Vorteile und Einsatz von frühzeitigen Leistungstests

Eine Möglichkeit zur Optimierung von Softwareerstellungsprozessen

---

Eine Ausarbeitung von Creative Intellect Consulting Thought Leadership

Creative Intellect Consulting UK befasst sich mit der Adaption von Tools und Best-Practices für Softwaretests. In diesem Dokument betrachten wir die Vorteile und Herausforderungen frühzeitiger Tests (Early Lifecycle Testing) in Bezug auf bestimmte technische Testtypen aus dem Bereich der Leistungstests. Ziel des Dokumentes ist es, Verantwortliche in diesem Bereich auf den neuesten Kenntnisstand zu bringen und gleichzeitig die Adaption blockierende Mythen und Hürden zu beseitigen.

Paul Herzlich, Research Analyst, Creative Intellect Consulting

Februar 2011

Kernaussagen

Frühzeitige Tests sind derzeit auf die Funktionstests beschränkt

Frühzeitige Tests sorgen für geschäftliche Vorteile da IT-Organisationen dadurch bei geringeren Kosten mehr erreichen können. Viele Organisationen haben sich jedoch auf die Einführung von frühzeitigen Funktionstests konzentriert und das Potenzial frühzeitiger Leistungstests weitestgehend ignoriert.

Die Wirkungskraft frühzeitiger Tests entfaltet sich jedoch genauso oder gar noch stärker bei Leistungstests

Frühzeitige Leistungstests – auch bekannt als Last- und Performancetests – sorgen auf zwei unterschiedliche Arten für Vorteile:

- Sie sparen Geld bei der Nachbesserung
- Sie senken die Risiken für Überraschungen in späten Projektphasen

Leistungsprobleme haben – möglicherweise noch stärker als funktionale Fehler – das Potenzial, ein Projekt „zurück an den Zeichentisch" zu schicken, beispielsweise dann, wenn Komponenten oder Infrastrukturen nicht die benötigte Leistung bieten. Diese Wirkungskräfte frühzeitiger Tests sind daher gerade für Leistungsprobleme von extremer Relevanz.

Viele Fachleute sind der Meinung, dass zwei große Hürden den effizienten Einsatz frühzeitiger Leistungstests erschweren

Es gibt eine ganze Menge an möglichen Problemen oder Hindernissen bei der Einführung von frühzeitigen Leistungstests. Die meisten dieser Schwierigkeiten entsprechen denen, die man auch bei der Adaption frühzeitiger Funktionstests findet. Es handelt sich großteils um ‚weiche' Probleme mit Personen und ihren Rollen und der Psychologie von Softwareentwicklern. Es gibt jedoch ziemlich ausgereifte Techniken aus dem Change-Management zum Umgang mit diesen Problemen.

Für die meisten Tester und Qualitätssicherungsmanager gibt es im Rahmen von Leistungstests jedoch zwei „Mythen" oder Herausforderungen, die frühzeitige Tests erschweren:

- Leistungstests können nur mit einem vollständigen System durchgeführt werden. Ein solches System existiert jedoch in der Frühphase üblicherweise noch nicht.
- Es existieren keine passenden Leistungsanforderungen für die Komponenten

Beide Herausforderungen lassen sich allerdings erfolgreich meistern.

Die Lösungen für diese Mythen erfordern erstens eine genauere Betrachtung dazu, wie sich Leistungstests zusammensetzen, und zweitens einen pragmatischen Ansatz dazu, wie Sie Ihre Ziele erreichen.

Die Einführung von frühzeitigen Leistungstests ist ein Change-Management-Projekt

Den Weg der frühzeitigen Leistungstests zu beschreiten bedeutet unter anderem, bei den später für diese Tests verantwortlichen Personen eine entsprechende Motivation zu wecken. Sie werden hier kontrollierte Änderungen an allen bestehenden Entwicklungsprozessen durchführen müssen. Sie werden die Unterstützung der entsprechenden Spezialisten benötigen, inklusive der vorhandenen Testfälle für die späten Leistungstests. Und Sie werden Tools adaptieren müssen, mit denen Sie eine Automatisierung erreichen, die bestmöglich mit Ihren bestehenden Entwicklungstools zusammenarbeitet. Die entsprechenden Veränderungen sollten Sie daher nur im Kontext eines sauber durchdachten Change-Management-Projektes angehen.

Die geschäftliche Notwendigkeit

Die meisten Organisationen aus dem Bereich der Softwareentwicklung erkennen die Notwendigkeit von Leistungstests an. Diese Tests werden jedoch im Allgemeinen sehr spät im Lebenszyklus einer Anwendung durchgeführt. Zu viele Organisationen warten sogar bis zur Einführung der Software in der Produktivumgebung.

Studien aus den letzten 30 Jahren zeigen uns, dass Tests im Frühstadium des Lebenszyklus große Vorteile bringen. Zwar bestehen noch immer vereinzelte Widerstände gegen diese Erkenntnis, die Argumente für frühzeitige Funktionstests haben sich jedoch im Wesentlichen durchgesetzt. Trotz dieser generellen Akzeptanz hinken Leistungstests in der Praxis weiterhin stark hinterher. Führende Manager aus dem Qualitätssicherungsbereich beginnen zwar inzwischen mit der Umsetzung entsprechender Tests, es herrscht jedoch noch immer die weitverbreitete Auffassung, dass frühzeitige Leistungstests zwar erwünscht, jedoch unpraktisch oder nicht durchführbar sind.

Die Herausforderungen bei der Umsetzung von frühzeitigen Funktionstests liegen in ‚weichen' Bereichen: Sie drehen sich hauptsächlich um Personen und Prozesse. Nachdem die Vorteile frühzeitiger Funktionstests klar sind, geht es darum, diese Verfahren in den Entwicklungsprozess zu implementieren und die Entwickler auf der Gefühls- und Verstandsebene für die Umsetzung zu gewinnen. Entwicklern widerstrebt traditionell der Gedanke, dass formalisierte Tests Teil Ihrer Arbeit seien. Textbook-Unit-Tests waren daher größtenteils eine reine Wunschvorstellung. Verfahren zur agilen Softwareentwicklung, leistungsstarke IDEs und Unit-Test-Frameworks haben die Lücke zwischen Test- und Entwicklungsaktivitäten jedoch mittlerweile erheblich verkleinern können. Frühzeitige Funktionstests halten in der Branche immer stärkeren Einzug.

