter Komponenten“ schloss er 1992 ab. Sowohl vor als auch nach seiner Promotion bekleidete er in verschiedenen mittelständischen Maschinenbauunternehmen leitende Funktionen als Projekt- und Vertriebsleiter. Seit 2004 ist er Geschäftsführer des Softwareunternehmens GADV in Böblingen. (gregor.diehl@gadv.de)

Michael Baatz

ist bei der SVA GmbH tätig und beschäftigt sich seit 1990 mit IBM Software Lösungen, sowohl im Infrastruktur- als auch im Anwendungsbereich. Er war 12 Jahre lang bei der IBM Deutschland GmbH als Softwarespezialist in verschiedenen Softwareabteilungen tätig. Seit vier Jahren ist Herr Baatz als Pre-Sales Consultant auf IBM Rational Software Lösungen fokussiert mit Schwerpunkt auf die Bereiche des Softwarequalitätsmanagements und der Testautomation. (michael.baatz@sva.de)

Testdatenermittlung für Geschäftsanwendungen: SAP-Praxisbeispiel/ Erfahrungsbericht

Geschäftsanwendungen unterliegen fortwährenden Modifikationen. Diese werden entweder durch geänderte betriebswirtschaftliche Rahmenbedingungen ausgelöst, oder durch System-Updates, die zur Fehlerbehebung oder aus Gründen der Systemsicherheit eingespielt werden. Für Systeme, deren Logik vollständig in den Händen des Systemanbieters liegt, ist das ein etablierter und wohlverstandener Ablauf.

Anders ist die Situation bei Anwendungen, die kundenspezifische Erweiterungen enthalten, wie im Falle von Komponenten der SAP-Anwendungssuite. Diese lassen sich durch in der Programmiersprache ABAP verfasste Funktionen und Programme auf vielfältige Weise an die speziellen Anforderungen eines Unternehmens anpassen. Nicht nur das Testen anwendungsspezifischer Module liegt dann in der Verantwortung des Kunden, sondern auch die Überprüfung, ob die Module im modifizierten System korrekt arbeiten. Solche Tests müssen auf dedizierten Testumgebungen durchgeführt werden. Die Modifikationen werden erst nach erfolgreichem Abschluss der Tests auf dem Produktivsystem nachgezogen. In der Regel werden diese Testumgebungen nur temporär benötigt. Daher sind Virtualisierungslösungen [CLD] hierfür besonders geeignet, da sie es erlauben, die für eine Testaufgabe benötigte Rechen- und Speicherkapazität in genau dem benötigten Umfang und für den geplanten Zeitraum zur Verfügung zu stellen.

Für das praxisnahe Testen einer Geschäftsanwendung müssen Daten aus der Produktivumgebung in das Testsystem übertragen werden. Ist der Datenbestand der zu testenden Anwendung nicht allzu umfangreich, so ist dies am einfachsten durch Erstellen einer SAP-Systemkopie möglich. Wenn Bewegungsdaten regelmäßig archiviert werden, kann man so auch bei größeren Systemen vorgehen. Bandbreitenbeschränkungen, Kapazitätsgrenzen der Testumgebung oder Sicherheitsrichtlinien [NIST], die eine Anonymisierung der Daten erfordern, führen in der Praxis oft zu der Einschränkung, dass nur eine Teilmenge der Produktivdaten für den Test zur Verfügung steht. Die Herausforderungen, die sich bei der Auswahl einer geeigneten Teilmenge stellen, sind Gegenstand dieses Artikels.

Geschäftsobjekte

Allen Testabläufen in Geschäftsanwendungen ist gemeinsam, dass sie, anders als reine Systemtests, von Experten aus Fachabteilungen durchgeführt werden, die genau wissen, welche Rolle die zu testende Geschäftsanwendung in den Prozessen des Unternehmens spielt, welche Daten den Abläufen zugrunde liegen und welche Sonderfälle einer spezi-

ellen Behandlung bedürfen. Gerade Grenzfälle spielen eine wichtige Rolle, weil sie möglicherweise von den Entwicklern der neuen oder geänderten Funktionsmodule nicht berücksichtigt wurden. Für den Testplan ergibt sich die Anforderung, die zu testenden Abläufe genau zu dokumentieren und deren Praxisrelevanz von Experten der beteiligten Abteilungen bestätigen zu lassen. Teil der Dokumentation ist die Bestimmung der zu testenden Daten mit besonderem Augenmerk auf Grenzfälle. Wir wollen das an einem sehr einfachen Beispiel aus dem Umfeld 'Gehaltsabrechnung im Personalwesen' illustrieren (Siehe Kasten: Gehaltsabrechnung im Personalwesen).

