WACHTER & KARBON
IT · CONSULTING

Performancetests im unternehmerischen Umfeld

Dr. Armin Wachter

Performancetests sind physikalische Experimente, in denen die Reaktionen des zu testenden Systems auf künstlich erzeugte Belastungskonstellationen vermessen werden. Als solche unterliegen sie folgenden drei Kardinalforderungen:

Reproduzierbarkeit
Identisch durchgeführte Tests müssen (statistisch) identische Ergebnisse liefern.

Interpretierbarkeit
Testergebnisse müssen verstanden und mit konkreten Erwartungshaltungen konfrontiert werden können.

Zielgerichtetheit
Tests müssen für die Beantwortung der jeweils gestellten Fragen geeignet sein.

In einer idealen Welt würden diese theoretisch motivierten Forderungen den alleinigen Maßstab darstellen, nach denen sich die konkrete Planung, Durchführung, Analyse und Interpretation von Performancetests zu richten haben. In der realen Welt, insbesondere im arbeitsteiligen unternehmerischen Umfeld, kommt jedoch eine weitere, praktische und weniger eindeutige Forderung hinzu, nämlich die der *Umsetzbarkeit*. In ihr spiegeln sich unter anderem die Tatsachen wider, dass Performancetests Geld kosten und dass mehrere Personen und Parteien daran beteiligt sind. Die realistische Betrachtung von Performancetests ist also recht komplex und umfasst einerseits *technische* und *methodische* Aspekte, die vor allem im Lichte der drei Kardinalforderungen zu sehen sind. Auf der anderen Seite haben wir wenigstens auch *logistische* und *organisatorische* Gesichtspunkte zu berücksichtigen, die sich mit Fragen der praktischen Umsetzbarkeit beschäftigen.

In diesem Artikel wollen wir eine überblicksartige Darstellung von Performancetests im unternehmerischen Umfeld geben, indem wir sie aus den Perspektiven Technik, Methodik, Logistik und Organisation nacheinander näher beleuchten. Im ersten Kapitel *Technische Aspekte* adressieren wir die Frage, wie Performancetests grundsätzlich aufgebaut sind und durch welche Faktoren sie beeinflusst werden. Das zweite Kapitel *Methodische Aspekte* umfasst einige zentrale Gedanken zur konkreten Gestaltung, Analyse und Interpretation von Performancetests mit Blick auf deren Zielsetzungen. Im dritten Kapitel *Logistische Aspekte* setzen wir uns mit den an Performancetests typischerweise beteiligten Parteien sowie den von ihnen zu erbringenden Leistungen auseinander. Im vierten Kapitel *Organisatorische Aspekte* versuchen wir auf der Grundlage der vorangegangenen Betrachtungen herauszuarbeiten, welche organisatorischen Maßnahmen sinnvoll sind, um allen vier Forderungen größtmöglich Rechnung zu tragen. Schließlich werden am Ende des Artikels seine wichtigsten Inhalte noch einmal zusammengefasst.

Der Artikel richtet sich in erster Linie an Performance-Laien, die an einer übersichtlichen und verständlichen Darstellung von Performancetests in Unternehmen interessiert sind. Zudem hoffen wir, dass er auch für Performance Manager und Performance Engineers eine wertvolle Bereicherung für ihre Arbeit darstellt.

1 Technische Aspekte

In Abbildung 1 ist der allgemeine Aufbau eines Performancetests dargestellt. Rechts befindet sich das *Testsystem*, das von den *Lasttreibern* halb links bzw. von den darauf simulierten *virtuellen Usern* durch nutzerähnliche Anfragen unter Last gesetzt wird. Dabei führt jeder virtuelle User immer wieder sein eigenes *Testskript* aus, in welchem ein *Testfall*, d.h. eine bestimmte Abfolge von Anfragen, kodiert ist. Während des Testlaufes werden die Antwortzeiten der Anfragen über entsprechende Messpunkte in den Testskripten sowie systemseitige Auslastungsdaten über diverse Monitore permanent gemessen und protokolliert. Sämtliche lastbezogenen Einstellungen des Tests (Zahl der Lasttreiber, Zuteilung der virtuellen User auf die Lasttreiber, Zuteilung der Testskripte auf die virtuellen User etc.) werden durch den links befindlichen *Controller* gesteuert. Nach Beendigung des Testlaufes werden alle Messdaten an zentraler Stelle zusammengeführt und dort ausgewertet.

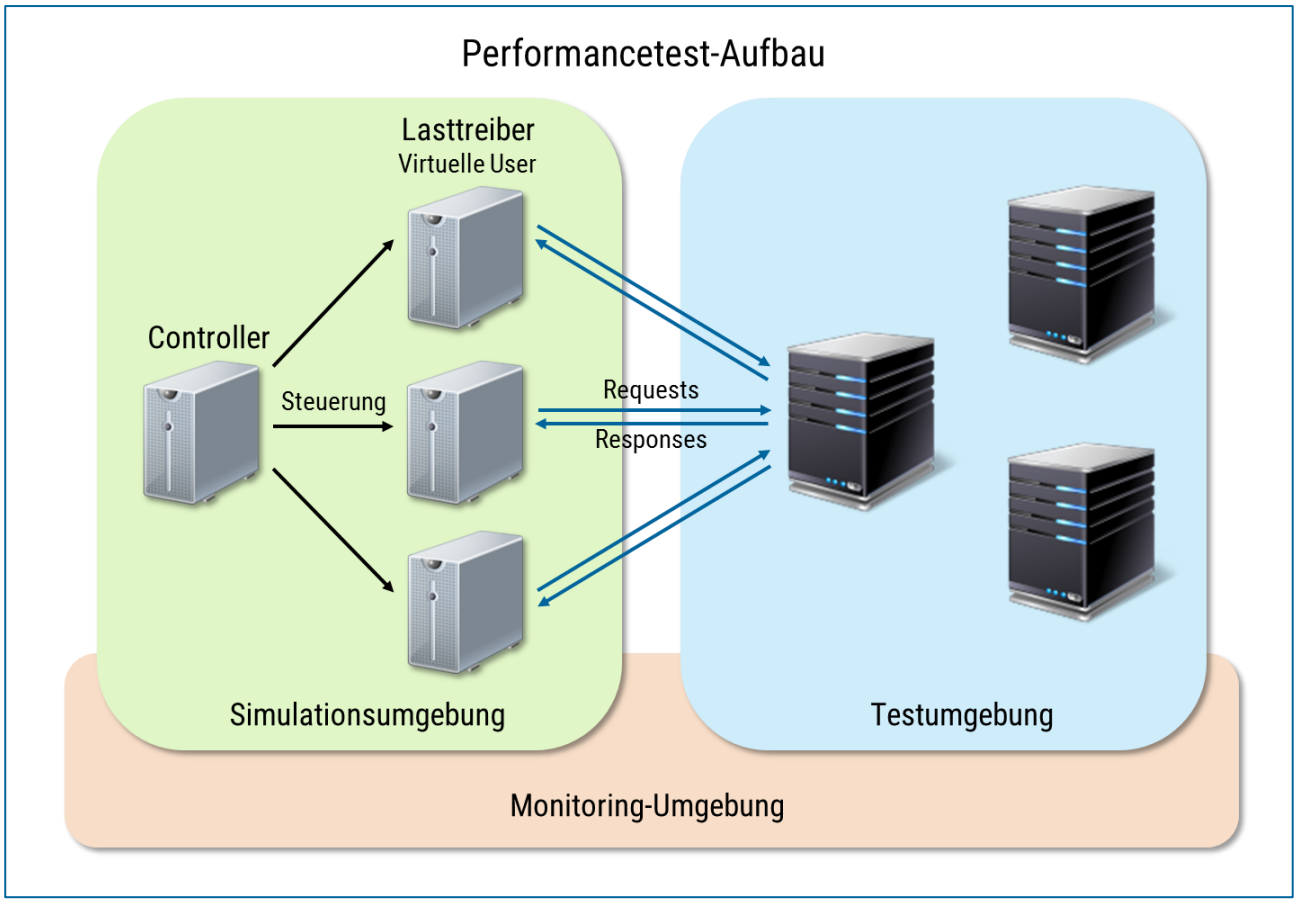

Abbildung 1: Experimenteller Aufbau eines Performancetests. Die Lasttreiber, die durch den Controller gesteuert werden, simulieren virtuelle User. Diese führen skriptbasiert bestimmte Folgen von Requests gegen das zu testende System aus. Unterdessen werden die zugehörigen Request-Response-Zeiten von den virtuellen Usern und anderweitige Systemreaktionen durch entsprechende Monitore gemessen.

Wichtig ist die Feststellung, dass wir uns hier und im folgenden auf *geschlossene Performancetests gegen interaktive Systeme* fokussieren wollen, weil sie allgemein und insbesondere im unternehmerischen Umfeld mit Abstand am häufigsten vorkommen. Die Bedeutung dieser Einschränkung wird aus zwei fundamentalen Unterscheidungen ersichtlich, die nun als erstes diskutiert werden.