Frühzeitige Leistungstests sind ein Verfahren, bei dem es nicht nur die gleichen, ‚weichen' Probleme wie bei frühzeitigen Funktionstests geht, sondern auch um einige ‚harte' technische Einschränkungen. Wie wir mit den ‚weichen' Problemen umgehen können, ist bekannt. Die technischen Probleme sind jedoch anscheinend für alle, bis auf ein paar sehr erfahrene Spezialisten für Testverfahren, schwer zu bewältigen. Eine weitverbreitete Ansicht besagt beispielsweise, dass aussagekräftige Leistungstests nur mit einem vollständigen System in einer mehr oder weniger der Produktivumgebung entsprechenden Umgebung durchgeführt werden können – was in eine frühen Phase des Lebenszyklus nicht gegeben ist. Wäre dies wirklich der Fall, dann wären frühzeitige Leistungstests in der Tat unmöglich. Tatsächlich handelt es sich hier jedoch nicht um technische Einschränkungen, sondern ganz einfach um

intellektuelle Vorbehalte, mit denen man umgehen kann. Kreative, innovative und fortschrittliche denkende Testexperten haben bereits entsprechende Lösungen entwickelt.

Wenn frühzeitige Leistungstests wenig intuitiv sind und im Gegensatz zu bestehenden Elementen stehen, warum ist es dann trotzdem gut, diesen Weg einzuschlagen? Die Antwort ist: Wir engagieren und verpflichten uns, die Softwareentwicklung besser an langfristigen geschäftlichen Anforderungen auszurichten – das heißt, mehr zu leisten und weniger zu kosten. Managern aus den Bereichen Test, Qualitätssicherung und Entwicklung, die sich an diesen geschäftlichen Zielen orientieren, ist klar, dass frühzeitige Leistungstests ein weiteres Tool für das Erreichen dieser Ziele darstellen.

Die Vorteile frühzeitiger Tests

Frühzeitige Tests bieten zwei Vorteile:

- Reduzierung von Kosten und Zeitaufwand
- Besser vorhersehbare Projekte

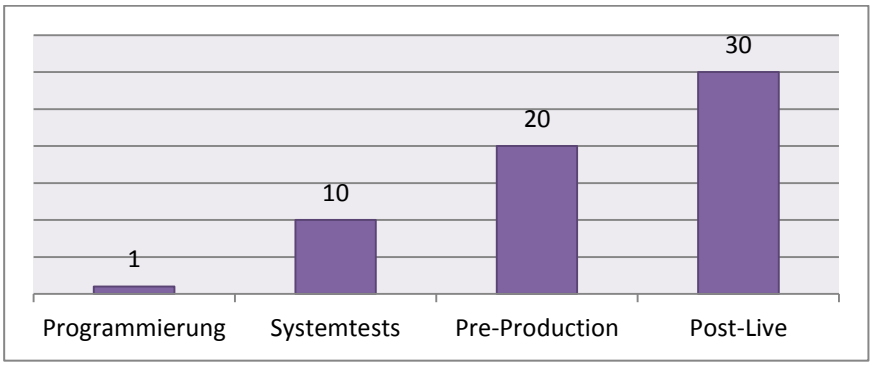
Beide Punkte sind in kommerziellen Umgebungen, in denen der geschäftliche Erfolg davon abhängt, ob die IT die Kosten minimieren und neue Marktanforderungen schnell umsetzen kann, von großer Bedeutung. Unternehmen können sich keine Kostenüberschreitungen und verzögerte Produkteinführungen leisten. Die Zeiten, in denen es oft bis zu 60 Prozent aller Projekte nicht schafften, zum richtigen Zeitpunkt die richtigen Systeme bereitzustellen, sind vorbei. Heute müssen Projektbudgets und Zeitrahmen eingehalten werden. Unsicherheiten sollten auf das Minimum beschränkt sein.

Nachbesserung – Weniger Kosten durch Rückschritte

Viele über die Jahre hinweg durchgeführte Studien kommen zu dem Schluss, dass frühzeitige Tests finanzielle Vorteile bieten. Die umfassende NIST-Studie (National Institute of Standards & Technology) mit dem Titel „The Economic Impacts of Inadequate Infrastructure for Software Testing" aus dem Jahr 2002 zeigt ein Gesamtbild auf. Sie demonstriert, dass die Behebung von Fehlern erheblich teuer wird, sobald der Programmcode vom Entwickler an die System- und Leistungstests übergeben wird. Die Studie sagt hierzu: „Es ist teurer, einen Fehler in der Komponenten- oder System-Entwicklungsphase zu beheben, als den gleichen Fehler schon in der Unit-Phase zu beseitigen, in der er produziert wurde."

Die Studie schätzt, dass die Lokalisierung und Beseitigung eines in der Codeentwicklungsphase eingeführten Fehlers in der Testphase zehnmal mehr kostet. Die Beseitigung in der Pre-Production-Phase kostet 20 mal mehr, und die Beseitigung im Betastadium kostet 30 mal mehr als die Beseitigung in der Unit-Phase, in der der Fehler eingeführt wurde. Programmierer erstellen Programmcode. Dieser Programmcode enthält Fehler. Die Programmierer dazu zu bewegen, diese Fehler zu finden und zu beheben, ist keine unzumutbare Anforderung.

Diagramm 1: Relative Kosten für die Behebung eines Fehlers in den verschiedenen Phasen des Lebenszyklus



Bei Fehlern in den Ausgangsanforderungen sieht das Bild ebenso dramatisch aus. Sich mit diesem Punkt zu befassen, lohnt sich – dann die meisten Fehler gehen aus falsch zusammengestellten oder fehlerhaft definierten Anforderungen hervor. Wenn man die Kosten für das Auffinden eines Anforderungsfehlers während der Anforderungsermittlungsphase (Requirements Capture Stage) mit dem Wert x definiert, dann betragen die Kosten in der Codeentwicklungsphase und der Unit-Test-Phase bereits das Fünffache von x, in den darauf folgenden Phasen das Zehnfache von x, in der Pre-Production-Phase das Fünfzehnfache von x und im freigegebenen System sogar das Dreißigfache.