Das genannte Beispiel illustriert noch einen weiteren Aspekt der Testdatenauswahl: Da der Zuständigkeitsbereich sowie die geographische Zuordnung in die Berechnung eingehen, reicht es nicht aus, den Basissatz der Mitarbeiterstammdaten zu extrahieren. Es werden darüber hinaus beschreibende Daten zur Unternehmensstruktur benötigt, die auch Informationen zu Niederlassungen und deren geographischer Zuordnung liefern. Bei der Teilmengengbildung muss darauf geachtet werden, dass diese referenzierten Daten in der Testdatenmenge enthalten sind. Die zu testenden Daten sind komplexe Geschäftsobjekte, die aus einer Vielzahl miteinander in Beziehung stehender Datensätze bestehen (Mitarbeiter → Abteilung → Rolle der Abteilung im Unternehmen, Einsatzort → Zuordnung an eine Niederlassung → Land). Es reicht nicht aus, eine Teilmenge der Mitarbeiterstammdaten zu extrahieren, sondern darüber hinaus werden die Datensätze benötigt, die Information über den relevanten Ausschnitt der Unternehmensstruktur enthalten. Eine Herausforderung für die Testdatenauswahl besteht nun darin, genau die Kombination von Datensätzen zu finden, die ein 'zusammengehöriges Geschäftsobjekt' beschreiben, in unserem Beispiel eine Instanz des Geschäftsobjekts 'Mitarbeiter und Rolle im Unternehmen'.

Gehaltsabrechnung im Personalwesen

Die Einführung eines neuen Vergütungsmodells hatte eine Änderung des Funktionsbausteins zur Folge, der den Mitarbeiterbonus berechnet. Um die korrekte Funktion der Gehaltsabrechnung sicherzustellen, werden entsprechende Tests vorbereitet. Das Testteam erstellt einen Plan, der den Ablauf der Gehaltsabrechnung beschreibt, um sicherzustellen, dass alle Pfade des betroffenen Moduls durchlaufen werden. Dabei gilt die Rolle, die der Mitarbeiterbonus darin spielt, ein besonderes Augenmerk. Aus dem Plan ist ersichtlich, dass der Bonus für Mitarbeiter verschiedener Geographien auf unterschiedliche Weise in die Gehaltsberechnung eingeht und dass der Zuständigkeitsbereich sowie die Gehaltsklasse einen Einfluss darauf haben, in welchem Maße sich der Bonus auf das Gesamtgehalt auswirkt.

Aus der Zusammenstellung der verschiedenen Berechnungspfade und -formeln leitet das Testteam ab, welche Kombination von Testdaten erforderlich ist, um sicherzustellen, dass alle relevanten Pfade der Geschäftslogik durchlaufen werden.

Die Problematik, die Grenzen eines Geschäftsobjektes zu bestimmen und sicherzustellen, dass alle Informationen vorliegen, die ein Geschäftsobjekt im benötigten Detailgrad beschreiben, ist komplex und nicht allgemein lösbar. In manchen Fällen, wie im dargestellten Beispiel, kann man sich eines Tricks bedienen und gewisse Datenbestände vollständig extrahieren. Dabei nimmt man in Kauf, dass mehr Daten extrahiert werden als für den Test erforderlich sind, solange gesichert ist, dass die benötigten Daten darin enthalten sind. In unserem Beispiel wäre es ausreichend, alle Daten zur Beschreibung der Unternehmensstruktur zu extrahieren, ohne im Einzelnen zu berücksichtigen, welche Datensätze daraus von den zu testenden Mitarbeiterstammdaten referenziert werden. Je nach Umfang der Mitarbeiterstammdaten können auch diese in vollständigem Umfang extrahiert werden, so dass sich die Teilmenge auf eine geeignete Untermenge von Bewegungsdaten beschränkt.

Geschäftsobjekte in SAP-Anwendungen

Idealerweise setzen sich Teams, die Geschäftsanwendungen testen, aus Mitarbeitern der Fachabteilung sowie Experten der Informationstechnik zusammen. Eine wichtige Aufgabe der IT-Experten besteht darin, die technischen Entsprechungen für Geschäftsobjekte in der Datenschicht der zu testenden Geschäftsanwendung zu lokalisieren.

Die Zusammenarbeit von Fachabteilung und IT ist in solchen Bereichen besonders wichtig, in denen es für betriebswirtschaftliche Zusammenhänge keine leicht auffindbaren Entsprechungen in der IT-Welt gibt. Wenn Kundeninformationen, Lagerbestände und Bestellungen durch Fremdschlüsselbeziehungen im Datenmodell miteinander verbunden sind, lassen sich die zum Testen eines Bestellprozesses benötigten Geschäftsobjekte durch Analysieren dieser Beziehungen identifizieren und Anfragen (Queries) für die Extraktion von Testdaten automatisch generieren. Je komplexer die Geschäftsanwendungen sind und je mehr Logik sich in den Schichten oberhalb der Datenhaltung befindet, umso wichtiger ist eine 'Top-Down-Analyse' der benötigten Daten unter Berücksichtigung betriebswirtschaftlicher Zusammenhänge.