Interaktive versus Batch-Systeme

In *interaktiven Systemen* wird jede Anfrage individuell verarbeitet, und es erfolgt auf jede Anfrage eine zugehörige Antwort. Dagegen werden in *Batch-Systemen* einzelne Anfragen zunächst gesammelt und anschließend als großer zusammenhängender Stapel (*Batch*) prozessiert. Demnach werden dort in der Regel allenfalls Rückmeldungen auf die Verarbeitung ganzer Batches zu erwarten sein. Nun wird man bei verteilten Systemen oftmals sowohl interaktive als auch batch-verarbeitende Komponenten antreffen, so dass eine klare Unterscheidung in Bezug auf das Gesamtsystem fraglich erscheint. Entscheidend ist jedoch, ob mit jedem nutzerähnlichen „äußeren Request" ein „äußerer Response" verbunden ist. Ist dies der Fall, so ist das Gesamtsystem interaktiv; andernfalls ist es ein Batch-System.

Eine wichtige Konsequenz von interaktiven und batch-verarbeitenden Systemkomponenten betrifft deren Belastungsfähigkeit. Betrachten wir hierzu zunächst ein System, das ausschließlich aus interaktiven Komponenten besteht. In ihnen wird jeder Job sofort verarbeitet bzw. auf Betriebssystemebene in die Process Queue gestellt. Somit ist zu erwarten, dass in Performancetests die Auslastungen aller Systemkomponenten mit der Zahl der virtuellen User innerhalb gewisser Grenzen wachsen und dadurch einigermaßen kontrolliert werden können. Nehmen wir nun an, das System enthalte wenigstens eine Batch-Komponente. Dann wird diese Komponente während der Batch-Sammelphasen praktisch nicht und während der Batch-Verarbeitungsphasen konstant belastet, unabhängig von der Zahl der virtuellen User. In diesem Fall wird es also eine kompliziertere Abhängigkeit zwischen der Zahl der virtuellen User und den Belastungen einzelner Systemkomponenten geben, zumal sich die Asynchronität der Batch-Verarbeitung auch auf das Verhalten anderer Systemkomponenten auswirkt.

Offene versus geschlossene Performancetests

Je nach Funktionsweise der virtuellen User unterscheidet man zwischen *offenen* und *geschlossenen Performancetests*. Bei offenen Performancetests werden von den virtuellen Usern ständig Requests, beispielsweise mit konstanter Frequenz, abgesetzt, unabhängig davon, ob der vorangegangene Request bereits beantwortet wurde. Dagegen setzen die virtuellen User in geschlossenen Tests den nächsten Request frühestens dann ab, nachdem sie einen Response auf den Vorgänger-Request erhalten haben. Während also bei offenen Tests die Request-Frequenz und damit die Zahl der im Testsystem befindlichen Jobs im Prinzip beliebig groß werden können, ist bei geschlossenen Tests die mittlere Request-Frequenz immer gleich der mittleren systemischen Response- bzw. Verarbeitungsfrequenz. Zudem ist dort die Zahl der Jobs im Testsystem durch die Zahl der virtuellen User beschränkt (siehe Abbildung 2).

Anders ausgedrückt: im Gegensatz zu offenen Performancetests genügen geschlossene Tests notwendigerweise dem Prinzip der *Job Flow Balance*, demzufolge innerhalb eines jeden hinreichend großen Zeitraumes ebenso viele Jobs das Testsystem betreten und verlassen. In der Praxis überträgt sich dieses Prinzip fast immer vom Gesamtsystem auf seine Teilkomponenten, unabhängig von deren konkreten Ausprägungen. Dies hat eine sehr weitreichende Konsequenz: überall dort, wo Job Flow Balance herrscht, gelten die *Operational Laws*. Sie stellen einfache Gesetzmäßigkeiten zwischen verschiedenen Performance-Metriken (beispielsweise Antwortzeiten, Durchsätzen, Auslastungen und Queue-Längen) her, die unter an-

derem dazu genutzt werden können, den fehlerfreien und plangemäßen Verlauf von Performancetests sowie die *qualitative Gesundheit* des zu testenden Systems zu prüfen.

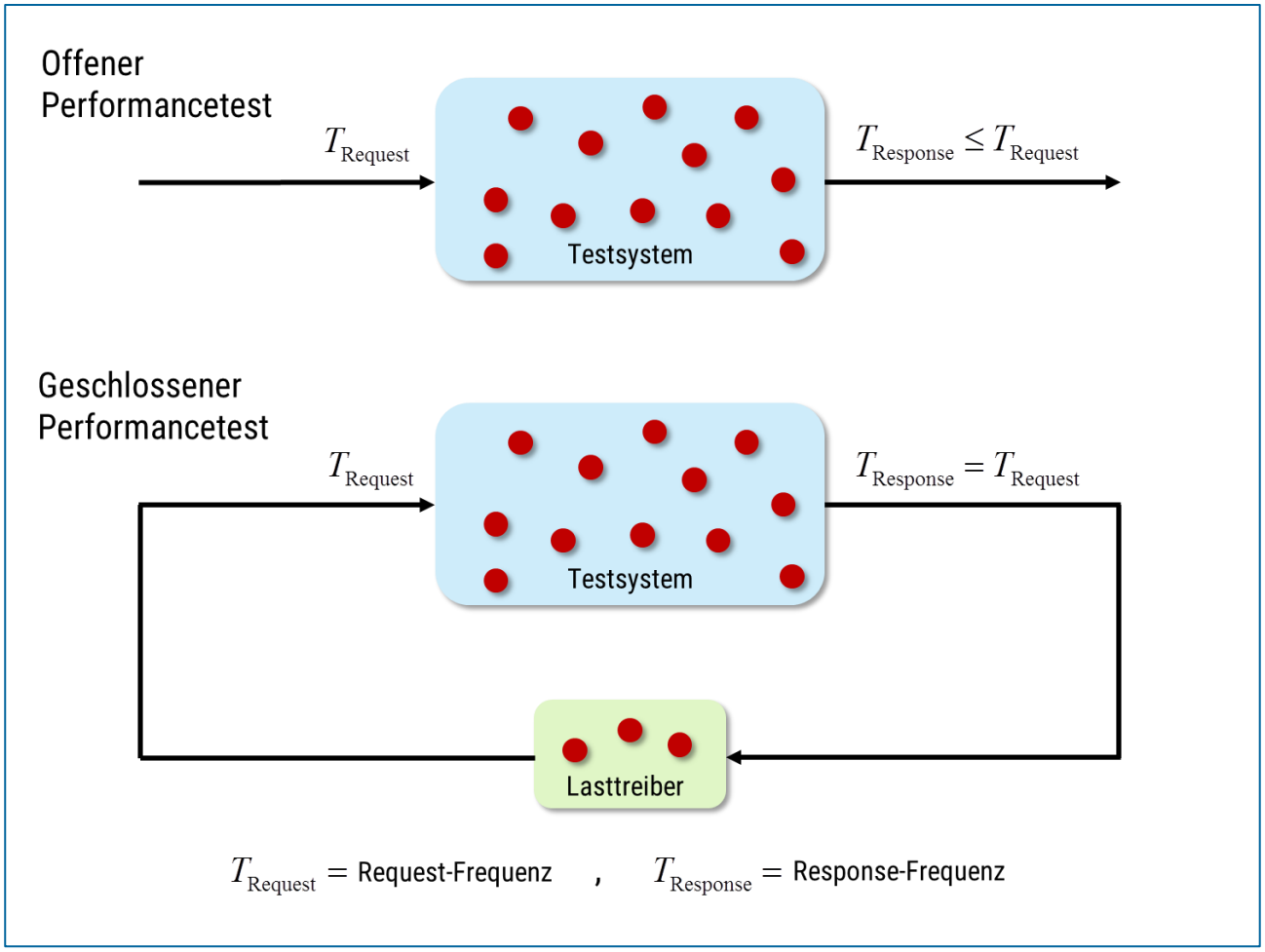

Abbildung 2. *Oben:* In offenen Performancetests ist der maßgebliche lastbezogene Freiheitsgrad die Request-Frequenz, mit der das Testsystem „beschossen" wird. Übersteigt die Request-Frequenz die maximal mögliche Response-Frequenz des Testsystems, so nimmt die Zahl der im Testsystem befindlichen Jobs mit der Zeit zu. *Unten:* In geschlossenen Performancetests wird die Systembelastung vor allem durch die Gesamtzahl der Requests bzw. Jobs gesteuert, die zwischen Testsystem und Lasttreiber zirkulieren. Hier ist die mittlere Request-Frequenz immer gleich der mittleren Response-Frequenz (*Job Flow Balance*). Solange die Zahl der zirkulierenden Jobs bzw. die Zahl der virtuellen User konstant gehalten wird, sind auch die mittleren Jobzahlen im Testsystem und im Lasttreiber konstant.

Testwerkzeuge

Die zur Durchführung von Performancetests notwendigen Werkzeuge lassen sich in die Kategorien *Simulation*, *Monitoring* und *Analyse* unterteilen. Die Simulationswerkzeuge decken den linken Teil der Abbildung 1 ab. Zusätzlich bieten sie Möglichkeiten zur Erstellung und Anpassung der Testskripte. Die Monitoring Tools im unteren Teil der Abbildung 1 übernehmen die Messung der Systemauslastungen während der Testläufe. Dies umfasst neben hardware- und betriebssystemseitigen Metriken auch applikationsspezifische Kenngrößen. Mit Hilfe der Analysewerkzeuge werden schließlich sämtliche Messdaten ausgewertet. Grundsätzlich hängt es vor allem von den Möglichkeiten der Monitoring- und Analysewerkzeuge ab, in welcher Breite und Tiefe Performancetests interpretiert werden können.

