von frank westphal und
johannes link

mehr zum thema:
www.frankwestphal.de/TestgetriebeneEntwicklung.html
fit.c2.com

FRÜHES UND HÄUFIGES TESTEN

„From Stoplight to Spotlight": Vormals zu einer Phase im Entwicklungsprozess erklärt und oft genug als Schlusslicht des Projekts betrachtet, spielt sich das Testen heute als Aktivität ab und geht bei testgetriebener Entwicklung sogar allem anderen mit rotgrünem Scheinwerferlicht voraus.

In der Softwarebranche breitet sich eine Kluft aus. Auf der einen Seite sehen wir Entwicklerteams, die ihre Software schnell und zuverlässig ändern können. Sie schreiben automatisierte Testfälle für alles, was womöglich schief gehen könnte, und halten ihren Code durch ständiges Refactoring sauber und flexibel. Auf der anderen Seite sehen wir Teams, deren Entwicklungsgeschwindigkeit mit fortschreitender Projektdauer immer weiter sinkt. Sie testen ihr System überwiegend manuell und ihr Code von gestern steht den Anforderungen von heute nicht selten im Weg. Während erstere ihren Kunden dabei helfen, neue Geschäftsideen möglichst schnell in hoher Qualität umzusetzen, stehen letztere unter einem immer höheren Druck und der Drohung, durch „billige" Teams in Offshoring-Ländern ersetzt zu werden.

Entwickler, die das Testen lieben gelernt haben?

Eingeläutet wurde der Wandel beim Testen vor nunmehr sieben Jahren durch *Extreme Programming (XP)* und *JUnit*. Testen war auf einmal cool. JUnits grüner Balken machte süchtig, sorgte er doch für ein lang vermisstes Gefühl: das Vertrauen, dass die produzierte Software tatsächlich wie gewünscht funktionierte. Doch schnellen Testerfolgen folgte erste Ernüchterung. Bestimmte Ecken ließen sich nicht so einfach automatisiert testen, so z. B. grafische Benutzungsschnittstellen, *Enterprise Java Beans (EJBs)*, Datenbanken, *Threads* und *Legacy Code*. Hier zeigte sich dann auch, wer wirklich testgetrieben entwickelte und wer nicht. Denn Testprobleme weisen zumeist auf eine Design- oder Prozessschwäche hin.

Heute gelten diese Herausforderungen als gemeistert. Was sich programmieren lässt, lässt sich auch testen. Der Weg zu dieser Erkenntnis führt über einige Hürden; erfolgreiches Testen verlangt eben mehr als die bloße Beherrschung des Test-Frameworks. Insbesondere zeigt sich meist recht schnell, dass zeitlich nachgeordnetes Testen wesentlich ineffektiver ist als das Schreiben der Tests vor der Implementierung: Zum einen lässt sich Testbarkeit im Nachhinein nur mit Mühe erzwingen, zum anderen können vorab geschriebene Tests das Design lenken. Gerade dieser Vorteil testgetriebener Entwicklung ist jedoch – auch in Projekten mit hoher Testkultur – in vielen Köpfen noch nicht gefestigt.

Wohin geht die testgetriebene Entwicklung?

Nachdem sich die meisten Projekte bisher nur auf Unit-Tests gestützt haben, entdecken viele den Nutzen von System- und Akzeptanztests. Wenn diese Tests aus der Feder des Kunden bzw. der Fachabteilung kommen, kann so eine wichtige Kommunikationslücke zwischen den Domänenexperten, die die Anforderungen kennen, und den Entwicklern, die diese realisieren, geschlossen werden. Hat sich JUnit als Defacto-Standard für Entwicklertests etabliert, so schickt sich derzeit Ward Cunninghams *Framework for Integrated Test (FIT)* an, das kundenfreundlichste Akzeptanztest-Framework zu werden. Der angezeigte Trend geht hier eindeutig in Richtung des ausführbaren Anforderungsdokuments. Man verbindet effektiv die Anforderung mit ihrem Abnahmetest. Geht die Formulierung dieser Tests der eigentlichen Entwicklung voraus, so besteht durchaus die Möglichkeit, das traditionelle Anforderungsmanagement auf den Kopf zu stellen. Dabei verändert sich die Rolle des Testers. Anstatt den Entwicklern wegen tausender Kleinigkeiten auf die Finger zu klopfen, kann er nun mit dem Kunden dessen Anforderungen in testbare und eindeutige Beispiele überführen. Auch auf das Projektmanagement wirken sich die featurebasierten Akzeptanztests positiv aus, stellt diese Testart doch ein untrügliches Mittel dar, um den tatsächlichen Projektfortschritt messen und damit „managen" zu können.