SAP-Anwendungen sind hierfür ein gutes Beispiel. Ein wesentlicher Teil der Geschäftslogik in SAP-Anwendungen ist in ABAP-Routinen [ABAP] hinterlegt, die eine Brücke bilden zwischen den in der Datenbank hinterlegten Informati-

onen und den darauf operierenden Geschäftsprozessen. Das logische Datenmodell einer SAP-Anwendung, das von der zugrundeliegenden Datenbank abstrahiert, ist im sogenannten Data Dictionary hinterlegt. Einige der Zusammenhänge zwischen Geschäftsobjekten sind als logische Fremdschlüsselbeziehungen über das Data Dictionary zugänglich und können von entsprechenden Werkzeugen gelesen und analysiert werden. Darüber hinaus gibt es Konfigurationstabellen, die weitergehende Informationen über Attribute bereithalten, wie beispielsweise Wertebereiche und Informationen darüber, ob Attribute zwingend oder optional sind.

Bereits die Information, wie Konfigurationstabellen und Attribute zusammenhängen oder wie ermittelt wird, wann ein Attribut zwingend ist, setzt ein gewisses Verständnis über innere Abläufe voraus und funktioniert nicht in allen SAP-Modulen auf dieselbe Weise. Komplexere Zusammenhänge, etwa welche Kombination von Objektinstanzen für einen bestimmten Typ von Bestellung erforderlich ist, kann am Datenmodell alleine nicht mehr abgelesen werden, weil die betreffende Logik in der Ablaufsteuerung durch Prozesse oder in ABAP-Routinen hinterlegt ist.

Ablauf der Testdatenextraktion

Nachfolgende Illustration veranschaulicht den Prozess der Testdatenermittlung anhand eines einfachen Ablaufs.

Die vier Schritte unterscheiden sich substantiell in dem Grad ihrer Automatisierbarkeit. Wenden wir uns zunächst Schritt 1 zu. Vereinfacht gesprochen geht es dabei um die Auswahl der zu behandelnden Datenquellen (Datenbanken, Schemata, Tabellen, Spalten) basierend auf einer Analyse der vorhandenen Metadaten. Mit entsprechenden Hilfsmitteln lassen sich Beziehungen zwischen Objekten des Datenmodells auf Datenbankebene oder auf der Ebene des SAP Data Dictionary analysieren und für die Bestimmung der relevanten Geschäftsobjekte heranziehen. Unterstützend hilft Expertenwissen über Zusammenhänge, die nicht unmittelbar aus den Metadaten ablesbar sind.

Im zweiten Schritt der Testdatenermittlung wird die relevante Teilmenge insbesondere der Bewegungsdaten bestimmt. Die Bestimmung basiert auf einer Analyse der zu testenden Funktion und der davon betroffenen Prozesse und Abläufe. Auch hierfür ist es wichtig, dass Mitglieder des Testteams über die benötigten betriebswirtschaftlichen Kenntnisse verfügen, um einschätzen zu können, welche Daten benötigt werden, um