Wenn Ihnen diese Zahlen übertrieben vorkommen, dann denken Sie einmal an die zum Auffinden und Beheben eines Fehlers erforderlichen Prozesse. Wird ein Fehler nach der Codeentwicklungs- und der Unit-Test-Phase entdeckt, dann muss er dokumentiert und an einen Entwickler zurückgeleitet werden. Ein Entwickler muss dann eine Umgebung einrichten, in der der Fehler reproduziert werden kann. Hierzu kann es erforderlich sein, Testdaten neu zu erstellen und eine ältere Version des entsprechenden Programmcodes wiederherzustellen. Da der Entwickler somit mit einem möglicherweise sehr komplexen System aus Komponenten statt mit einfachen Units umgehen muss, wird das Auffinden des Fehlers – also der Ursache für das eigentliche Fehlverhalten – komplizierter.

Muss sich ein Entwickler um einen Fehler kümmern, so muss er außerdem seine andere Arbeit unterbrechen. Dies hat logischerweise Folgeauswirkungen auf Aktivitäten, die gar nichts mit dem Projekt, aus dem der Fehler stammt, zu tun haben. Ist der Fehler behoben, so muss der entsprechende Programmcode eingeführt und in späteren Phasen getestet werden – und zwar mit allem Aufwand, der für das Quellcode- und Build-Management erforderlich ist.

Es ist somit klar, dass jeder durch die Möglichkeit zur Erstellung und Durchführung eines Tests für eine Unit in der ursprünglichen Programmierumgebung gefundenen Fehler erheblich kostengünstiger zu lokalisieren und zu beheben ist.

Berechenbarkeit – Risiken verringern und Überraschungen vermeiden

Natürlich geht niemand davon aus, dass frühzeitige Tests alle Fehler aufdecken können. Werden die entsprechenden Prozesse effektiv eingesetzt, so profitiert ein Unternehmen jedoch durch kürzere Projektzeiträume und geringere Projektkosten. Frühzeitige Tests haben noch einen weiteren vorteilhaften Effekt: Sie verbessern die Berechenbarkeit der Softwareentwicklung – ein Faktor, dessen

Wert nicht ganz einfach zu definieren ist, der jedoch bei unwillkommenen Überraschungen in späteren Projektphasen schmerzlich vermisst wird.

Frühzeitige Tests tragen zu mehr Berechenbarkeit bei. Sie generieren bereits in frühen Phasen zuverlässigere Informationen zum tatsächlichen Status eines Projektes. Bei Wechsel von der Programmierung zur Integration oder zu Systemtests ist der Programmcode mit erheblich größerer Gewissheit entsprechend ausgereift, wenn die Unit-Tests durch die Entwickler durchgeführt wurden (hier sind echte Unit-Tests gemeint – nicht die mehr oder weniger mythischen Test oder Ad-hoc-Tests, mit denen die meisten Entwickler bisher gearbeitet haben). Bei Programmen, die mit einem vollständigen Satz durchgeführter und bestandener Unit-Tests übergeben werden, kann ein Verantwortlicher erheblich mehr Vertrauen darin setzen, dass die Programmierung vollständig abgeschlossen ist, als bei Programmen ohne dokumentierte Tests.

Dieses Vertrauen kommt nicht nur bei Übergaben innerhalb eines Projektes zum Tragen, sondern auch außerhalb des Projektes. Ohne abschließende Nachweise in Form von bestandenen Tests verfallen auch die erfahrensten Verantwortlichen in ein 95-Prozent-fertig-Syndrom und geben diese Fehlinformationen an ihre Vorgesetzten und Auftraggeber sowie an die Endbenutzer weiter. Eine solche Selbsttäuschung bedeutet, dass es für jeden Beteiligten zu unwillkommenen Überraschungen kommt – und zwar nicht nur in Bezug auf die Softwarequalität, sondern auch auf den Projektfortschritt.

Ein weiteres Beispiel für die immateriellen Vorteile von frühzeitigen Tests wird deutlich, wenn man entsprechende Tests als Technik zur Risikominimierung begreift. Tatsächlich kann man sagen, dass sich alle Tests grundsätzlich um die Minimierung von Risiken drehen. Je früher Sie ein Risiko ausschließen, desto zuversichtlicher können Sie mit den späteren Phasen fortfahren und desto besser können Sie sich auf die späteren Aktivitäten konzentrieren.

Anwendung von Prinzipien für frühzeitige Tests auf die Leistungstests

Wer frühzeitige Funktionstests durchführt, der erkennt, dass diese für klare geschäftliche Vorteile und ein besseres Projektmanagement sorgen. Bringt die Erkennung und Behebung von Leistungsproblemen in frühen Phasen also ebenfalls mögliche Vorteile mit sich? Ja, das tut sie – und zwar in hohem Maße.

Leistungsprobleme müssen wie Fehler behandelt werden. Egal, ob die entsprechenden Leistungsanforderungen explizit oder implizit vorliegen – sobald jemand im Projekt eine nicht ausreichende Leistung deklariert, ist diese Diskrepanz, genau wie jede andere Abweichung zwischen Anforderung und Implementierung, als möglicher Fehler zu behandeln. Es wird genau die gleiche Überprüfung wie bei einem funktionalen Problem erforderlich. Diese Überprüfung wird in der Erkenntnis münden, ob das Problem tatsächlich einen Fehler darstellt oder nicht.

Finden Sie während einer späten Stufe ein Leistungsproblem, dann steigen die Kosten zur Ermittlung der Ursache und zur Überarbeitung der Lösung um ein Mehrfaches. Funktionale Codefehler (solange nicht durch Anforderungsfehler verursacht) lassen sich gewöhnlich auf einzelne und kurze Codesequenzen zurückführen. Werden Leistungsprobleme in späten Phasen entdeckt, dann können diese eine vollständige Architektur oder Implementierung in Frage stellen. Die Problemursache kann in einer Codesequenz, in der Konfiguration oder in der Umgebung zu finden sein. Sind

Programmierfehler mit Auswirkungen auf die Leistung bereits rechtzeitig beseitigt worden, so ist dies bei der Durchführung von umfassenden Leistungstests für eine vollständige Anwendung ungeheuer hilfreich. Gehen Sie entsprechend vor, so können Sie Leistungstests in späten Phasen mit mehr Vertrauen in den Programmcode zuversichtlich entgegenblicken und sich bei diesen Tests auf die Infrastruktur und die Konfiguration konzentrieren.