Kommerzielle Werkzeuge bestehen in der Regel aus Tool-Komponenten, die alle drei Kategorien abdecken. Diesem All-Inclusive-Vorteil steht allerdings der Nachteil einer gewissen Inflexibilität gegenüber, insbesondere in Bezug auf das Monitoring und die Analyse. So lassen sich beispielsweise externe Monitoring-Daten oft nur schwer in die Standardanalyse integrieren. Zudem wird man möglicherweise auf Analyseverfahren zurückgreifen wollen, die die Standardanalyse nicht hergibt.

Im Freeware-Bereich gibt es mittlerweile einige Werkzeuge, die in puncto Simulation an kommerzielle Produkte heranreichen. Allerdings bieten sie aktuell allenfalls rudimentäre Unterstützung in Richtung Monitoring und Analyse. Hier wird man also in jedem Fall unterschiedliche Produkte für die Simulation, das Monitoring und die Analyse kombinieren müssen.

Simulation auf Protokollebene

Reale Nutzer interagieren mit IT-Systemen über *User Interfaces*, beispielsweise über Web Browsers oder Java Clients. In den Testskripten von Performancetests bleiben diese jedoch meistens unberücksichtigt. Stattdessen umfassen und messen die Skripte lediglich den Teil der Interface-System-Kommunikation, der auf Protokollebene vom Netzwerk übertragen wird. Unter anderem darin unterscheiden sich Performancetests von automatisierten Funktionstests: dort werden Testskripte eingesetzt, um bestimmte Aktionen in real existierenden User Interfaces auszulösen. Es ist der protokollbasierten Natur der Testskripte von Performancetests zu verdanken, dass dort viele virtuelle User auf einem Lasttreiber simuliert werden können.

Testsystem

Der wichtigste und interessanteste Einflussfaktor auf die Ergebnisse von Performancetests ist selbstverständlich das zu testende System selbst. Dabei spielen die leistungsbezogenen Eigenschaften seiner Komponenten auf Hardware- und Software-Ebene eine ebenso große Rolle wie seine topologischen Merkmale. Hierin liegt ein weiterer Unterschied zu Funktionstests: deren Ergebnisse werden im Normalfall nicht von systemischen Details abhängen, sofern die applikationsseitige Verarbeitungslogik davon unberührt bleibt.

Lastprofil

Die zweitwichtigste experimentelle Einflussgröße ist die Last, mit der das Testsystem stimuliert wird. Hierzu bieten Simulationswerkzeuge in der Regel genügend Konfigurationsmöglichkeiten für Lasttreiber, virtuelle User und Testskripte, um beliebige Lastprofile synthetisch zu erzeugen. Die Vorstellung allerdings, dass die simulierte Last ein vom Testsystem entkoppelter Freiheitsgrad ist und als solcher von außen darauf einwirkt, ist nur im Rahmen von offenen Performancetests gerechtfertigt (siehe die Diskussion *Offene versus geschlossene Performancetests*). In den von uns betrachteten geschlossenen Tests sind Lasttreiber und Testsystem vielmehr als ebenbürtige Bestandteile eines Supersystems anzusehen, zwischen denen sämtliche Requests zirkulieren. Insofern herrscht dort eine enge Korrelation, die nur bestimmte Lastkonstellationen zulässt.

Lasttreiber

Jede atomare Verarbeitungseinheit ist in allen Zeitpunkten entweder zu 0% oder zu 100% ausgelastet. Dazwischen liegende Messwerte sind notwendigerweise über mehrere Einheiten und/oder Messzeitpunkte gemittelt. Wenn nun beispielsweise im Rahmen eines Performancetests bei einem Lasttreiber mit einem Prozessor eine mittlere CPU-Auslastung von 50% gemessen wird, dann bedeutet dies, dass der Prozessor in der Hälfte des betreffenden Messzeitraumes (voll) ausgelastet war und in dieser Hälfte eine retardierende Wirkung entfaltete, die letztlich und in ungewollter Weise auch die Testresultate beeinflusst. Ähnliches gilt für Lasttreiber mit mehreren Prozessoren. Oftmals wird gerade dieser Umstand zu wenig berücksichtigt. Die vorherrschende Meinung ist, dass eine mittlere Auslastung der Lasttreiber unterhalb von 100% prinzipiell unkritisch sei.

Metriken und Monitoring

Bei Funktionstests existiert im wesentlichen nur eine Metrik, die überdies binär ist, nämlich: *funktioniert* oder *funktioniert nicht*. Dagegen haben wir es bei Performancetests je nach Umfang des Monitorings mit einer großen Zahl von nichtbinären, zumeist kontinuierlichen Metriken zu tun, deren Messwerte komplexen Abhängigkeiten unterliegen und interpretiert werden müssen. In technischer Hinsicht ist zu berücksichtigen, dass jedes Monitoring in Abhängigkeit von seiner konkreten Umsetzung ebenfalls einen mehr oder weniger großen, unerwünschten Einfluss auf das Systemverhalten und damit auf die Testergebnisse ausübt.

Testlaufzeiten und statistische Analyse

Schauen wir während eines Performancetests zu verschiedenen Zeitpunkten in das Testsystem hinein, so werden wir immer andere Verteilungen von Jobs auf die einzelnen Systemkomponenten finden und somit auch andere Auslastungswerte messen. Dies bedeutet, dass das Testsystem auf mikroskopischer Ebene nicht stationär ist. Immerhin können wir unter gewissen Voraussetzungen – im wesentlichen: konstante Zahl der virtuellen User und keine Aufschaukelungseffekte – erwarten, dass das Testsystem *makroskopisch stationär* ist, dass also bei hinreichend vielen Testskriptdurchläufen bzw. bei einer hinreichend langen Testlaufzeit die Messmittelwerte aller Metriken stabil sind.

Dies ist ein Grund, warum wir in Performancetests die virtuellen User ihre Testskripte nicht nur einmal, sondern über einen längeren Zeitraum sehr oft ausführen lassen. Ein anderer Grund könnte sein, dass wir gerade die makroskopische Stationarität des Testsystems in Frage stellen und deshalb an diversen Aufschaukelungseffekten (Memory Leaks, Session Overflows etc.) interessiert sind, die sich möglicherweise erst nach längerer Testlaufzeit bemerkbar machen. Die Folge ist jedenfalls, dass je nach Test- und Monitoring-Umfang Massendaten im Megabyte- oder Gigabyte-Bereich anfallen, die mit statistischen Mitteln analysiert werden müssen.

Weil IT-Systeme in funktionaler Hinsicht als deterministisch angenommen werden, findet in jedem Funktionstest die Prüfung einer bestimmten Funktionalität nur einmal statt. Dementsprechend ist dort die statistische Analyse von Messdaten kein Thema.

Testdaten

In jedem Testskript ist eine bestimmte Abfolge von Anfragen kodiert. Nun müssen beim Absetzen einer Anfrage fast immer auch Daten mitgesendet werden, auf die das Testsystem nur dann funktional korrekt reagieren kann, wenn es über einen dazu passenden Datenbestand verfügt. Um die funktional korrekten Durchläufe eines Testskriptes zu gewährleisten, werden also in der Regel *Bewegungsdaten* benötigt, die auf Skriptseite verankert sind, sowie entsprechende *Stammdaten*, die im Testsystem vorgehalten werden. Ferner ist zu berücksichtigen, dass Bewegungsdaten unter Umständen nach ihrem erstmaligen Gebrauch verfallen und nicht wiederverwendet werden können.

Mindestens genauso wichtig wie diese funktionalen Abhängigkeiten ist der Umstand, dass die qualitative und quantitative Beschaffenheit der Testdaten einen großen Einfluss auf das Leistungsverhalten des Testsystems haben kann. Hat man es beispielsweise mit einem Online-Banking-System zu tun, an das sich alle virtuellen User über denselben Account anmelden, so wird dieses System möglicherweise völlig anders beansprucht, als wenn jeder virtuelle User einen anderen Account benutzt. Ähnliches gilt für die Füllstände der beteiligten Datenbanken.

Schließlich ist in Bezug auf den Umfang der Testdaten auch zu beachten, dass innerhalb eines Performancetests durchaus Hunderte oder Tausende von virtuellen Usern involviert sein können, die ihren Testfall hundert- oder tausendfach ausführen.

Freiheitsgrade

Ergänzend zu den bisherigen Betrachtungen fassen die folgenden vier Tabellen die experimentellen Freiheitsgrade zusammen, die es bei geschlossenen Performancetests gegen interaktive Systeme typischerweise zu berücksichtigen und festzulegen gilt. Aspekte, die sich normalerweise dem Einfluss des Performance Engineers bei seiner Testgestaltung entziehen, bleiben hierbei unberücksichtigt.

Testumgebung	
Freiheitsgrad	**Bemerkungen**
Art und Zahl der System- und Peripheriekomponenten	Leistungsbezogene Merkmale
Hardware- und softwareseitige Ausstattungen und Konfigurationen	
Virtualisierungsgrade	
Netzwerkseitige Anbindungen	
Testfremde Belastungen	
Schaltungen (seriell/parallel)	Topologische Merkmale
Routing- und Loadbalancing-Strategien	
Abhängigkeiten zu externen Systemen	

Tabelle 1: Experimentelle Freiheitsgrade der Testumgebung.