Wie gut ein Prozess wirklich „sitzt", zeigt sich, wenn der Stresspegel zunimmt. Noch sehen wir häufig, dass Tests weggelassen werden, wenn die Zeit knapp wird. Nur wer seine Reflexe soweit verändert hat, dass er trotz Zeitdruck weiterhin an seinen guten Gewohnheiten festhält, hat sich überzeugt: Testen spart Zeit und Geld. Zum einen führen intensive Unit-Tests zu deutlich reduzierten Fehlerraten. Von einigen XP-Teams wird mittlerweile berichtet, dass ihre Software praktisch fehlerfrei sei. So hätten die Anwender, nachdem sich die Software länger als ein Jahr in Produktion befände, noch keinerlei Fehler entdeckt. Zum anderen ermöglichen erst automatisierte Tests eine schnelle, evolutionäre Anpassung des Designs an neue Anforderungen. Die Testautomatisierung erweist sich damit als eine Kosten sparende und Effizienz steigernde Maßnahme, sobald die initiale Lernkurve überwunden ist. Wie nicht zuletzt durch Toyota im Automobilbau demonstriert: Produktivität und Qualität sind keine widersprüchlichen Ziele.

Die Grundannahme, dass Software nun mal „buggy" ist, wird sich sicher noch eine Zeit lang halten, jedoch mehr und mehr entkräftet werden. Wer effektiv testet, wird seine Software jederzeit ausliefern können, ohne dass die Qualität einknickt. Die Zykluszeiten von einer neuen Anforderung zum produktiven Code werden zukünftig drastisch kürzer werden und dadurch neue Geschäftsmodelle ermöglichen. Entwickelt wird in Zukunft nur noch geschäftswertorientiert: Die Anforderung mit der größten Wertschöpfung kommt zuerst. Inkrementelle Entwicklung wird zum Synonym für inkrementelle Finanzierung. Möglich wird dieser Wandel unter anderem durch automatisierte Tests. ■

Frank Westphal
(business@frankwestphal.de)
Johannes Link
(johannes.link@andrena.de)

„Aktuelle Trends der Softwaretechnik" bilden den Schwerpunkt dieser Ausgabe von OBJEKTspektrum. Peter Hruschka und Gernot Starke haben in Zusammenarbeit mit Frances Paulisch eine Auswahl besonders praxisrelevanter Themen getroffen und namhafte Experten als Autoren gewinnen können.

Jens Coldewey, freier Berater mit dem Schwerpunkt agile Entwicklung, Vorstandsmitglied der Agile Alliance.

Thilo Frotscher, freiberuflicher Systemarchitekt, Trainer und Autor mit den Schwerpunkten J2EE, Web-Services und Patterns.

Jürgen Hahn, Trainer und Berater für Systemarchitekturen und OO-Vorgehensweisen bei der Sophist Group, Autor von „UML 2 glasklar".

Dr. Peter Hruschka, Trainer, Berater und Coach für Software- und Systemarchitekturen und IT-Strategie.

Dr. Thorsten Janning, Mitbegründer der Kegon AG mit den Schwerpunkten komplexe IT-Architekturen und marktorientierte Organisation von IT-Dienstleistern.

Janos Koppany, Mitgründer und Geschäftsführer der Firma Intland, dort für kollaborative Softwareentwicklungsplattformen zuständig.

Dr. Arne Koschel, Senior Software Architect, Middleware-Experte, Application Integration Team Lead, Dozent und Autor.

Johannes Link, Projektleiter bei der andrena objects ag in Karlsruhe und Autor des Buchs „Softwaretests mit JUnit".

Mustafa Öztürk, Berater bei der Kegon AG, mit dem Schwerpunkt Innovations- und Technologiemanagement in der IT-Strategie.

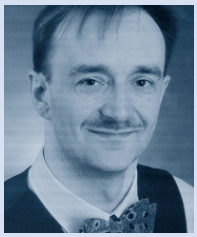
Dr. Stefan Queins, Trainer und Berater für Systemarchitekturen und OO-Vorgehensweisen bei der Sophist Group, Autor von „UML 2 glasklar".

Martin Rösch, Spezialist für innovative Softwaretechnik, Berater, Trainer und Coach.

Chris Rupp, Geschäftsführerin der Sophist Group, mit den Schwerpunkten Requirements-Engineering und Objektorientierung.

Dr. Holger Schwichtenberg, selbständiger Softwarearchitekt, Dozent und Fachjournalist mit dem Schwerpunkt .NET.

Michael Stal, Senior Principal Engineer bei Siemens Corporate Technology, Chefredakteur JavaSPEKTRUM und Koautor der Buchreihe POSA.



Dr. Gernot Starke, Trainer, Berater und Coach für Software- und Systemarchitekturen und IT-Strategie.

Stefan Tilkov, Geschäftsführer der innoQ Deutschland GmbH, einem auf SOA und MDA spezialisierten Beratungshaus.

Markus Völter, freiberuflicher Berater und Coach mit den Schwerpunkten Softwarearchitektur, modellgetriebene Entwicklung und Komponenten-Middleware.

Tim Weilkiens, Trainer und Berater bei der oose.de GmbH, Autor von Buch- und Zeitschriftenartikeln, Mitglied der Arbeitsgruppen zu SysML und UML 2.1

Frank Westphal, freier Softwareentwicklungscoach, Buchautor, Podcaster der agilen Audiokolumne „Tonabnehmer".

Ralf Westphal, freier Autor, Berater, Trainer und Sprecher auf Entwicklerveranstaltungen, einer von 130 Microsoft Regional Directors.