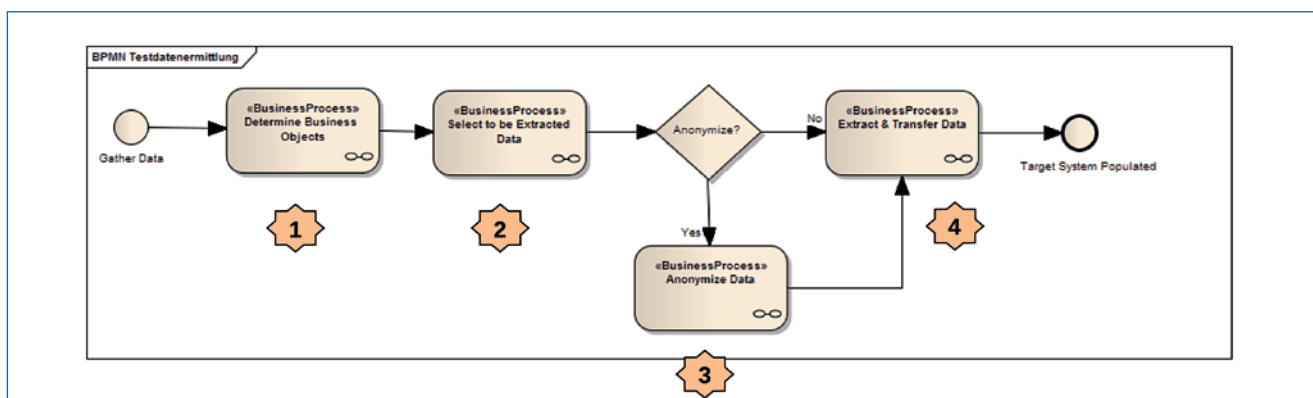


Abb. 1: Einfacher Prozess zur Testdatenermittlung

bestimmte Pfade der zu testenden Logik zu durchlaufen. Für jeden dieser Pfade sind bei einem Funktionstest entsprechende Datensätze zu ermitteln. Bisweilen ist ein Datensatz pro zu durchlaufendem Pfad ausreichend. Bei Tests, in denen das Datenvolumen eine wichtige Rolle spielt, kann es erforderlich sein, den vollständigen Datensatz z. B. für einen bestimmten Zeitabschnitt heranzuziehen, also beispielsweise alle abrechnungsrelevanten Daten für ein Fiskaljahr.

In der Einleitung wurde erwähnt, dass Geschäftsanwendungen typischerweise in dedizierten Umgebungen getestet werden, die auf einer vom Rechenzentrum zur Verfügung gestellten Virtualisierungslösung basieren können. In diesem Falle verlassen die zu testenden Daten nicht nur die Produktivumgebung, sondern möglicherweise auch eine abgesicherte Zone, welche die Einhaltung gesetzlicher Datenschutzvorgaben oder dem Schutz von Unternehmensgeheimnissen dient. Mitglieder des Testteams sind möglicherweise nicht befugt, auf die zu testenden Daten (z. B. Gehaltsabrechnung) zuzugreifen. Daher muss sichergestellt werden, dass die Testdaten keine Rückschlüsse auf die Daten erlauben, aus denen sie abgeleitet wurden.

In Fällen wie diesen schließt sich ein dritter Schritt an, in dem die sensitiven Datensätze identifiziert und durch geeignete Mechanismen [OPT01], [OPT02] vor unbefugtem Zugriff geschützt werden. Es ist nicht einfach, sich auf einen Katalog von Kriterien zu verständigen, nach denen eine Menge von Datensätzen als sensitiv einzustufen ist. Auch hierfür reicht es nicht aus, jeden Datensatz für sich genommen zu betrachten. Oft birgt nur die Kombination mehrerer Datensätze schützenswerte Informationen. Das kann beispielsweise eine Verbindung zwischen einem Datensatz sein, der ein Individuum identifiziert, einem geschützten Bereich und einem Authentifizierungsschlüssel, der dem Individuum den Zugriff auf den geschützten Bereich gewährt.

Individuen oder Objekte können durch eindeutige Schlüssel wie Personalnummern oder Materialstamnummer identifiziert werden, oder auch durch eine Kombination bestimmter Eigenschaften, die jede für sich genommen harmlos aussehen. Sind diese Eigenschaften Teil der zu testenden Daten, so kann nicht autorisiertes Personal Zugang zu sensitiven Daten erlangen.

Welche Herausforderungen stellen sich bei der Anonymisierung sensitiver Daten? Generell sollte die Funktion der zu testenden Abläufe durch die Anonymisierung nicht beeinträchtigt werden. Die Konsistenz und Korrektheit der zu testenden Geschäftsobjektinstanzen muss gewahrt bleiben. Werden Daten anonymisiert, die Fremdschlüsselbeziehungen mit anderen Objekten eingehen, so muss sichergestellt sein, dass der Schlüsseltausch konsistent über alle beteiligten Objekte hinweg erfolgt. Datentyp sowie Wertebereich des Substituts müssen korrekt sein.

Nach erfolgter (optionaler) Anonymisierung werden in Schritt 4 Testdaten aus dem Produktivsystem extrahiert und zur Testumgebung übertragen. Je nach Rahmenbedingungen des Tests kann es erforderlich sein, die extrahierten Daten zwischenspeichern, um die Testumgebung einfach wieder auf ihren ursprünglichen Zustand zurücksetzen zu können oder um Testkriterien zu verfeinern, ohne die Schritte 1 bis 3 nochmals durchlaufen zu müssen. Dabei muss der Speicherort denselben Sicherheitsbestimmungen genügen, die auch für das Testsystem gelten.