Simulationsumgebung	
Freiheitsgrad	**Bemerkungen**
Zahl der Lasttreiber	Grundsätzlich ist eine möglichst niedrige Auslastung der Lasttreiber anzustreben.
Leistung der Lasttreiber	
Netzwerkseitige Anbindungen der Lasttreiber	Unterschiedliche Anbindungen machen Sinn, um netzwerkseitige Einflüsse zu studieren.

Tabelle 2: Experimentelle Freiheitsgrade der Simulationsumgebung.

Lastprofil	
Freiheitsgrad	**Bemerkungen**
Zahl der Testfälle/Testskripte	Bestimmen die Komplexität und Heterogenität der generierten Last.
Request-Folgen der Testfälle/Testskripte	
Protokollspezifische Einstellungen in den Testskripten	Kopfdaten, die in Requests mitgesendet werden (z.B. TCP- oder HTTP-Header).
Synchronisationspunkte in den Testskripten	Dort warten virtuelle User aufeinander, um anschließend gleichzeitig fortzufahren.
Think Times in den Testskripten	Wartezeiten zwischen zwei aufeinander folgenden Requests.
Stammdaten des Testsystems	Müssen in ausreichendem Umfang vorhanden und funktional aufeinander abgestimmt sein.
Bewegungsdaten der Testskripte	
Zuordnung: virtuelle User → Lasttreiber	Scheduling: Wo und wann führen wie viele virtuelle User welche Testskripte aus?
Zuordnung: Testskripte → virtuelle User	
Aktivierung und Deaktivierung der virtuellen User während der Testlaufzeit	

Tabelle 3: Experimentelle Freiheitsgrade des Lastprofils.

Monitoring-Umgebung	
Freiheitsgrad	**Bemerkungen**
Betriebssystemseitige Messmetriken für jede Systemkomponente und für die Lasttreiber	Bestimmen die mögliche Qualität der Performancetest-Analyse.
Applikationsseitige Messmetriken für jede Systemkomponente	
Messprozeduren	Pulling, Pushing, Instrumentierung etc.
Messfrequenzen	Hier muss abgewogen werden zwischen der Blindheit gegenüber kurzzeitigen Effekten (kleine Frequenzen) und zusätzlichen Belastungen (große Frequenzen).

Tabelle 4: Experimentelle Freiheitsgrade der Monitoring-Umgebung.

Insgesamt haben wir es also mit experimentellen Freiheitsgraden in der Größenordnung von einigen Dutzend zu tun. Sie alle, und im Einzelfall noch weitere, üben Einfluss auf die Ergebnisse von Performancetests aus.

2 Methodische Aspekte

Notwendige Voraussetzung für die Konzeption von sinnvollen Performancetests ist, dass eine fundierte Vorstellung davon existiert, welche konkreten experimentellen Setups sich zur Beantwortung der jeweils gestellten Fragen eignen, welche Messgrößen relevant sind und wie sich die Antworten aus den Messwerten extrahieren lassen. Von den unzähligen, in diesem Zusammenhang zu berücksichtigenden methodischen Aspekten wollen wir im folgenden einige herausgreifen, die wir einerseits als fundamental erachten und andererseits gerade im unternehmerischen Umfeld nicht immer ausreichend gewürdigt sehen.

Fragen und Ziele

Typische performancetest-relevante Fragestellungen, die während der Entwicklung von IT-Systemen (*konstruktive Phase*) oder in der Diagnostik bestehender Systeme (*analytische Phase*) aufgeworfen werden, lauten:

Was sind die wesentlichen performance-beeinflussenden und performance-beschreibenden Größen des IT-Systems?

In dieser sehr grundsätzlichen Frage geht es darum, wie sensibel das System auf die Änderung bestimmter Rahmenbedingungen reagiert und in welchen Metriken dies zum Ausdruck kommt. Derartige Betrachtungen sind nützlich, um die Performance-Zusammenhänge des Systems besser zu verstehen. Zudem helfen sie, die Ziele von zukünftigen Performancetests exakter zu formulieren, deren Setups zielgerichteter zu gestalten und die Zahl der relevanten experimentellen Freiheitsgrade effektiv einzugrenzen.

In welcher leistungsmäßigen Beziehung stehen verschiedene Systeme zueinander?

Systemvergleiche spielen zum Beispiel eine Rolle, wenn es um die strategische Frage geht, welche Plattform in welcher Dimensionierung sich am besten zur Bewältigung der anfallenden Aufgaben eignet. Denkbar sind auch sehr spezielle Detailvergleiche auf Komponentenebene, von denen insbesondere die Systementwicklung und der Systembetrieb profitieren können. Im Rahmen von Updates oder Upgrades ist schließlich der damit insgesamt verbundene Performance Impact von Interesse.

Wie verhält sich das System unter extremen Belastungskonstellationen?

Systemarchitekten müssen bei der Konzeption von IT-Plattformen immer auch mögliche Änderungen des Nutzerverhaltens im Blick haben. Diese können schleichend verlaufen, etwa aufgrund eines stetig wachsenden Geschäftsvolumens, oder kurzfristig und heftig eintreten, wenn beispielsweise für einen Online-Shop plötzlich Werbespots im Fernsehen geschaltet werden.

Was sind die Ursachen der Performance-Defizite?

Diese Frage taucht typischerweise immer dann auf, wenn sich ein System im Produktionsbetrieb als zu leistungsschwach („zu große Antwortzeiten“) oder instabil („zu viele Ausfälle“) herausstellt. In solchen Fällen besteht der Wunsch, diese, oftmals subjektiv empfundenen Mängel zu objektivieren, deren Ursachen zu identifizieren und durch entsprechende Tuning-Maßnahmen zu beseitigen.

Genügt das System den Performance Requirements?

Performance Requirements stellen Grenzen für Messwerte dar, die unter bestimmten Rahmenbedingungen nicht überschritten werden sollten. Ihr Erfüllungsgrad entscheidet in der Regel über die Abnahme von Software-Produkten und IT-Systemen.

Erwartungshaltungen

Performancetests verlangen eine konkrete und plausible Haltung bezüglich der zu erwartenden Ergebnisse. Im Falle eines Abnahmetests sollte sich diese vor allem in den Performance Requirements widerspiegeln, dergestalt, dass ihre Rahmenbedingungen, Messgrößen und Grenzwerte vollständig und widerspruchsfrei definiert und auf den Test abgestimmt sind. Bei Tests, in denen etwa das Systemverhalten unter variierenden Lastverhältnissen im Vordergrund steht, können die resultierenden Kurvenverläufe ohne eine dezidierte Erwartungshaltung nicht kritisch gewürdigt, sondern lediglich wertfrei zur Kenntnis genommen werden.

Ohne Übertreibung lässt sich sagen, dass die Generierung von performancetest-bezogenen Erwartungshaltungen die größte methodische Herausforderung darstellt. Im allgemeinen wird man hierzu einen umfangreichen experimentellen Erfahrungsschatz aufbauen müssen. Ebenso wichtig ist die theoretische Beschäftigung mit dem Thema *Performance-Modellierung von IT-Systemen*, wie es unter anderem im Rahmen der *Warteschlangentheorie* behandelt wird.

Bei funktionalen Tests ist dieser Punkt vergleichsweise einfach: dort genügt normalerweise ein Blick ins Pflichtenheft, um eine eindeutige Erwartungshaltung bezüglich der zu testenden Funktionalität zu entwickeln.

Testsystem versus Produktionssystem

In der Regel wird das Testsystem schon allein aus finanziellen oder logistischen Gründen nicht in allen Belangen dem Produktionssystem gleichen, für dessen Performance-Eigenschaften man sich eigentlich interessiert. Zudem kann es in Abhängigkeit von der jeweiligen Fragestellung deutlich effektiver sein, eine anders geartete oder kleiner dimensionierte Testumgebung zu bevorzugen, in der die relevanten Phänomene prägnanter und mit kleinerem experimentellen Aufwand (weniger virtuelle User, weniger Testdaten, kürzere Testlaufzeiten) zu Tage treten. In all diesen Fällen wird man sich genau überlegen müssen, welche Testumgebung für welche Fragestellung geeignet ist und inwiefern die auf der Testumgebung gewonnenen Erkenntnisse auf die produktive Situation extrapoliert werden können. Auch hierfür sind umfangreiche experimentelle Erfahrungen und theoretische Modellbetrachtungen unentbehrlich.

Realitätsnähe versus Interpretierbarkeit

Relativ unabhängig von den zugrundeliegenden Fragestellungen ist die konkrete Ausgestaltung von Performancetests häufig von dem prinzipiellen Wunsch nach maximaler Realitätsnähe geprägt, um die Testergebnisse möglichst direkt und unvermittelt auf die produktiven Verhältnisse übertragen zu können. Dies betrifft neben den eben besprochenen systemischen Verhältnissen auch die Systembelastung. So werden oftmals höchst komplexe und heterogene Lastprofile in Form von umfangreichen Testfällen und Testfallkombinationen konzipiert, von denen man annimmt, dass sie das reale Nutzerverhalten einigermaßen realistisch widerspiegeln.