Was ist mit Leistungstests genau gemeint?

Bis jetzt waren wir nicht sonderlich präzise, wenn wir ganz allgemein über Leistungstests gesprochen haben. Um mögliche Ansätze und Chancen für frühe Codetests im Leistungsbereich erkennen zu können, wollen wir nun weiter ins Detail gehen.

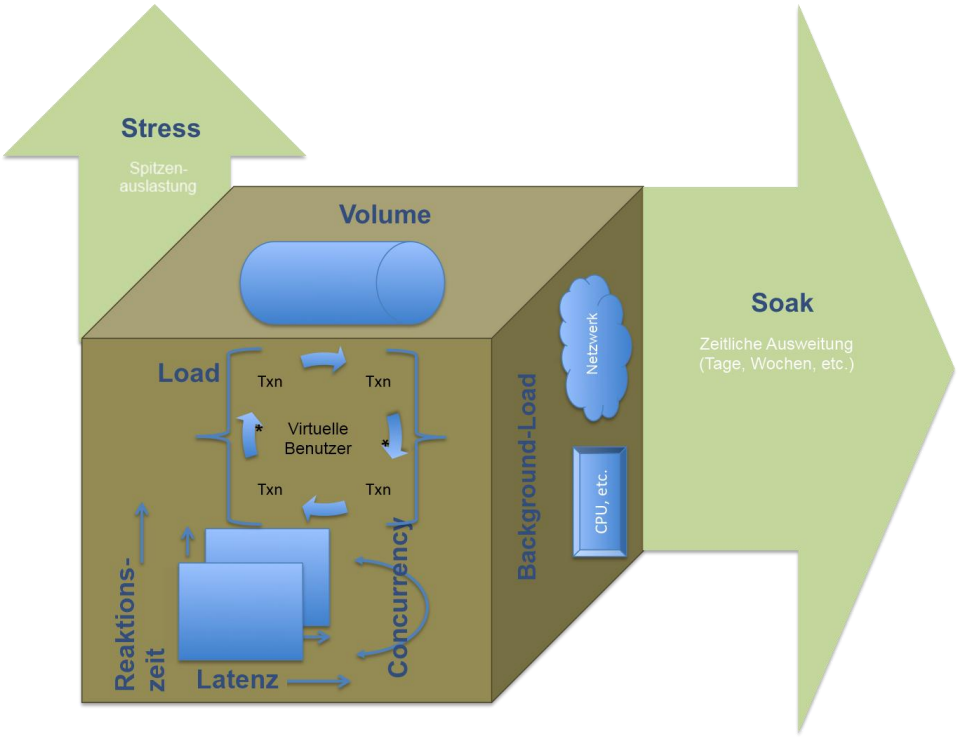
Der Begriff Leistungstest wird im Allgemeinen als Sammelbegriff für mehrere entsprechende Testtypen genutzt. Ziel dieser Testtypen ist es, sicherzustellen, dass ein Endbenutzer eine funktional korrekte Resonanz erhält und dass diese Resonanz in einem akzeptablen Zeitrahmen bereitgestellt wird. Einige der entsprechenden Testtypen sind beispielsweise *Load-Tests*, *Volume-Tests*, *Stress-Tests*, *Concurrency-Tests* und *Soak-Tests*. Um überblicken zu können, welche Testtypen für frühzeitige Tests am besten geeignet sind, müssen wir uns die einzelnen Testtypen näher anschauen.

Concurrency-Tests erstrecken sich auf beide Bereiche – die Funktionalität und die Leistung. Mehr als eine Instanz eines Prozesses gleichzeitig auszuführen, kann zu funktionalen Problemen wie beispielsweise Inkonsistenzen durch gleichzeitige Datenbankupdates und zu Leistungsproblemen führen.

Im engeren Sinne kann man Leistungstests als einen Maßstab für die *Reaktionszeit* oder *Latenz* und die für diese Reaktionszeit oder Latenz erforderlichen Ressourcen betrachten. Um aussagekräftige Ergebnisse zu erhalten, sind Tests unter Bedingungen erforderlich, die denen in der Produktivumgebung so genau wie möglich entsprechen oder die auf die Produktivumgebung abgebildet werden können. Softwareentwickler und Tester führen Reaktionstests daher normalerweise unter Lastbedingungen und mit Datenvolumen aus, die denen der Produktivumgebung entsprechen. Dies gilt gleichermaßen für *Load-Tests* und *Volume-Tests*.

Leistungs-, Load- und Volume-Tests werden üblicherweise gegen Zielwerte auf Basis normaler Auslastungen und Spitzenauslastungen aus Zeiträumen von weniger als einem Arbeitstag gemessen. Wird ein Test mit dem Ziel ausgeführt, ein System an seine Belastungsgrenze zu bringen, so handelt es sich um einen *Stress-Test*. Basieren die Zielwerte auf Zeiträumen von mehreren Tagen, Wochen oder gar Monaten, so handelt es sich um *Soak-Tests*.

Diagramm 2: Beziehungen zwischen verschiedenen Testtypen aus der Familie der Leistungstests



Sie sollten wissen, dass die meisten Leistungstests nicht ohne entsprechende Tools durchgeführt werden können. Sie benötigen Tools zur Ausführung, zur Messung von Aktivitäten und für das Reporting der Ergebnisse. Sogar für Massentests (Crowd-Source-Tests) sind entsprechende Tools erforderlich. Load- und Stress-Tests stellen besondere Anforderungen an die Tools. Es gibt zwei leicht unterschiedliche Ansätze für den Einsatz eines Load-Test-Tools.

