Die Quadratur des Kreises

Schnelle Lasttests für die agile Softwareentwicklung

Gregor Mayer

Kurze Entwicklungszyklen, fachübergreifende Kollaboration, frühe Integration des Qualitätsmanagements in Entwicklungsprozesse sowie der direkte Übergang zwischen Softwareentstehung und Produktivbetrieb prägen die agile Methodik. Angesichts der hohen Prozesstaktung wird die Verankerung von Last- und Performancetests in den Projekten zur Herausforderung. Um sie zu meistern und eine lösungsorientierte Interaktion zwischen Entwicklung und Qualitätssicherung zu gewährleisten, müssen Testverantwortliche in ihrem Verhalten, ihren Methoden und Planungen selbst „agile" werden.

Aufgrund des hohen Kosten- und Zeitdrucks spielen Agile-Methoden* eine immer größere Rolle in der Softwareentwicklung. Kurze Wege zwischen Code-Erstellung, Prüfung und Go-Live beschleunigen die Entstehung von (mobilen) Webanwendungen. Die für viele noch ungewohnte Betonung von Kollaboration und die zahlreichen parallel geschalteten Prozesse können Agile-Projekte allerdings fehleranfällig machen. Das hohe Projekttempo erfordert mehr denn je professionelle Qualitäts- und Performanceprüfungen – und stellt sie gleichzeitig als vermeintlichen Bremsklotz in Frage.

Der Druck zur Code-Ablieferung ist in Agile-Teams hoch: Der Verzicht auf Performancetests bringt allerdings keine Entlastung. Vielmehr erhöhen die Kostenrisiken einer im Produktivbetrieb instabilen Anwendung den Druck zusätzlich. Generell ist ein solides Qualitätsmanagement ausschlaggebend für die Agile-Akzeptanz in Unternehmen. Entwicklerteams setzen Innovationen heute „unglaublich" schnell um. Neue Funktionen werden in die Quellcode-Kontrolle eingecheckt, durchlaufen noch ein Continuous-Integration-Build mit automatisierten Tests, bevor sie in wenigen Minuten auf den Produktionsserver angewendet werden.

Manchen geht das zu schnell. Vor allem Betriebsteams fürchten Anwendungsausfälle als Geschäftsrisiko und betrachten Agile-Verfahren wie Continuous Deployment und DevOps mit Skepsis. Sie sehen bestehende Systeme in ihrer Stabilität und Leistungsfähigkeit gefährdet, wenn Code-Änderungen geradewegs in die Produktion gehen. Gefordert sind daher Performance-Checks, die gründlich sind, ohne Arbeitsabläufe zu belasten – das ist der Kreis, den Tester und Entwickler gemeinsam „quadrieren" müssen.

Verhalten der Testspezialisten

Im Hinblick auf eine hohe Prozesseffizienz müssen Entwickler und Tester ihre Zusammenarbeit enger aufeinander abstimmen als bislang – was in der Praxis nicht immer einfach ist: Denn kurz vor dem Ende eines Sprints produzieren Entwickler mit hoher Geschwindigkeit Code, um noch möglichst viele Aufgaben abzuschließen. Da Tester eine große Bandbreite an Modul-, Regressions-, Last- und Performancetests verantworten, können sie diese Tempoverschärfung kaum mitgehen.

Ein Umdenken ist gefordert. Qualitätsmanagement ist heute nicht mehr nur retrospektiv, sondern begleitet „live" die Entstehung neuer Software. Die Performanceprüfung sollte daher schon im Vorhinein eng mit Entwicklern abgestimmt werden. Tester in Agile-Umgebungen haben dabei den Vorteil, täglich in kurzen Meetings den aktuellen Stand der Entwicklungsprojekte zu erfahren. Die fachübergreifende Interaktion ermöglicht es, die Prüfung der aktuell bearbeiteten Anwendungen frühzeitig abzustimmen. Zumeist sind Testparameter nur geringfügig zu modifizieren, was bereits vorausschauend vorgenommen werden kann. Solange sie mit dem Team eng in Kontakt bleiben, profitieren Tester von ihrem Informationsvorsprung.