Hierzu und auch zum vorherigen Diskussionspunkt ist grundsätzlich festzustellen, dass Komplexität immer in einem Spannungsverhältnis zu der Kardinalforderung nach *Interpretierbarkeit* steht: je komplexer das experimentelle Setup eines Performancetests ist, umso weniger wird man in der Lage sein, dessen Ergebnisse zu verstehen und mit ihren konkreten Ursachen in Verbindung zu bringen. Der experimentelle Aufbau von Performancetests sollte daher immer möglichst einfach gehalten werden und sich vor allem an den eigentlichen Testzielen orientieren.

Realitätsnähe und deren Nachweis

Es mag Performancetests geben, bei denen die Realitätsnähe trotz aller Vorbehalte ein legitimes Gestaltungskriterium ist. Dann stellt sich aber die Frage, wie realitätsnahe Lastkonstellationen simuliert werden können. Ignorieren wir die sehr grundsätzliche Tatsache, dass das reale Nutzerverhalten meistens eine Mischung aus initialen Aktionen und feedbackgebundenen Interaktionen ist, von denen allenfalls letztere im Rahmen von geschlossenen Performancetests adäquat berücksichtigt werden können, so sind prinzipiell zwei Herangehensweisen vorstellbar:

Nutzerbezogener Ansatz

Der Grundgedanke hierbei ist, jeden virtuellen User seinem Namen entsprechend als Stellvertreter eines realen Nutzers agieren zu lassen. Die lastprofilbezogenen experimentellen Freiheitsgrade (Zahl der virtuellen User, Art und Zahl der Testskripte, Think Times etc.) werden dabei auf der Basis von Abschätzungen in Bezug auf diverse durchschnittliche Verhaltensweisen der realen Nutzer festgelegt. Dieses, nicht selten zu beobachtende Vorgehen ist aus zwei Gründen ausdrücklich nicht zu empfehlen: zum einen werden die Schätzungen in den allermeisten Fällen nur unzureichend legitimiert werden können. Zum anderen erfordert es einen enorm hohen experimentellen Aufwand, verbunden mit vielen virtuellen Usern, vielen Testskripten, langen Testlaufzeiten und deshalb auch mit einem hohen Risiko von Testinstabilitäten.

Systembezogener Ansatz

Bei dieser Vorgehensweise werden die virtuellen User als abstrakte Lasterzeugungsinstanzen ohne jeglichen Bezug zu realen Nutzern betrachtet. Ihre Aufgabe besteht allein darin, die reale Last, wie sie vom Produktionssystem „erfahren“ wird und anhand von Logfiles ermittelt werden kann, nachzustellen. Die Festlegung der zugehörigen experimentellen Freiheitsgrade erfolgt hier in iterativer Weise über *Kalibrierungstests*, bei denen die testinduzierten mit den

produktionsbetrieblichen Logfiles hinsichtlich der relevanten Parameter verglichen werden. Weil diese Strategie nicht nur den nachvollziehbaren Beleg der Realitätsnähe automatisch in sich trägt, sondern auch zu einfacheren experimentellen Setups führt (weniger virtuelle User, weniger Testskripte, kürzere Testlaufzeiten, größere Teststabilität), ist sie gegenüber dem nutzerbezogenen Ansatz grundsätzlich zu bevorzugen.

Think Times

Der Hang zum nutzerbezogenen Ansatz zeigt sich besonders deutlich in den Wartezeiten, die man innerhalb von Testskripten einbaut, um das reale Nutzerverhalten in Bezug auf Interaktionspausen zu simulieren. Wir wollen deshalb einmal die damit verbundenen Konsequenzen anhand eines konkreten Beispiels veranschaulichen. Dazu betrachte man ein makroskopisch stationäres System, das von 610 virtuellen Usern unter Verwendung eines einzigen Testfalles/Testskriptes und Think Times mit einer Gesamtlänge von 60 Sekunden unter Last gesetzt wird (nutzerbezogener Ansatz). Jeder virtuelle User messe eine mittlere Nettotestfallzeit von 1 Sekunde und dementsprechend eine mittlere Bruttotestfallzeit von 61 Sekunden.

Würde man im Testskript sämtliche Think Times eliminieren, so lässt sich errechnen, dass man dann lediglich 10 virtuelle User bräuchte, um praktisch dieselbe mittlere Nettotestfallzeit und damit dieselbe Systembelastung zu provozieren (systembezogener Ansatz). Hinzu kommt, dass dann jeder virtuelle User innerhalb eines gegebenen Zeitraumes 61 mal so viele Testskriptdurchläufe registrieren würde wie die virtuellen User zuvor, so dass eine statistisch hinreichend große Durchlaufzahl schon sehr viel früher erreicht wäre. Insgesamt würde also der Wegfall aller Think Times zu einer enormen Reduktion des experimentellen Aufwandes führen:

Zahl der virtuellen User: 610 → 10, Testlaufzeit: einige Stunden → einige Minuten.

Die Reduktion würde sogar noch drastischer ausfallen, wenn die ursprünglichen Think Times größer oder das System in Bezug auf den simulierten Testfall performanter gewählt worden wären.

Nun wurde in dieser Argumentation eine wesentliche Voraussetzung unterschlagen, nämlich, dass das Systemverhalten durch etwaige persistente User-System-Verbindungen nicht beeinflusst wird. Sollte eine derartige Beeinflussung gegeben sein, so wird man um der Realitätsnähe willen auf die Simulation von vielen virtuellen Usern nicht verzichten können. Aber auch dann ist es von Vorteil, die Zahl der virtuellen User und die Think Times aus der systembezogenen Perspektive heraus geschickt zu wählen.

Letztlich sollte es nur einen legitimen Grund für die Verwendung von Think Times geben, um nämlich bei einer systemisch begründeten großen Zahl an virtuellen Usern die Systembelastung kontrollieren und variieren zu können.

Umrechnung von Userzahlen

Ein weiteres, häufig geäußertes Argument für den Einbau von Think Times lautet, dass insbesondere der Auftraggeber mit dem systembezogenen Ansatz nichts anzufangen weiß und deshalb auf eine direkte Entsprechung von virtuellen und realen Nutzern besteht. Hierzu und auch zu allen anderen Einlassungen in diese Richtung sei folgendes gesagt: sofern persis-

tente Verbindungseffekte vernachlässigt werden können, ist es immer möglich, aus den Ergebnissen eines Performancetests, in welchem die Wartezeit-Userzahlkombination {W,N} zur Ausführung eines bestimmten Testskriptes verwendet wurde, auf andere Kombinationen {W‘,N‘} hochzurechnen, die mit hoher Genauigkeit zur selben Systembelastung führen würden.

Diese Möglichkeit der analytischen Umrechnung (*Approximate User Mapping)*, die wir übrigens im obigen Beispiel genutzt haben, untermauert noch einmal die letzte Aussage des vorigen Diskussionspunktes.

Konsistenzprüfung

Unabhängig von der Frage, inwiefern ein Performancetest zielführend konzipiert wurde, macht seine Interpretation allenfalls dann Sinn, wenn er plangemäß verlief und konsistente Ergebnisse produzierte. Typische Punkte, die es in diesem Zusammenhang zu klären gilt, sind:

- Haben alle Lasttreiber plangemäß am Performancetest teilgenommen?
- Sind alle virtuellen User plangemäß auf die einzelnen Lasttreiber verteilt worden?
- Wurden alle virtuellen User plangemäß zeitlich aktiviert und deaktiviert?
- Haben alle virtuellen User plangemäß ihre Testskripte ausgeführt?
- Gab es funktionale oder technische Fehler in der Ausführung der Testskripte?
- Wurden sämtliche Messdaten korrekt gemessen?

In dieser Hinsicht ist manchmal eine gewisse Nachlässigkeit bzw. „Tool-Gläubigkeit“ zu beobachten, indem davon ausgegangen wird, dass die verwendeten Werkzeuge einen nicht plangemäßen Testverlauf in jedem Fall anzeigen würden. Hier ist dringend anzuraten, sich nicht darauf zu verlassen, sondern vor allem die Testresultate diesbezüglich kritisch zu hinterfragen. So lassen sich beispielsweise anhand der gemessenen Testfallfrequenzen und Testfallzeiten in Verbindung mit den *Operational Laws* die Fragen klären, ob zu jedem Testzeitpunkt die geplante Zahl von virtuellen Usern aktiv war und ob die eingestellten Think Times in den Testskripten korrekt simuliert wurden.

Genauigkeit und Reproduzierbarkeit

Zum Schluss dieses Kapitels wollen wir noch auf einen statistisch-analytischen Gesichtspunkt aufmerksam machen, der hauptsächlich die Forderung der *Reproduzierbarkeit* betrifft.

Auf der einen Seite ist klar, dass die Ergebnisse eines Performancetests wertlos sind, wenn nicht abgesehen werden kann, welche Resultate sich bei abermaliger und identischer Durchführung des Tests ergäben. Andererseits ist ebenso nachvollziehbar, dass zwei Tests nie exakt gleiche Resultate produzieren werden, selbst dann nicht, wenn sie in identischer Weise durchgeführt wurden (siehe die Ausführungen zu *Testlaufzeiten und statistische Analyse*). Dies gilt sowohl für die Einzelmessungen als auch für die zugehörigen statistisch aggregierten Größen.