Bei Load-Tests versuchen die Tester im Allgemeinen, über ein Load-Test-Tool realistische Transaktionsszenarien zu entwickeln und diese über eine große Anzahl virtueller Benutzer, die mit Datenvolumen arbeiten, die denen der Produktivumgebung entsprechen, auszuführen. Während der Testdurchführungen werden Daten für die Softwarekomponenten und die getestete Infrastruktur gesammelt. Die entsprechenden Aufgaben sind größtenteils logistischer Natur. Sie sind das, was sich die meisten Softwareentwickler unter dem Begriff ‚Leistungstests' vorstellen.

Load-Tests können jedoch auf deutlich einfachere Art durchgeführt werden. Dieses Verfahren bezeichnen wir als *Background-Load*-Tests. Ohne dass versucht wird, eine realistische Benutzeraktivität abzubilden, werden virtuelle Benutzer dazu genutzt, eine Hintergrundaktivität (Background-Load) für die Infrastruktur (Festplattenzugriff, CPU-Nutzung, Netzwerkverkehr) zu schaffen, die einer bestimmten ‚normalen' Auslastung der Produktivumgebung entspricht. Dann werden manuell Benutzertransaktionen durchgeführt, und die Leistung dieser manuell durchgeführten Auslastung wird gemessen.

Wenn wir uns damit befassen, wie frühzeitige Leistungstests durchgeführt werden sollten, dann müssen wir alle verfügbaren Testtypen berücksichtigen und dürfen uns nicht einfach auf das bestehende Konzept fertiger und großflächiger Leistungstests beziehen.

Die Bedenken gegen frühzeitige Leistungstests

Wenn frühzeitige Leistungstests so viel bringen, warum werden sie dann noch immer so selten durchgeführt und bleiben die Domäne einiger visionärer Vorreiter aus dem Test- und Qualitätssicherungsbereich und einiger auf Testdienstleistungen spezialisierter Anbieter?

Das Waterfall-Modell aus dem Entwicklungsbereich und, in geringerem Maße, auch das unter Testern weithin bekannte V-Modell fördert eine Arbeitsweise, bei der technische Tests (wie beispielsweise die Leistungstests) zu einem späten Zeitpunkt des Entwicklungszyklus durchgeführt werden. Zumindest wird dies im Allgemeinen so gehandhabt – auch wenn die Modelle es nicht exakt so vorsehen. Auch die Hersteller von Testtools haben zu diesem Zustand beigetragen. Aus kommerziellen Gründen haben sie sich auf das späte Testen im Lebenszyklus von Anwendungen konzentriert. Aus diesem Grund sind die verfügbaren Tools an entsprechende Vorgehensweisen angepasst.

Weiter oben haben wir die „weichen Faktoren" in Bezug auf Personen und Prozesse erwähnt, die für das Untergraben von frühzeitigen Funktionstests sorgen können. Die entsprechenden Probleme bestehen auch bei Leistungstests: Entwickler mögen keine Tests und sind grundsätzlich nicht für Tests verantwortlich. Ein Mitarbeiter, der für die Durchführung von Tests verantwortlich ist, wird als technisch weniger kompetent betrachtet (dies ist, nebenbei gesagt, bei Leistungs- und Sicherheitstests ganz offensichtlich falsch). Tester haben einen geringeren Status. Sie werden von Natur aus als uninteressant, weniger attraktiv und möglicherweise sogar als Abstieg auf der Karriereleiter angesehen. Und schließlich hat auch noch niemand einen Film mit dem Titel *The Social Testing Network* gedreht.

Agile-Development-Prozesse sind weniger vorurteilsbeladen. Fachübergreifende Teams sind hier der Normalfall, und es ist auch normal, dass Entwickler Tests durchführen. Mit Agile-Development-Prozessen und einem guten Überblick zu den Vorteilen von frühzeitigen Funktionstests und durch bessere IDEs und Unit-Test-Frameworks ist es möglich, erfahrene Entwickler dafür zu gewinnen, Verantwortung für Funktionstests zu übernehmen (unerfahrene Entwickler müssen möglicherweise trotzdem erst ein- oder zweimal auf die Nase fallen, bevor sie sich mit diesem Konzept anfreunden können).

Die Umsetzung von Agile-Development löst das Problem jedoch nicht vollständig. Frühzeitige Leistungstests werfen für die Entwickler zwei wichtige Herausforderungen auf (und zwar sowohl bei Projekten nach dem Waterfall-Modell als auch bei Agile-Development-Projekten):

Herausforderung 1: Leistungstests können nur für ein vollständiges System und in einer Umgebung ähnlich der Produktivumgebung durchgeführt werden.

Herausforderung 2: Es ist unmöglich, auf Ebene einzelner Komponenten aussagekräftige Leistungsanforderungen zu formulieren.

Um diese Herausforderungen zu meistern, sind Ideen motivierter und kreativer Praktiker aus dem Test- und Qualitätssicherungsbereich erforderlich.

Ein pragmatischer Ansatz für frühzeitige Leistungstests

Frühzeitige Leistungstests zu Ihrem Vorteil nutzen zu können, erfordert eine pragmatische Denkweise. Wenn Sie auf hundertprozentige Informationen (beispielsweise bei den Anforderungen) oder auf

absoluten Realismus (beispielsweise bei der Testumgebung) aus sind, dann werden Sie nie loslegen. Sie werden dann niemals von den positiven Aspekten der Überprüfung der Leistung Ihrer Software im Frühstadium des Lebenszyklus profitieren. Ihr Ziel soll es daher sein, mit den vorhandenen Ressourcen so viele Leistungsprobleme wie nur möglich auszuschließen.

Sehen wir uns nun genauer an, wie Sie frühzeitige Leistungstests im echten Leben tatsächlich durchführen können. Es gibt hierbei kein Patentrezept. Unser Ziel ist es, die Möglichkeiten aufzudecken und von der überladenen Komplexität der Load-Tests in späten Phasen des Lebenszyklus wegzukommen.