Kollaboration & kurze Lösungswege

„Agile" steht für interaktive Problemlösungen jenseits von Silo- und Bereichsdenken. Damit Teammitglieder mit ihrer jeweiligen Expertise bei der Erstellung und Analyse von Prüfroutinen Hand in Hand zusammenwirken, sollten Testtools kollaborativ ausgelegt sein. Die Option, Elemente wie virtuelle Benutzer, Überwachungskonfigurationen, Lastprofile und Testergebnisse mit Teammitgliedern zu teilen, unterstützt diese Arbeitsweise. Grafische Nutzeroberflächen (GUI) für eine intuitive Definition auch von komplexen Szenarien und die Wiederverwendbarkeit grundlegender Skripte (z. B. An- und Abmeldevorgänge) helfen „Performance-Laien", sich zum Beispiel als Entwickler in die Qualitätssicherung einzubringen.

Grundsätzlich sollte jedes Teammitglied mit vorkonfigurierten Grafiken und Filterfunktionen klar verständliche Berichte erstellen und die Resultate als Handlungsgrundlage interpretieren können. Auf diese Weise stehen Entwickler im direkten Dialog mit dem Qualitätsmanagement, um beispielsweise erforderliche Code-Berichtigungen zeitnah vornehmen.

Planung: Einbettung der Testläufe in Agile-Prozesse

Mit beschleunigten Entwicklungszyklen sorgen Agile-Methoden für ein kurzes Time2Market und Wettbewerbsvorteile im umkämpften Markt der (mobilen) Webanwendungen. Damit die erwartete Produktivitätssteigerung tatsächlich erzielt wird, müssen Performancetests nahtlos in die Projektabläufe eingebettet werden. Die Leistungsprüfung aus Zeitgründen zurückzustellen, ist riskant, da Anwendungen im Einklang mit den Service Level Agreements (SLAs) festgelegte Leistungswerte erreichen müssen.

* im Sinne des „Manifesto for Agile Software Development", **http://www.agilemanifesto.org/**

Dennoch widmen viele Agile-Teams der Performanceoptimierung zu wenig Aufmerksamkeit. User Stories werden dann typischerweise aus funktionaler Perspektive geschrieben, zum Beispiel: „Als Benutzer kann ich auf die Schaltfläche ‚Warenkorb anzeigen' klicken und die Seite ‚Mein Warenkorb' anzeigen." Vollständig wäre die User Story allerdings erst, wenn sie auch messbare Performanceanforderungen spezifiziert: „Als Benutzer kann ich auf die Schaltfläche ‚Warenkorb anzeigen' klicken und die Seite ‚Mein Warenkorb' in weniger als einer Sekunde anzeigen, während bis zu 1.000 andere Benutzer gerade die Seite besuchen."

Quantifizierte und damit messbare Leistungsgrößen gehören in die Aufgabenlisten und sollten als Akzeptanztest für jede Softwareerweiterung herangezogen werden. Ähnlich wie Funktionstests sind Leistungschecks ein verbindlicher Punkt in der „Definition of Done"-Agenda. Eine Story- und Code-Modifikation sollte erst als „erledigt" gelten, wenn sie die SLA-Vorgaben erfüllt, zum Beispiel: „Jede Webpage muss mit allen Funktionserweiterungen weiterhin in weniger als einer Sekunde geladen sein."

Integration mit dem Build-Server

Performance-, Rauch- und Regressionstests sollten in den Prüfroutinen für jeden Entwicklungsschritt hinterlegt, automatisiert und am besten direkt mit dem Build-Server integriert werden. Eine vollständige Testeinbettung sieht vor, dass die Ergebnisse sofort im Build-Tool angezeigt und mit den aktuellen Entwicklungsleistungen beziehungsweise Build-Operationen korreliert werden. Auf diese Weise werden Ursachen für Performanceschwächen leicht eingegrenzt und zügig beseitigt.

Die automatisierte Interaktion zwischen Performancetests und Build-Servern hat neben der fachlichen auch eine psychologische Bedeutung: Performancemessungen werden so auf jeder Iterationsstufe zur Routine und eher als Qualitätsstandard verinnerlicht. Führen Agile-Teams den SLA-Nachweis dagegen ausschließlich manuell durch, wird er aus Zeitmangel leicht zurückgestellt oder in den folgenden Sprint verschoben.