Nun gibt es glücklicherweise den *Zentralen Grenzwertsatz der Statistik*. Ihm zufolge kann jedem Mittelwert, der sich aus vielen gleichartigen Messwerten einer Messreihe zusammensetzt, ein *Konfidenzintervall* zugeordnet werden, das jenen Bereich umfasst, innerhalb dessen

der Mittelwert aus einer identisch durchgeführten (fiktiven) Messreihe mit einer bestimmten Wahrscheinlichkeit zu erwarten wäre. Zudem besitzt das Konfidenzintervall die wünschenswerte Eigenschaft, dass seine Länge mit zunehmender Zahl der Messwerte kleiner wird und deshalb über diese Zahl gesteuert werden kann. Somit trägt es der intuitiven Erwartung Rechnung, dass ein Mittelwert umso vertrauenswürdiger ist, je mehr Messwerte er umfasst. Dies trifft auf die Standardabweichung nicht zu, weil sie im wesentlichen die Streubreite der Messwerte um ihren Mittelwert beschreibt, und diese Breite ist unabhängig von der Zahl der Messwerte. Insgesamt steht uns also mit dem Konfidenzintervall ein stochastisches Genauigkeits- bzw. Reproduzierbarkeitsmaß zur Verfügung, mit dem wir das oben beschriebene Dilemma auflösen können.

Gerade im Umfeld von Performancetests sind diese Zusammenhänge nicht nur von akademischem Interesse. Vielmehr liefern sie eine entscheidende Rechtfertigungsgrundlage für die Relevanz und somit für die Interpretationswürdigkeit aller Testergebnisse. Insbesondere schützen Sie davor, aus statistisch insignifikanten und daher irrelevanten Resultaten die falschen Schlüsse zu ziehen, zum Beispiel wenn es bei Vergleichstests um die Frage geht, ob das neue Software Release performanter ist als das alte (siehe Abbildung 3).

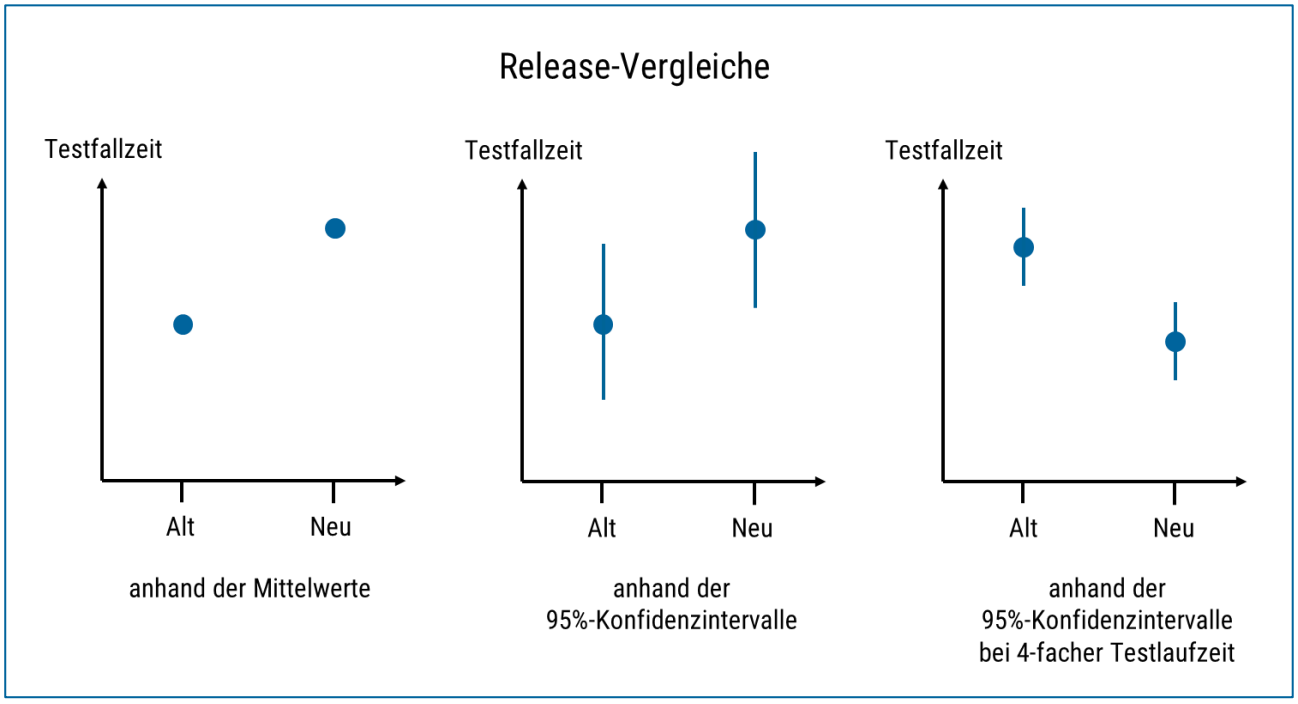

Abbildung 3: Man stelle sich vor, dass mehrere gleichartige Performancetests mit konstanter virtueller Userzahl und einem einzigen Testfall gegen Release *Alt* und Release *Neu* durchgeführt wurden, um die damit verbundenen Performance-Unterschiede zu bestimmen. *Links:* Der Vergleich der Mittelwerte der Testfallzeiten ist wertlos, weil daraus nicht erkennbar ist, wo die Mittelwerte bei erneuter Durchführung der Tests lägen. *Mitte:* Bei Hinzunahme der 95%-Konfidenzintervalle sehen wir, dass die Tests etwaige Performance-Unterschiede nicht auflösen können, weil sich die Intervalle in beträchtlicher Weise überdecken. *Rechts:* Bei erneuter Durchführung der Tests mit vierfacher Testlaufzeit halbieren sich die Konfidenzintervalle. Diese liegen nun überschneidungsfrei auseinander, so dass ein signifikanter Performance-Unterschied konstatiert werden kann.

Leider ist festzustellen, dass dem Aspekt der Genauigkeit von Performancetests im allgemeinen zu wenig Aufmerksamkeit geschenkt wird. Hiermit konfrontiert, verlegt man sich meistens

auf qualitative, gefühlsmäßige Betrachtungen, oder man argumentiert fälschlicherweise über die Standardabweichung. Diese Haltung wird auch durch die Tatsache gefördert, dass die wenigsten Analysewerkzeuge Konfidenzintervalle in die statistische Datenaggregation einbeziehen.

3 Logistische Aspekte

Nachdem wir bislang Performancetests hauptsächlich vor dem Hintergrund der drei Kardinalforderungen *Reproduzierbarkeit*, *Interpretierbarkeit* und *Zielgerichtetheit* diskutiert haben, wollen wir uns in diesem und im nächsten Kapitel vor allem auf praktische Gesichtspunkte bezüglich der *Umsetzbarkeit* von Performancetests im arbeitsteiligen unternehmerischen Umfeld konzentrieren. Ausgangspunkt unserer logistischen Betrachtungen in diesem Kapitel ist die Frage, welche Leistungen von welchen Parteien für die Planung, Durchführung, Analyse und Interpretation von Performancetests typischerweise erbracht werden müssen, und welche Probleme damit einhergehen.

Beteiligte Parteien

In der Regel lassen sich vier Parteien identifizieren, die in Performancetests hauptsächlich involviert sind. Der Einfachheit halber sollen diese im weiteren Verlauf als unternehmensinterne Einheiten vorausgesetzt werden:

Fachbereich

Jeder Fachbereich verantwortet einen bestimmten unternehmerischen Geschäftszweig und hat unter anderem dafür Sorge zu tragen, dass die zugehörigen Geschäftsziele durch den Einsatz von IT-Systemen optimal unterstützt werden. In diesem Zusammenhang tritt er mindestens dreimal als Kunde gegenüber anderen Parteien auf, nämlich (i) bei der Entwicklung von Systemen, (ii) bei deren Qualitätssicherung und (iii) bei deren Betrieb.

Testteam

Der Arbeitsschwerpunkt des Performancetestteams (im folgenden: Testteam) liegt in der Durchführung von Performancetests, mit dem übergeordneten Ziel, die Produktionstauglichkeit von IT-Systemen in Bezug auf Leistung und Stabilität sicherzustellen. Dabei wird es hauptsächlich als qualitätsprüfende Instanz von den Fachbereichen beauftragt. Als weitere Kunden kommen die Systementwicklung und der Systembetrieb in Betracht, zum Beispiel wenn es um spezielle architektonische, programmatische oder konfigurative Performance-Fragestellungen geht.

Systementwicklung

Die Systementwicklung entwickelt neue und aktualisiert bestehende IT-Systeme im Auftrag der Fachbereiche. Dies umfasst die programmatische Umsetzung der Fachbereichsvorgaben in Form von Applikationen sowie die Konzeption von geeigneten Systemarchitekturen für deren Betrieb. Systementwicklung und Testteam kommen typischerweise spätestens dann in Berührung, wenn am Ende einer Entwicklungsphase die qualitätsprüfenden und abnahmerelevanten Performancetests anstehen.

Systembetrieb

Der Systembetrieb verwaltet sämtliche IT-Systeme auf technischer Ebene und ist für deren ordnungsgemäßen und störungsfreien Betrieb verantwortlich. Neben den Produktionssystemen gehören hierzu alle sonstigen Umgebungen, die für nichtproduktive Zwecke wie Forschung und Entwicklung, Qualitätssicherung, Schulungen etc. benötigt werden. In dem von uns betrachteten Kontext besteht eine wichtige Aufgabe des Systembetriebes darin, dem Testteam geeignete Umgebungen für seine Performancetests zur Verfügung zu stellen.