Auswahl der Testbereiche – Umgang mit Risiken

Sie könnten natürlich versuchen, für jede Unit der Entwicklungsumgebung Leistungstests durchzuführen. Da dies jedoch nicht durchführbar ist (denken Sie gar nicht erst darüber nach), müssen Sie sich auf die Komponenten mit den größten Risiken konzentrieren. Risikoreiche Komponenten sind beispielsweise Elemente mit den folgenden Eigenschaften:

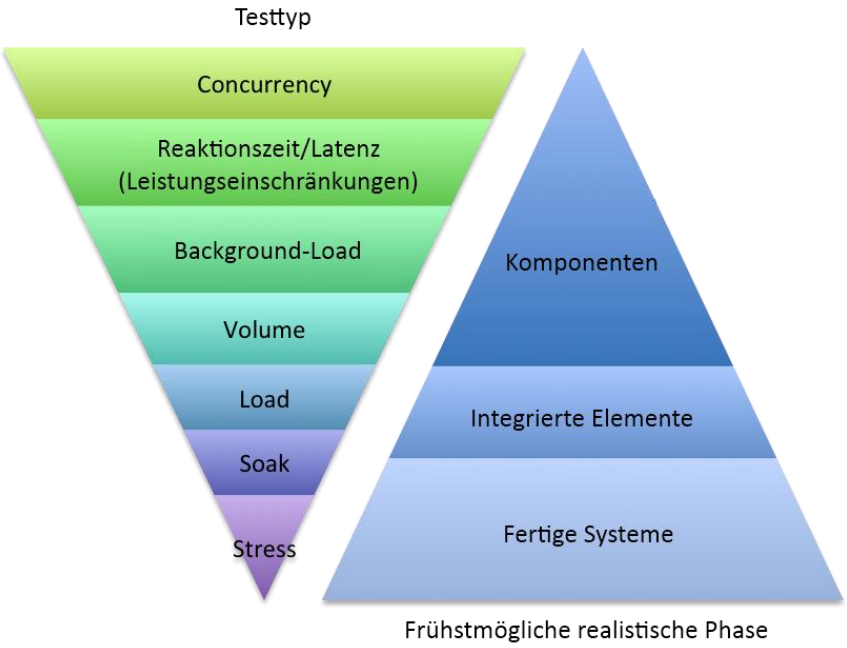
- Ressourcenintensiv (Bandbreite, CPU, Datenträgerzugriffe, etc.)
- Zeitkritisch (Latenz innerhalb der Komponente ist von kritischer Bedeutung)
- Datenbankroutinen (eine permanente Ursache für Leistungsengpässe)

Der Entwickler selbst ist am besten dafür geeignet die technischen Risiken der von ihm erstellten Komponenten zu bewerten.

Auswahl der Testtypen

Kein Testtyp aus dem Bereich der Leistungstests ist für frühzeitige Tests vollkommen ungeeignet. Welche Testtypen genutzt werden, hängt von der Architektur des zu testenden Systems und von den Risikobereichen ab. Die folgenden Bereiche sind bei der Entwicklung eines generellen Prozesses für frühzeitige Leistungstests in einer typischen kommerziellen Softwareorganisation mehr oder weniger geeignet (absteigende Sortierung nach Eignung).

Diagramm 3: Eignung von Testtypen für frühzeitige Leistungstests



Stress-Tests sind keine guten Kandidaten für frühzeitige Leistungstests. Load-Tests (gegen eine Hintergrundauslastung) und Soak-Tests sind hauptsächlich als Teil der laufenden Integration möglich. Alle anderen Testtypen sind für die Durchführung Seite an Seite mit den funktionalen Unit-Tests durch die Entwickler gut geeignet.

Strategien für fehlende Leistungsanforderungen

Die meisten Leistungsanforderungen werden in übergreifenden geschäftlichen Begriffen formuliert – beispielsweise Reaktionszeiten für vollständige Transaktionen, Datenbankkapazitäten und SLAs. Die Entwickler können so ganz einfach argumentieren, dass Leistungstests für einzelne Komponenten ohne Leistungsanforderungen für diese Komponenten, Methoden oder Units nicht möglich sind.

Die offensichtlichste Lösung ist es, die übergreifenden Leistungsanforderungen zu analysieren und so Anforderungen für die einzelnen Komponenten zu erstellen. Ist niemand im Team hierzu in der Lage (nicht einmal einer der Softwarearchitekten oder -designer), dann können Sie hieraus schließen, dass Sie nicht mit Sicherheit sagen können, ob die fertige Anwendung nach der Zusammenstellung aus den einzelnen Komponenten die erforderliche Leistung bieten wird. In diesem Fall wird es sogar noch wichtiger, genauere Leistungsanforderungen zu definieren und frühzeitige Tests durchzuführen.

Tatsächlich stellen wir fest, dass Organisationen, die frühzeitige Leistungstests durchführen, in der Lage sind, aussagekräftige und *funktionierende* Leistungsanforderungen für Komponenten zu erstellen. Abhängig davon, ob es einen Referenzpunkt gibt oder nicht, nutzen diese Organisationen hierzu zwei unterschiedliche Verfahren.

Gibt es kein bestehendes System oder vergleichbare Komponenten, so nutzen sie einen vereinfachten und unkomplizierten Weg zur Entwicklung der Low-Level-Leistungsanforderungen: Sie nehmen die High-Level-Anforderungen und nutzen grobe Durchschnittswerte (beispielsweise für die Anzahl der Methoden pro Transaktion) und Annahmen für die clientseitige Latenz und die Netzwerklatenz.

Gibt es bestehende Systeme, so nutzen sie alte Leistungsdaten als Basis. Agile-Development-Projekte haben hier den Vorteil, dass sie schnell Basisdaten produzieren.

Auch wenn beide Ansätze auf Schätzungen basieren, produzieren sie doch aussagekräftige Ergebnisse (besonders im Zusammenhang mit der Erfahrung und dem Einblick der Entwickler hinsichtlich des Programmcodes). Die aus diesen Tests gewonnenen Informationen zeigen möglicherweise nicht hundertprozentig, ob die Leistung des Programmcodes ausreichend ist – doch wenn ein entsprechender Test fehlschlägt, dann ist dies für den Entwickler ein brauchbares Signal dafür, sich näher mit der Sache zu beschäftigen.

Keine dieser Methoden ist dazu geeignet, den Bedarf an aussagekräftigen Leistungsanforderungen für risikoreiche Komponenten zu verringern oder zu beseitigen. Sie eigenen sich jedoch hervorragend für eine pragmatische Herangehensweise.