Performanceprobleme rechtzeitig finden

Früher wurden Performance-Checks erst zum Ende eines Entwicklungszyklus angesetzt. Schon in klassischen IT-Projekten geriet der späte Testzeitpunkt zum Ernstfall, wenn Entwicklern kaum Zeit für Korrekturen blieb und Termin- oder Qualitätsziele verfehlt wurden. Für die kurzen, fein aufeinander abgestimmten Agile-Prozesse sind Verzögerungen noch problematischer. Kommen sie aus dem Takt, sind Softwareveröffentlichungen kaum noch fristgerecht zu leisten. Aus Terminnot gehen Anwendungen dann häufig trotz ihrer Schwachstellen in Betrieb, wobei man das kaum kalkulierbare Performancerisiko in Kauf nimmt. Was bleibt, ist das „Prinzip Hoffnung" …

Ein kontinuierliches Testen der einzelnen Builds ist dieser Praxis vorzuziehen. Die frühe Feststellung von Performanceproblemen mit umgehender Rückmeldung an die Entwickler reduziert die Kosten für Fehlerkorrekturen. Mehrere Wochen zuvor bearbeiteten Code zu reparieren, bedeutet für Entwickler dagegen einen enormen Aufwand, der sie für aktuelle Aufgaben blockiert.

Entwickler benötigen Rückmeldung – am besten sofort …

Kontinuierliche Integration (CI) und die Engführung von Agile-Entwicklung und Betrieb (DevOps) verlangen einen intensiven Austausch zwischen Produktion und Qualitätskontrolle. Dabei benötigen Entwickler mehr Informationen als nur die, dass ihr Code Performanceprobleme hervorruft: Sie müssen wissen, bei welchem Arbeitsschritt die Leistungswerte zuerst abgefallen sind. Testlösungen sollten daher eine zentrale Sicht auf die Ergebnistrends aus Regressionstests und damit die Performanceentwicklung über zahlreiche Builds hinweg anzeigen. Damit ist häufig schon auf einen Blick erkennbar, bei welcher Code-Änderung die Probleme begonnen haben. Statt sich lange mit der Fehlersuche aufzuhalten, haben Tester und Entwickler das Problem direkt lokalisiert und können sich reaktionsschnell über effektive Reparaturmaßnahmen verständigen.

Testläufe auf Art und Umfang des Builds zuschneiden

CI-Builds, nächtliche Builds und zum Sprint-Abschluss erstellte Builds unterscheiden sich deutlich in ihrem Umfang. Die Skala reicht von einer einzigen an einen Versionskontrollserver übergebenen Änderung (CI-Build) bis hin zu allen im Laufe eines Tages beziehungsweise eines Sprints übergebenen Entwicklungsleistungen. Daher sollten Lasttests an Art und Größe des ausführten Builds angepasst werden. Generell hat es sich bewährt, im kleinen Rahmen zu beginnen.

Bei CI-Builds sollten Tests umgehend nach der Übergabe einer Änderung ausgeführt werden, damit der Entwickler direkt die Auswirkungen auf das System erkennt. Dazu eignen sich am besten begrenzte Performance- und Rauchtests, die die am weitesten verbreiteten Szenarien abdecken, wobei die Last üblicherweise von internen Lastgeneratoren (On-Premise-Servern) erzeugt wird.

Bei nächtlichen Builds sollten bereits Ausnahmefälle simuliert werden, um Performanceprobleme zu ermitteln, die bei CI-Tests unentdeckt blieben. Am Ende des Sprints werden Anwendungen „gestresst", indem das Team Last aus der Cloud maximal skaliert, um Belastungsgrenzen, Fehlerbilder und Erholungszeiten einer Anwendung nach einem Absturz festzustellen.

Methodik: Konsolidierung des Testmanagements

Webanwendungen werden heute mit einer Vielzahl an mobilen Endgeräten, Betriebssystemen und Browsern sowie mit heterogenen Bandbreiten und Netzverfügbarkeiten genutzt. Nur wenige Agile-Teams verfügen über eigene Performance-Ingenieure, die auf Zuruf passende Testskripte für alle diese Sonderfälle erstellen. In einigen Fällen helfen Entwickler sich selbst, schreiben eigene Skripte und produzieren insulare Zwischenlösungen für den Eigenbedarf.

Für Agile-Projekte mit ihrer hohen Interaktion ist das Verfahren wenig geeignet. Ein einheitliches und konsolidiertes Testmanagement ist hier die bessere Lösung. Dazu sollte teamübergreifend ein einziges Tool eingesetzt werden, das es nicht nur Spezialisten vergleichsweise einfach macht, über eine grafische Nutzeroberfläche komplexe Szenarien selbst zu konfigurieren. Performancetests sollten dabei …

- *Netzwerke und Datenraten realitätsnah emulieren:* Da für mobile Geräte zumeist geringere, zudem schwankende Bandbreiten zur Verfügung stehen, gilt es, Performanceprobleme bei Apps besonders sorgsam auszuschließen: Erhöhte Latenzzeiten und Paketverluste fallen bei geringer Signalstärke für mobile Anwender deutlich stärker ins Gewicht als bei der PC-Nutzung im DSL-Standard. WAN-Emulation mit Band-

breitenbeschränkung sowie die Simulation von Latenzzeit und Paketverlust sind unverzichtbar. Sie erst erlauben eine authentische Ermittlung der Nutzererfahrung mit dem Smartphone, das Webinhalte je nach Mobilfunkstandard, Region, Tageszeit und Netzauslastung mit unterschiedlichen Datenraten herunterlädt.