Fazit

Dieser Artikel beschreibt einige der Herausforderungen, die sich beim Testen von Geschäftsanwendungen ergeben, mit besonderem Augenmerk auf der Auswahl repräsentativer Testdaten. In einfachen Fällen können Datenbankwerkzeuge hierbei eine wichtige Hilfe sein. Bei komplexen, mehrschichtigen Anwendungen verlagern sich die Anforderungen jedoch von einer datenorientierten Sicht hin zu einer prozess- oder anwendungsorientierten Sicht, die wesentlich mehr betriebswirtschaftlicher Expertise bedarf. Von der Vielzahl der auf dem Markt befindlichen Werkzeuge für die Testunterstützung eignet sich nur ein kleiner Teil für das Testen solcher Anwendungen. Dabei ist zu beachten, dass diese Werkzeuge eine betriebswirtschaftliche Sicht auf die zu testenden Anwendungen vermitteln, die für alle Mitglieder des Testteams, die Experten aus den betroffenen Fachabteilungen und aus der IT gleichermaßen nachvollziehbar ist. ■

Literatur & Links

[NIST] NIST Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)
<http://csrc.nist.gov/publications/nistpubs/800-122/sp800-122.pdf>

[OPT01] IBM Optim Solutions 'Simplify Test Data Management'
<http://www-01.ibm.com/software/data/optim/streamline-test-data-management/>

[OPT02] Optim Development Studio 2.2 Series – How to enforce data privacy
<http://youtu.be/DK94WwVHMAU>

[CLD] IBM SmartCloud
<http://www.ibm.com/cloud-computing/us/en/>

[ABAP] Advanced Business Application Programming
<http://en.wikipedia.org/wiki/ABAP>

Dr. Peter Gerstl

Softwarearchitekt im Böblinger Entwicklungszentrum der IBM Deutschland GmbH mit über 15 Jahren Praxiserfahrung im Umfeld von Content Management und Textanalyse. Hat in unterschiedlichen Positionen Projekte betreut, die sich mit Informationsintegration und Data Governance befassen. Sein momentaner Arbeitsschwerpunkt ist die Automatisierung von Risikoabschätzungen in SAP Migrationsprojekten.
(gerstl@de.ibm.com)

Daniel S. Haischt

Softwareentwickler im IBM Deutschland Research & Development Labor Böblingen und setzt sich für agile Softwareentwicklungsmodelle basierend auf der Rational Jazz-Plattform ein. In seiner Freizeit arbeitet Daniel S. Haischt als Committer der Apache Software Foundation an zahlreichen Open-Source-Softwareprojekten und lässt die so gewonnene Erfahrung aus dem Open-Source-Umfeld hinsichtlich Test-Automatisierung & Continuous-Integration mit in die täglich Arbeit bei IBM einfließen.
(daniel.haischt@de.ibm.com)

Warum die Tester oft unschuldig sind – häufige Fehler beim Modultest

Testen kostet Geld, nicht testen noch mehr. Am teuersten aber ist schlechtes Testen – wenn man schon so viel Geld ausgibt, dann sollte man mehr bekommen, als nur einen bunt bedruckten Berg Papier. Im Artikel finden sie einige Beispiele aus der Praxis, was man alles falsch machen kann und wie sich die Probleme vermeiden lassen. In vielen Fällen hat man schon verloren, bevor die Tester überhaupt angefangen haben.

Wer über Modultests sprechen will, sollte zuerst definieren, was er unter einem „Modultest“ versteht. In diesem Beitrag geht es dabei um die unterste Testebene, die man oft auch als „Entwicklertest“ oder „Unit-Test“ bezeichnet. Es gibt leider keine einheitliche Terminologie – wer mit zwei Kunden spricht, hört wahrscheinlich drei unterschiedliche Begriffe, die für diese Tätigkeit benutzt werden. Ebenso verhält es sich mit den „Modulen“, die man manchmal auch Units oder Komponenten nennt – hier sind damit die kleinsten Einheiten gemeint, die getestet werden. Da auch Begriffe wie „Spezifikation“, „Anforderung“, „SW-Requirement“ oder „Design-Dokument“ unterschiedlich verstanden werden, werde ich vereinfacht über Spezifikation sprechen, ohne mich dabei auf die Begriffe aus der theoretischen Informatik zu berufen.

„Anforderungen haben wir irgendwo“

Wenn Probleme im Bereich von Modultest untersucht werden, findet man sehr oft die Ursachen in den anderen Disziplinen der Softwareentwicklung. Fangen wir beim Anforderungsmanagement an: Ohne Spezifikation ist es mit dem Testen so eine Sache – wer nicht weiß, wo er hin will, merkt in der Regel auch nicht, ob er ankommt. Und Testen bedeutet nicht einfach „Fehlersuche“, sondern der Abgleich von Spezifikation und Software. Wo keine vernünftige Spezifikation existiert, lässt sich das Testen höchstens vortäuschen. Die unglaubliche aber authentische Aussage eines Projektleiters: „Anforderungen haben wir zwar irgendwo, ich weiß aber viel besser, was die Software machen soll“ macht klar, dass es die Tester nicht überall leicht haben.