Strategien für die Arbeit ohne Produktivumgebungen

Entwicklern steht normalweise keine vollständige Anwendung oder eine Produktivumgebung zur Durchführung frühzeitiger Leistungstests zur Verfügung. Manchmal ist dies allerdings doch der Fall.

Wir tendieren dazu, Änderungen in Softwareentwicklungsprozessen im Kontext ganz neuer Systeme zu betrachten. Die Entwicklung vollständig neuer Systeme stellt jedoch nur einen kleinen Teil der Softwareprojekte dar. Bei der Erweiterung bestehender Systeme ist oft eine vollständige Anwendung für Tests vorhanden (eine Anwendung, die sich unter anderem aus einer oder mehreren neuen Komponenten zusammensetzt). Der Einwand, in frühen Entwicklungsphasen keine vollständige Anwendung zur Verfügung zu haben, ist daher bei weitem nicht so oft zutreffend wie gedacht.

Der Einwand, die Entwicklungsinfrastruktur entspräche nicht in ausreichendem Maße der Produktivinfrastruktur, ist ebenfalls zu entkräften. Es ist machbar, Analysen für einen Vergleich von Entwicklungssystemen mit Produktivsystemen durchzuführen. Kapazitätsplaner führen solche Berechnungen tagtäglich durch.

Lassen wir die Probleme aus dem Prozess-Management im Rahmen der Pflege einer Testumgebung für die Entwickler einmal für einen Moment beiseite. Einige Organisationen verfügen tatsächlich über Testumgebungen, die in einem bekannten Verhältnis zur Produktivumgebung stehen. Diese Umgebungen werden im Allgemeinen für späte Leistungstests genutzt. Sie können jedoch auch für frühzeitige Leistungstests eingesetzt werden. Die Durchführung von Tests außerhalb der Programmierumgebung kann Änderungen am Source-Management und den Build-Verfahren nach sich ziehen. Dies sind jedoch keine unüberbrückbaren Hindernisse.

In anderen Fällen sind die Kapazität und Größenordnung der Entwicklungsumgebung im Vergleich mit der Produktivumgebung bekannt – sie können zum Abgleich der Ergebnisse genutzt werden. Auch hier machen Agile-Development-Projekte Vergleiche einfacher.

In unserem pragmatischen Ansatz bei der Durchführung frühzeitiger Tests können wir mit ungefähren Zahlen arbeiten. Entwicklungsumgebungen bieten nicht so viel Leistung wie Produktivumgebungen. Sie sind jedoch auch nicht so ausgelastet wie Produktivumgebungen. Eine Komponente, die in frühzeitigen Leistungstests schlecht abschneidet, kann als rechtzeitiges Warnsignal dienen und weitere Überprüfungen anstoßen.

Der umfassende Test zum Ende des Entwicklungszyklus ist weiterhin erforderlich

Genau wie Sie nach frühzeitigen Funktionstests trotzdem Systemtests durchführen müssen, müssen Sie in späten Phasen auch weiterhin vollständige Leistungstests durchführen. Sie sollten diese späteren Leistungstests jedoch mit weniger Unsicherheitsfaktoren und mehr Vertrauen in die Leistung des Programmcodes angehen können. Sie werden weniger programmcodebezogene Leistungsprobleme finden. Da Sie so weniger Nacharbeiten durchführen müssen, sparen Sie Zeit und Kosten ein.

Einführung von Testverfahren für frühzeitige Leistungstests

Die Einführung frühzeitiger Leistungstests sollte wie jedes andere Change-Management-Projekt gehandhabt werden. Eine gute Kommunikation und Einfühlungsvermögen sind immer von grundlegender Bedeutung. Es gibt jedoch auch ein paar besondere Punkte, die Sie berücksichtigen sollten.

Motivation fördern

Sie müssen zwei verschiedene Botschaften vermitteln: Die Vorteile frühzeitiger Leistungstests sind den Aufwand wert, und die Tests sind realisierbar.

Startpunkt muss eine klare Erläuterung dazu sein, wie und warum frühzeitig gefundene Fehler sowohl finanzielle Vorteile für das Unternehmen haben und wie die Verwaltbarkeit und Berechenbarkeit des Projektes verbessert werden. Sie müssen außerdem klar machen, dass die Projektzeitpläne an den zusätzlichen Aufwand angepasst werden.

Und welche Vorteile hat die ganze Sache für die Leute, die sie umsetzen müssen? Das hängt von Ihrer Unternehmenskultur ab. Ein klarer Vorteil ist, dass der zusätzliche Entwicklungsaufwand zu weit weniger heiklen Problemen beim späteren Debuggen und Nachbessern führt.

Neben der Vermittlung dieser Vorteile ist es wichtig, den Entwicklern in Bezug auf die Machbarkeit Rückendeckung zu geben. Hierzu müssen Sie sicherstellen, dass ein erfahrener Verfechter oder Evangelist zum Change-Management-Projekt gehört – eben jemand, der Vertrauen weckt.

Den Programmierern sollte im Gegenzug klar sein, dass sie für die frühzeitigen Leistungstests (in welchem Umfang sie auch immer geplant sind) genauso verantwortlich sind wie für die Programmierung und die funktionalen Unit-Tests.

Integration mit bestehenden Prozessen

In diesem Zusammenhang ist der Entwicklungsgrad der Prozesse in der jeweiligen Organisation von Bedeutung. Es ist erforderlich, dass die Organisation bereits mit ausgereiften Testprozessen arbeitet, bevor es dazu kommt, dass frühzeitige Leistungstests eingesetzt werden.

Wir sehen keine Probleme bei frühzeitigen Leistungstests, die spezifisch für Waterfall- oder Agile-Development-Projekte sind. Agile-Development-Projekte verfügen jedoch über einige Charakteristiken, die die Einführung neuer Testtypen etwas einfacher machen. Diese sind unter anderem:

- Die fachübergreifende Natur von Agile-Development-Teams sorgt dafür, dass diese es gewohnt sind, dass Programmierer und Tests zusammenarbeiten.
- Beim Agile-Development stehen die zur Definition von Anforderungen auf Komponentenebene und zur Abbildung der Leistung aus der Entwicklungsumgebung auf die Produktivumgebung erforderlichen Informationen schnell bereit.
- Die Kultur in Agile-Development-Projekten ist allgemein dynamischer und akzeptiert Änderungen und Unsicherheiten leichter. Entwickler in solchen Projekten sind gegenüber unvollständigen Anforderungen pragmatischer eingestellt.