- *beliebige Plattformen und Browser simulieren:* Browser und Endgeräte belasten die Server beim Webseitenaufruf jeweils unterschiedlich stark mit parallelen Anfragen. Mit steigender Anzahl an Serververbindungen können sich dann die Antwortzeiten merklich verlängern. Lasttests sollten daher die Anfragen im erforderlichen Umfang parallelisieren, damit der Qualitätsmanager sieht, wie der Nutzer das Verhalten und die Verfügbarkeit einer Anwendung erlebt.
- *die Anwenderperspektive nachstellen:* Abgesehen vom (Unternehmens-)Intranet, greifen Benutzer fast immer von außerhalb der Firewall auf Webanwendungen zu. Um nachzuvollziehen, wie sich die Performance vom jeweiligen Nutzerstandort darstellt, sollte Last von weltweit verteilten Cloud-Servern erzeugt werden. Dabei wird die gesamte „Lieferkette" von der Browser-Anfrage an Webserver bis zum Download wiedergegeben und das Verhalten von Router, Firewall, Serverweiche oder Loadbalancer gemessen. Die Cloud ergänzt als komplementäre Option interne Tests. Dann lassen sich in „hybriden" Testverfahren die Resultate von dies- und jenseits der Firewall direkt vergleichen, um Schwachstellen schneller verorten zu können.

Glossar

- *Build* – Automatisierte Umwandlung von Code in eine Anwendung.
- *Continuous Integration (CI)* – fortlaufende Zusammenfügung von Komponenten zu einer Anwendung.
- *Definition of Done* – Aufstellung der Mindestanforderungen, die Entwicklungsleistungen erfüllen müssen, um als „erledigt" zu gelten.
- *DevOps* – Engführung von Entwicklung (Development) und Betrieb (Operations), sodass Agile-Teams gleichzeitig die Softwareentstehung und ihre stabile Nutzung verantworten.
- *Performancetests* – Prüfung von Verhalten und Lasthärte einer Applikationen im simulierten Produktivbetrieb.
- *Rauchtests* – Erste, schnelle Vergewisserung, ob eine Entwicklungsleitung grundsätzlich die Anforderungen erfüllt.
- *Regressionstest* – Wiederholte Anwendung derselben Testparameter auf ein jeweils erweitertes System, um eine Beziehung zwischen Software- und Leistungsänderung zu ermitteln.
- *Sprint* – Kurzer Entwicklungszyklus (Intervall) mit Umsetzung einzelner Funktionalitäten oder Anwendungserweiterungen.
- *User Story* – Aus Anwendersicht formulierte Funktions- und Leistungsanforderungen.

Fazit

Agile-Softwareentwicklung soll mit hoher Schlagzahl Anwendungen erstellen, die jederzeit verfügbar sind – ganz gleich, wie viele Nutzer parallel auf sie zugreifen. Das setzt voraus, dass Last- und Performancetests als integraler Bestandteil der Agile-Projekte akzeptiert und verbindlich in den Prozessen verankert sind. Damit das gesamte Team von Lasttests profitiert, sollten …

- Tester sich kontinuierlich mit dem Team abstimmen, Silodenken vermeiden und verständlich aufbereite Resultate mit den Entwickeln teilen;
- Leistungs-SLAs in den „Definition of Done"-Listen hinterlegt werden, damit neuer Code erst freigegeben wird, wenn er die zuvor definierte Lasthärte nachgewiesen hat;
- Testläufe möglichst automatisiert werden und in ihrer Komplexität jeweils an die Build-Größe angepasst werden, um die Produktionsprozesse so wenig wie möglich zu belasten;
- Performanceprüfung und Qualitätsmanagement in einem Tool konsolidiert werden, um die fachübergreifende Interaktion der Agile-Teams zu erleichtern.

Dipl.-Ing. Gregor Mayer ist Territory Manager DACH (D – Deutschland, A – Österreich, Ch – Schweiz) bei dem französischen Unternehmen Neotys. E-Mail: gregor.mayer@neotys.com