Nehmen wir aber an, die Spezifikation existiert, und wurde sogar genug verfeinert und ausgearbeitet, um Modultests zu ermöglichen. Dann brauchen wir noch etwas zum Testen – und zwar die Module. So gelangen wir zur Architektur, wo die Komponentenstruktur definiert ist. Wenn die Module nur auf dem Papier existieren, dann wird es spätestens jetzt schwierig. Beim Modultest wollen wir doch die einzelnen Module isoliert betrachten und tes-

ten. Dabei sind wir gezwungen, die Schnittstellen zwischen einem Modul und deren Umgebung mit Daten zu versorgen. Bei vernünftiger Architektur hält sich die Anzahl von solchen Verbindungen in Grenzen, da die Module in sich geschlossene Einheiten bilden. Wehe aber die Modulgrenzen sind zufällig gewählt und die Software lässt eine vernünftige Architektur vermissen – dann explodiert die Anzahl von zu versorgenden Schnittstellen und der Modultest wird extrem aufwändig bis unmöglich. Mit dem Ergebnis, dass die Tests extrem unproduktiv verlaufen.

Damit der Entwickler den Tester nicht sabotieren kann

Gehen wir davon aus, dass die Architekten ihre Hausaufgaben gemacht haben. Aber auch die Entwickler können die Arbeit der Tester sehr effektiv sabotieren. Manchmal wird es zum Problem, dass es zu wenige zugängliche Schnittstellen gibt. Gut versteckte Variablen (in C zum Beispiel als „static“ innerhalb einer Funktion deklariert) können wichtige Daten verbergen – wie soll ich denn deren Wert überprüfen, wenn ich mit meinem Test-Tool nicht an sie herankomme? Die Komplexität von einzelnen Funktionen sollte im Rahmen bleiben. Wenn empfohlen ist, dass eine Funktion mit maximal 10 Testfällen testbar sein sollte, die Metrik aber stattdessen zeigt, dass sich viele Funktionen im Bereich von 10 bis 20 Testfällen bewegen, ist das noch nicht tragisch. Bei 450 notwendigen Testfällen für eine einzige Funktion ist man jedoch sprachlos. Denn wer soll hier noch den Überblick behalten?

Auch die Performance-Optimierung soll klare Grenzen haben. Hier wäre es angebracht, diese Monsterfunktion in mehrere aufzuteilen, auch wenn dann das Programm ein wenig größer oder geringfügig langsamer wird. Hier hilft es, wenn die Entwickler selbst testen müssen und dadurch sehen, welche Folgen manche Entscheidungen für die Testbarkeit haben. Darüber, ob sie die eigenen Module oder die Module von Kollegen testen sollten, lässt sich streiten, aber Hauptsache sie testen. Und wenn sie schon nicht selber testen, sollten sie zumindest über Tests gut Bescheid wissen und das Test-Tool kennen. Wenn ein Entwickler einen Test-Case verstehen, modifizieren und starten kann und das Testergebnis interpretieren kann, ist das von großem Vorteil. Die gefundenen Fehler muss ja auch jemand beheben, oder gegebenenfalls nachweisen, dass der Tester mit dem Test falsch lag. Wer denn sonst, wenn nicht der Entwickler? Und nicht zuletzt: Wenn zwischen Tester und Entwickler eine Personalunion herrscht, kann der eine kaum die Arbeit des anderen sabotieren.

Auch in der Projektplanung und Leitung kann einiges schief gehen – der Klassiker hier ist zu spät und mit zu wenigen Ressourcen anzufangen. Hier kann man zum Beispiel Aussagen wie diese hören: „An dem Modul arbeiten wir seit fünf Jahren und es existiert in Dutzenden Subvarianten – dann ist es mit ihrem Tool aber kein Problem das in sechs Wochen komplett zu testen, oder?“

Noch schwieriger wird es, wenn das Testteam weit von der Entwicklung entfernt ist. Bestimmt man intern nur Neulinge zu mehr oder weniger freiwilligen Testern, wird

es auch nicht besser. Egal ob die Modultests intern oder extern vergeben werden, muss man dafür sorgen, dass die Tester wirklich gut über die Entwicklung Bescheid wissen. Und nicht dass sie (wie leider so oft) einfach nur die Quellen mit - oder sogar ohne - Spezifikation über den Zaun geworfen bekommen. Die Tester haben so nicht die notwendigen Kenntnisse, die Tests zu schreiben. Reine Programmier- und Test-Toolkenntnisse reichen hier nicht aus. Man muss nicht nur die zu testenden Module gut verstehen, sondern auch die Problematik dahinter, sonst gewinnen die Tester die Sollwerte einfach durch Reverse-Engineering. Wenn die Tester die Spezifikation nicht haben oder nicht richtig verstehen, lassen sich Abweichungen von Spezifikation höchstens mit Hellseherfähigkeiten entdecken.