Damit nun das Testteam ein *reproduzierbares*, *interpretierbares* und *zielführendes* Performancetest-Projekt durchführen kann, ist es in nahezu allen Projektphasen auf die Mitarbeit der übrigen Parteien in Form von Informations- und Bereitstellungsleistungen angewiesen (siehe Abbildung 4).

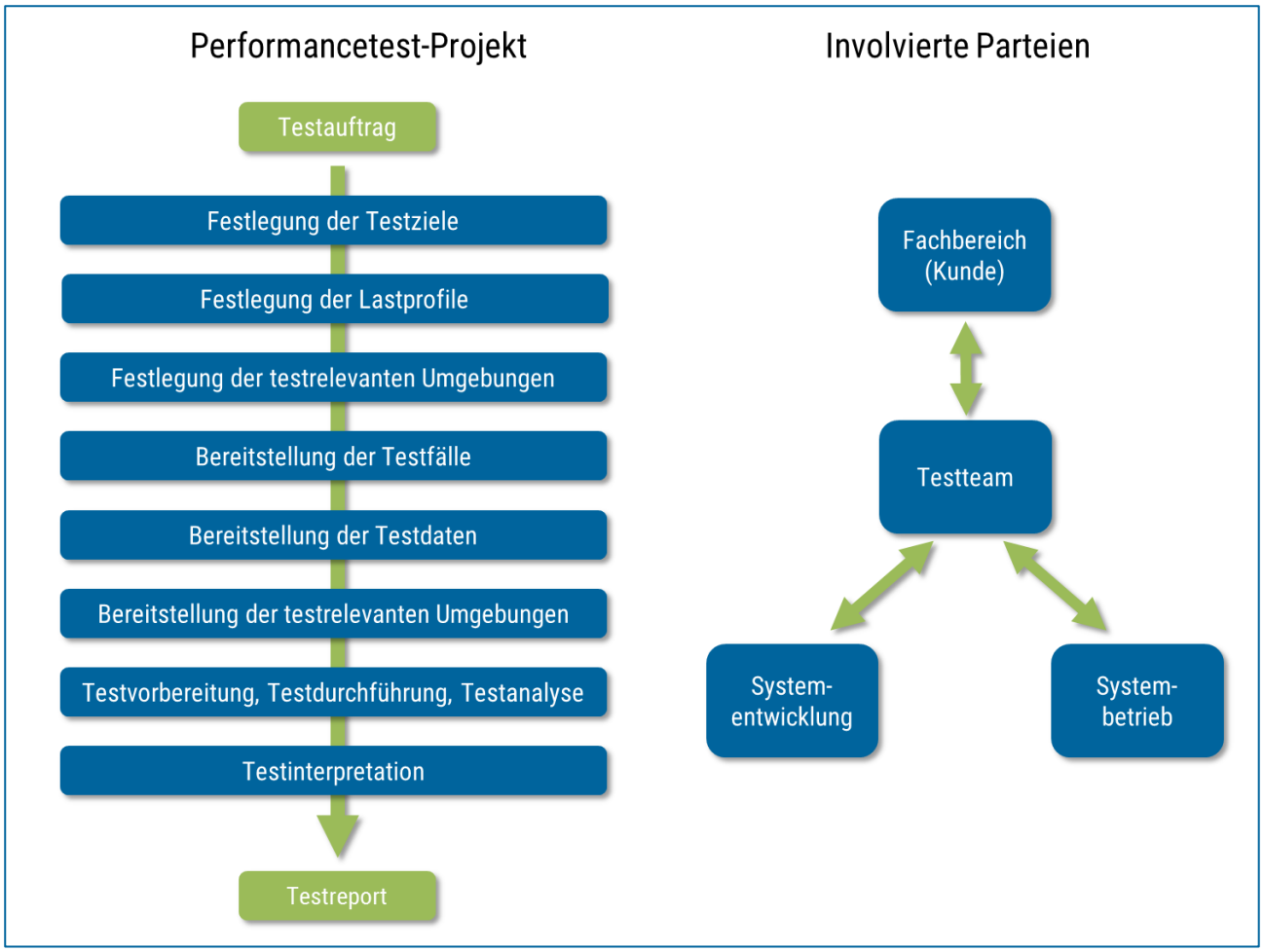

Abbildung 4: *Links:* Idealisierter Verlauf eines typischen Performancetest-Projektes. Nicht berücksichtigt sind zeitliche Überschneidungen und zyklische Abhängigkeiten aufgrund von Fehlerkorrekturen, Kalibrierungstests, Regressionstests und ähnlichem. *Rechts:* In fast allen Projektschritten ist eine Zusammenarbeit des Testteams mit dem Fachbereich, der Systementwicklung und/oder des Systembetriebes erforderlich.

Festlegung der Testziele

Am Anfang des Projektes steht die Beauftragung durch den Kunden, verbunden mit bestimmten Zielsetzungen. Diese sind in der Regel umso abstrakter formuliert, je weniger der Kunde mit den technischen Details des zu testenden Systems befasst ist. Somit obliegt es dem Testteam, die Kundenziele in geeignete operationale Testziele umzuformulieren. Je gründlicher, plausibler und weitsichtiger dieser Transfer vollzogen und kommuniziert wird, umso leichter wird es das Testteam später haben, notwendige Leistungen zu qualifizieren und von anderen Parteien einzuholen.

Basierend auf den Testzielen müssen als nächstes sämtliche experimentellen Freiheitsgrade der anstehenden Performancetests definiert werden.

Festlegung der Lastprofile

Hier ist das Testteam auf die intensive Zuarbeit aller übrigen Parteien angewiesen. So werden unter anderem und unabhängig von den Testzielen fast immer folgende Informationen benötigt:

- Leistungsbezogene, topologische und funktionale Eigenschaften des Produktionssystems; Job-Propagation durch das System; verarbeitungsrelevante Besonderheiten
- Eigenschaften der produktiven Belastungskonstellation und der in diesem Zusammenhang beobachteten Auffälligkeiten
- Eigenschaften der produktiven Stamm- und Bewegungsdaten sowie ihre (vermuteten) Einflüsse auf das Systemverhalten
- Business-relevante Nutzer-System-Interaktionen

Auf dieser Grundlage sollten die in den Performancetests zu simulierenden Lastprofile im Einklang mit den Testzielen von allen Parteien gemeinsam diskutiert und festgelegt werden. Wichtig ist, dass in diesem schwierigen Prozess das Testteam die geistige Führerschaft behält und auf die anderen Parteien lenkend einwirkt. So neigen beispielsweise Fachbereiche gerne dazu, Testfälle und Testfallkombinationen mit dem Fokus auf Realitätsnähe aus der nutzerbezogenen Perspektive heraus zu definieren, ungeachtet der Probleme, die sich daraus für das Testteam bezüglich des experimentellen Aufwandes und der drei Kardinalforderungen ergeben.

Festlegung der testrelevanten Umgebungen

Bei der Definition der Test-, Simulations- und Monitoring-Umgebungen wird das Testteam hauptsächlich auf die Systementwicklung und den Systembetrieb zugreifen wollen, um mit ihnen Fragen der folgenden Art zu erörtern:

- Welche Testumgebungen eignen sich am besten zur Verfolgung der Testziele?
- Wie können testfremde Systembelastungen und Anbindungen zu externen Systemen unterdrückt werden?
- Wie muss die Simulationsumgebung gestaltet sein, um Messartefakte weitestgehend auszuschließen?

- Welche speziellen Metriken und Messprozeduren müssen ins Monitoring einbezogen werden?

Auch hier sollte das Testteam als Leuchtturm fungieren, indem es immer wieder die drei Kardinalforderungen in den Mittelpunkt der Diskussionen rückt. Folgende Beispiele mögen dies verdeutlichen:

Wiederverwendbarkeit (→ Reproduzierbarkeit)
Für vergleichende Performancetests, zum Beispiel im Rahmen von Release-Wechseln, ist unbedingt darauf zu achten, dass die betreffenden Umgebungen in allen ursprünglich definierten Ausstattungsmerkmalen immer wieder verwendet bzw. neu bereitgestellt werden können.

Virtualisierung (→ Interpretierbarkeit)
Bei Performancetests, in denen etwa das Auffinden von Bottlenecks oder die Suche nach dem optimalen Sizing im Vordergrund stehen, ist die leistungsmäßige Konstanz aller Testsystemkomponenten eine wünschenswerte Voraussetzung. In solchen Fällen ist es daher sinnvoll, virtuelle Testumgebungen weitestgehend zu „entvirtualisieren", indem sämtlichen Komponenten fixe Ressourcen zugeteilt werden.

Anbindung von externen Systemen (→ Zielgerichtetheit)
Ein typischer Reflex zur Eliminierung von Abhängigkeiten zu externen Systemen besteht darin, diese durch aufwändige *Mock-Systeme* zu simulieren. Sofern es hierbei lediglich um die Unterdrückung von externen retardierenden Effekten geht, lässt sich dies möglicherweise ebenso gut aber mit deutlich weniger Aufwand erreichen, indem man die Testumgebung sehr klein dimensioniert, so dass die externen Systeme vergleichsweise transparent werden.

Nach der Festlegung der experimentellen Setups folgt in den nächsten Schritten deren konkrete Umsetzung.