Ihr Prozess sollte eine risikobasierte Analyse umfassen, über die Sie entscheiden können, ob frühzeitige Leistungstests für eine Komponente durchgeführt werden sollen und wie viele und welche Tests (aus den oben besprochen Testtypen) durchgeführt werden sollen.

Es gibt außerdem verschiedene bestehende Prozesse, die wahrscheinlich von frühzeitigen Leistungstests berührt werden:

- Das Testen in der Programmierphase
- Der Build-Prozess und/oder die fortlaufende Integration
- Das Quellcode-Management

Wir würden die Prozesse für die weiter oben ermittelten Testtypen der Programmierphase zuweisen. Es ist außerdem möglich, ausgereifte Integrations- und Build-Prozesse mit integrierten funktionalen Unit-Tests-Suites auf Leistungstests auszuweiten. Die entsprechenden Leistungstests können jegliche Komponentenleistungstests, aber auch zukünftige Leistungstests für die zusammengestellten Komponenten umfassen. Zusätzlich zu den frühzeitigen Backround-Load-Tests, die Sie für Komponenten durchführen können, sind in dieser Phase einige Load-Tests möglich.

Werden Leistungstests außerhalb der Programmierumgebung durchgeführt, so können Änderungen beim Quellcode-Management und bei den Build-Verfahren erforderlich werden.

Um flexibel zu bleiben, sollte der entsprechende pragmatische Ansatz übergreifend betrieben werden. Es ist von grundlegender Bedeutung, dass fehlende Informationen zu den Anforderungen und der Umgebung nicht als Ausrede zur Nichtdurchführung von Tests genutzt werden, solange die entsprechenden Lücken – mit der richtigen Geisteshaltung zu diesem Thema – durch Erfahrung und Sachverstand geschlossen werden könnten.

Fachkräfte

Sie müssen die Mitarbeiter, die normalerweise mit den späteren Leistungstests befasst sind, mit integrieren – inklusive aller Spezialisten für Leistungstests, Architekten und Infrastrukturexperten (Netzwerk, Datenbank, Dienste). Diese Personen müssen ihre eigenen Prozesse verändern, um frühzeitige Leistungstests zu unterstützen. Manchmal werden die hierzu erforderlichen Schritte sogar als Chance gesehen. Die entsprechenden Personengruppen werden nämlich oft direkt mit den Problemen konfrontiert, die durch die Verlagerung aller Tests an das Ende des Entwicklungszyklus entstehen. Da sich die Branche jedoch hin zu einer Ausrichtung auf späte Tests entwickelt hat, kann es

auch durchaus sein, dass die entsprechenden Testspezialisten den Eindruck haben, ihre Aufgaben würden durch frühzeitige Tests untergraben.

Tools

Für alle Leistungstests sind entsprechende Tools erforderlich. Für frühzeitige Leistungstests benötigen Sie Tools, die sich in die Programmier- und Integrationsumgebungen einpassen und bei denen der Unterbau und die Vorbereitung aus Tools für späte Leistungstests fehlen. Die entsprechenden Tools müssen Middleware- und Back-End-Komponenten genauso einfach unterstützten wie clientseitige Skripte.

Eine landläufige Idee ist, dass funktionale Unit-Tests die Basis für Leistungstests bilden können. Dies stellt sich jedoch in der Mehrzahl aller Fälle als falsch heraus – verlassen Sie sich daher besser nicht auf dieses Konzept.

Es sollte gesagt werden, dass für Background-Load-Tests (bei denen Sie nur eine Hintergrundauslastung schaffen, gegen die Sie manuelle Transaktionen ausführen) keine so dermaßen genaue Kontrolle über die Zeitabläufe und den Einsatz automatisierter Skripte erforderlich ist wie dies bei vollständigen Load-Tests der Fall ist. Der Einsatz virtuelle Benutzer kann relativ einfach sein. Sie müssen sich nicht um die Leistung einzelner Transaktionen virtueller Benutzer kümmern. Stattdessen interessieren Sie sich für die Auslastung, die sie insgesamt für die Ressourcen verursachen.

Ein Schlüsselfaktor beim Abwägen der Kosten und Vorteile frühzeitiger Leistungstests sind die Kosten für die Tools. Wenn diese Tools über die IDEs der Entwickler zur Verfügung stehen, sind die Kosten gering. Die Kosten für virtuelle Benutzer zur Durchführung von Load-Tests können jedoch erheblich sein. Es ist wichtig, eine aussagekräftige Schätzung zur Menge der erforderlichen virtuellen Benutzer anzustellen. Zwar gibt es hierbei einerseits eine Tendenz dazu, die erforderliche Anzahl zu überschätzen, andererseits kann es jedoch ein Fehler sein, bei der Menge der erforderlichen virtuellen Benutzer zu geizig zu sein und dann später über den Umfang der Skripte den Versuch zu starten, diesen Fehler auszugleichen. Es ist von grundlegender Bedeutung, dass Ihnen zum Erreichen Ihrer Ziele genügend virtuelle Benutzer zur Verfügung stehen – egal, ob es sich um 25 oder 1000 oder noch mehr handelt.

Einen letzten Punkt sollten Sie sich noch bewusst machen: Sie müssen die Tool-Spezialisten fördern, damit diese andere anleiten können.

Fazit

Frühzeitige Leistungstests sind möglicherweise nicht die erste Verbesserung der Testprozesse, die Organisationen angehen. Es gibt jedoch keinen wirklichen Grund dafür, diese Tests zu einfach zu übergehen. Prozesse aus dem Agile-Development und leistungsstarke Entwicklungsumgebungen mit integrierten Testfunktionalitäten fördern einen entsprechend pragmatischen Ansatz. Geringere Kosten im Rahmen der Nachbesserung und besser verwaltbare Projekte zu realisieren bilden die entsprechende Motivation.