Geheimtipp: Testen Sie die Firma

Mein Geheimtipp lautet daher: bevor Sie sich irgendwo als (Modul)-Tester bewerben, versuchen Sie herauszufinden, wie es mit den oben genannten Punkten in dem Projekt und in der Firma generell bestellt ist. Sonst kann es Ihnen passieren, dass Sie von Anfang an auf verlorenem Posten stehen.

Bedeutet das aber, dass die Tester bei Problemen unschuldig sind? Keinesfalls, auch wenn die Entwicklung gut strukturiert ist und nicht unter mangelhaftem Projektmanagement leidet, bleiben immer genug Möglichkeiten, wo es direkt beim Testen gründlich schief gehen kann. Ein Problem, das oft vorkommt, nenne ich „Coverage-orientierte Tests“. Die Testabdeckung zu messen ist ja beim Testen notwendig, darf aber nicht zum Selbstzweck werden. Man vergisst sehr oft, dass die Testabdeckung „nur“ eine Hilfsmetrik ist, die sagt, ob ich schon fertig bin. Wenn ich 100% Coverage erziele, aber gleichzeitig meine Spezifikation ignoriere, sind meine Tests wertlos. Leider wird oft die gewünschte Testabdeckung für sich schon als Ziel definiert.

Schauen wir uns genauer an, wie ein „Coverage-getriebenen“ Tester denkt. Normalerweise sollte man ja beim Test einer Funktion zuerst aus der Spezifikation ermitteln, was diese Funktion zu tun hat, dann entsprechende Tests implementieren und dabei die Coverage messen. Bei Teilen der Funktion, die nicht aufgerufen wurden, sollte man sich fragen, wozu sie gut sind. Und: Welchen Teil der Spezifikation soll dieser Code implementieren? Erst dann sollte man die Tests entsprechend der Spezifikation erweitern und im Coverage-Bericht nachschauen, ob die Überlegung richtig war. Der Coverage-getriebene Tester sucht aber oft Abkürzungen. Wenn die ersten Test-Cases gelaufen sind und rote, nicht abgedeckte Teile im Source bleiben, wird kurzerhand geschaut – wie komme ich denn dahin? Aha, zum Beispiel wenn Parameter „x“ auf 10 und Variable „y“ auf 25 gesetzt werden. Und was kommt dann raus? Das kann ich doch im Debugger feststellen oder gar im ersten Testlauf – hm, 30, wird schon richtig sein, also schreibe ich in den Test-Case, dass ich 30 als Rückgabewert erwarte. Was „x“ und „y“ überhaupt bedeuten, muss man nicht wissen, das würde einen nur von Arbeit ablenken, oder?

Um solche „Tests“ zu schreiben, brauche ich keine Fachkenntnisse, ich muss die zu testende Software nicht verste-

hen. Und es geht noch schlimmer – im Zweifelsfall könnte ich sogar die Ergebnisprüfung weglassen oder so trivial gestalten, dass der „Test“ so gut wie nie schiefgehen kann. Und fertig ist das Testskript, das natürlich immer klappt, da wir alles aus der Source „reverse-engineered“ haben. Dabei bleibt völlig unberücksichtigt, ob der untersuchte Programmteil der Spezifikation entspricht und ob der überhaupt notwendig ist. So können sehr schnell ganze Berge von Tests und ausgedruckten Testberichten entstehen, die zwar 100% Coverage und 100% erfolgreiche Test-Cases belegen sollen. In der Realität handelt es sich aber um wertloses Papier.

„Sag mir kurz auf Deutsch, was hier getestet wird?“

Dabei kann man solche vermeintlichen Tests sehr einfach im Review enttarnen. Gute Fragen hierzu sind z. B.: „Sag mir kurz auf Deutsch, was hier getestet wird?“ und „Wo steht das in der Spezifikation?“ Man kann auch beim Test-Review das Modul abändern, versuchsweise einen Fehler einbauen und kontrollieren, ob dies beim Test entdeckt wird. Falls nicht, sollten die Tests ergänzt oder überarbeitet werden. Bitte nicht missverstehen, die Coverage-Daten sind wichtig. Nur muss man dabei auch den Rest im Auge behalten und nicht aus Begeisterung über 95% C0-Coverage alles andere ignorieren.

Kann man also wenigstens hier dem Tester die Schuld in die Schuhe schieben? Nicht ganz, hier geht mein Appell auch an die Auftraggeber, OEM und Prüfer: Verlangen sie nicht nur Modultests mit C0-Abdeckung, sondern sinnvolle Modultests, die 100% der Spezifikation und 100% Coverage (mit welcher Coverage-Tiefe und -Definition auch immer) haben, sonst werden ihre Kunden zu leicht in Versuchung gebracht und die Tester sogar durch Termin- und Ressourcenknappheit zu dem beschriebenen unerwünschten Ergebnissen gezwungen.

Fazit: Nicht nur die eigentliche Testimplementierung ist wichtig, auch das, was vor und nach dem Testen gemacht wird, trägt entscheidend zum Erfolg oder Misserfolg bei. Konsequente Reviews der gelieferten Tests und geordnete SW-Entwicklungsdisziplinen helfen beim Testen mehr, als man erwarten würde. ■

Ladislav Rehak

IBM Rational Technical Sales
(ladislav.rehak@de.ibm.com)

Intelligente Technologien für einen smarten Planeten

Was bedeuten 3 Millionen Zeilen Programmcode für einen Koffer?

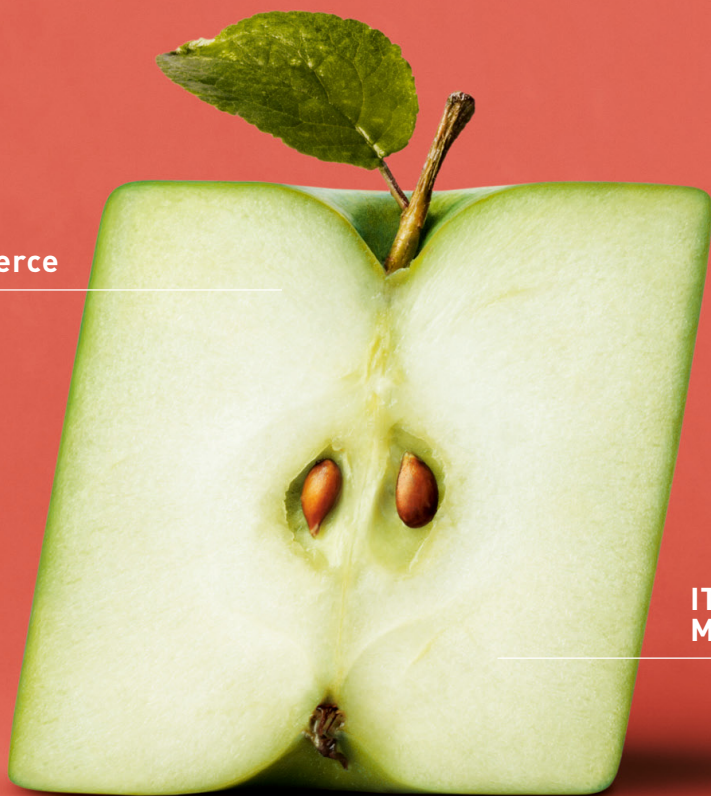
Sie bedeuten, dass man am Flughafen Amsterdam Schiphol bald jedes Jahr 70 Millionen Gepäckstücke zuverlässig und effizient bewegen kann – 20 Millionen mehr als zuvor. Das automatisierte Gepäcksystem ermöglicht es, die Kapazität um bis zu 40% zu steigern. Denn die Nachfrage wächst beständig, und Schiphol ist heute bereits einer der verkehrsreichsten Flughäfen in Europa. Das System basiert auf IBM Rational® und Tivoli® Software und läuft auf Power Systems™ Servern. Ein smartes Unternehmen braucht intelligente Software, Systeme und Services.

Machen wir den Planeten ein bisschen smarter. ibm.com/gepaeck/de



Hier werden Daten sichtbar gemacht, die bei der Gepäckabfertigung am Flughafen Amsterdam Schiphol entstehen.

IT-E-Commerce



IT-Systemhaus & Managed Services

BEGEISTERUNGSFÄHIGKEIT

BEHARRLICHKEIT

ZUVERLÄSSIGKEIT

BODENHAFTUNG

Bechtle verbindet zwei Geschäftsbereiche zu einem großen Ganzen: auf der einen Seite rund 60 IT-Systemhäuser in Deutschland, Österreich und der Schweiz, auf der anderen IT-E-Commerce in 13 europäischen Ländern. Dazu kommen 56.000 Kunden und 5.400 exzellente Mitarbeiter, Kundennähe, Partnerschaft und maßgeschneiderte IT-Lösungen. Außerdem herstellerneutrale Beratung, Beschaffung, Integration, Managed Services und Schulungen sowie 44.000 IT-Produkte und kundenindividuelle Online-Shops.

Ihre bessere IT-Hälfte.

Ganze Arbeit leisten wir auch bei den Bechtle Werten. Deshalb können Sie die Bodenhaftung, Beharrlichkeit, Zuverlässigkeit und Begeisterungsfähigkeit unserer Mitarbeiter in der täglichen Zusammenarbeit spüren. Und auch für unseren Anspruch, Sie immer wieder zu verblüffen, geben wir alles.



Sie suchen eine Software-Assetmanagement-Lösung, die Ihnen in Echtzeit Ihre Server-Lizenzbestände anzeigt?

Ihr starker IT-Partner.
Heute und morgen.

BECHTLE