Bereitstellung der Testfälle

In der Regel ist das Testteam mit den fachlichen Abhängigkeiten der vereinbarten Testfälle nicht hinreichend vertraut. Um die zugehörigen Testskripte erstellen zu können, benötigt es daher detaillierte Testfallbeschreibungen vom Fachbereich oder von der Systementwicklung. Diese sollten idealerweise in Form von selbstkonsistenten „Drehbüchern" inklusive Screenshots von Eingabemasken und Eingabeaktionen abgefasst sein.

Bereitstellung der Testdaten

Die für die Performancetests erforderlichen großen, variablen und funktional stimmigen Testdatenbestände müssen entweder aus Produktionsdatenbanken in anonymisierter Weise extrahiert oder synthetisch erzeugt werden. Je nach Datenhoheit ist hierfür meistens der Fachbereich oder die Systementwicklung verantwortlich. Leider kommt es aufgrund des damit verbundenen hohen logistischen Aufwandes häufig vor, dass die Testdaten deutliche Qualitätsmängel aufweisen, die sich zudem erst nach zeitraubenden Prüfungen durch das Testteam offenbaren. Um in solchen Fällen weitere Verzögerungen zu vermeiden, bleibt dem Testteam

nichts anderes übrig, als eigene Workarounds zu entwickeln oder sich mit der minderen Testdatenqualität abzufinden.

Bereitstellung der testrelevanten Umgebungen

Die testrelevanten Umgebungen werden vom Systembetrieb bereitgestellt, gegebenenfalls unter Mitwirkung der Systementwicklung, wenn es etwa um spezielle Software-Instrumentierungen für das Monitoring geht. Dabei sieht sich das Testteam nicht selten mit folgenden Problemen konfrontiert:

Sicherheit

In vielen Unternehmen sind die Sicherheitsstandards so hoch, dass das Testteam faktisch keinen Zugriff auf die testrelevanten Umgebungen hat und daher auch bei trivialen administrativen Tätigkeiten, etwa dem Starten oder Stoppen von Monitoren, auf die ständige Zuarbeit des Systembetriebes angewiesen ist.

Transparenz

Weil die testrelevanten Umgebungen zudem häufig schlecht dokumentiert sind, fällt es dem Testteam schwer, deren Ausstattungsmerkmale im Hinblick auf die getroffenen Vereinbarungen zu validieren, so dass diesbezügliche Fehler oder Missverständnisse unentdeckt bleiben.

Initialisierung

Zur Durchführung von Performancetests unter maximal kontrollierten Bedingungen fehlt es oft an geeigneten Prozeduren, um die Testumgebung und insbesondere alle ihre Datenbanken schnell und unkompliziert in ihren jeweiligen initialen Zustand zurückzuversetzen.

Testvorbereitung, Testdurchführung und Testanalyse

Mit den genannten Leistungen versorgt, sollte das Testteam nun in der Lage sein, die Performancetests final vorzubereiten, durchzuführen und anschließend auszuwerten. Die wesentlichen Schritte in diesem Zusammenhang sind:

- Abbildung der Testfälle in Testskripte
- Einbau von Mess- und Prüfpunkten in die Testskripte
- Kopplung von Testskripten und Bewegungsdaten
- Kommunikationsbezogene Dynamisierung der Testskripte
- Konfiguration und Kalibrierung der Simulationsumgebung
- Initialisierung der Testumgebung und Aktivierung der Monitore, gegebenenfalls mit Unterstützung des Systembetriebes
- Durchführung der automatisierten Performancetests unter ständiger Kontrolle der Testverläufe, beispielsweise mittels Online-Monitore
- Zusammenführung und statistische Aggregation der Messdaten
- Konsistenz- und Plausibilitätsprüfung
- Allgemeine und testzielbezogene Interpretation
- Erarbeitung von Handlungsempfehlungen

Testinterpretation

Um in dem abschließenden Testreport eine möglichst vollständige, fehlerfreie, plausible und allseits akzeptable Interpretation der Performancetest-Resultate zu gewährleisten, sollte das Testteam seine Erkenntnisse mit den übrigen Parteien intensiv diskutieren und abgleichen. Zudem werden hierdurch das gemeinsame Performancetest-Verständnis aller Parteien und somit auch deren Motivation zur Zusammenarbeit nachhaltig gefördert.

Zum Schluss unserer logistischen Betrachtungen weisen wir darauf hin, dass der von uns zugrunde gelegte chronologische Verlauf von Performancetest-Projekten stark vereinfacht ist. So werden etwa die diskutierten Schritte in den seltensten Fällen konsekutiv, überschneidungsfrei und ohne Rekursionen durchführbar sein. Unabhängig hiervon lässt sich jedenfalls konstatieren, dass Performancetest-Projekte im unternehmerischen Umfeld keine alleinige Angelegenheit des Testteams sind, sondern in fast allen Projektphasen die aktive und teilweise aufwändige Mitarbeit der anderen Parteien erfordern. Dies ist in Abbildung 5 noch einmal zusammenfassend dargestellt. Um hierbei Missverständnisse und Reibungsverluste möglichst gering zu halten, sollte das Testteam seine technische und methodische Testkompetenz überall lenkend einbringen und sich nie mit den Rollen des Erfüllungsgehilfen und Bittstellers zufrieden geben.

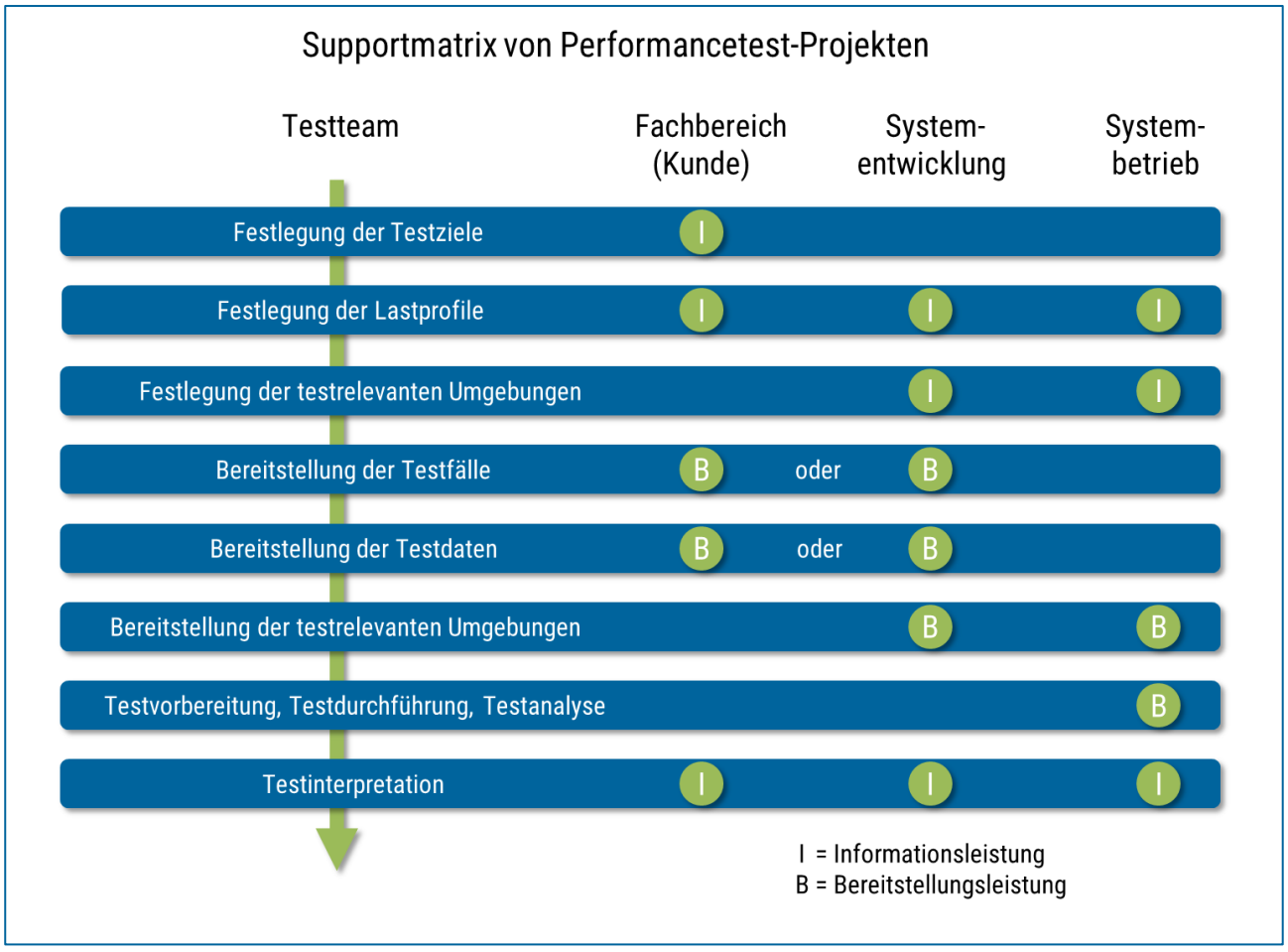

Abbildung 5: Idealisierte Supportmatrix von Performancetest-Projekten aus Sicht des Testteams. Auf einige Leistungen kann gegebenenfalls verzichtet werden, sofern diese im Rahmen von gleich gelagerten Vorgängerprojekten bereits erbracht wurden.

4 Organisatorische Aspekte

Vor dem Hintergrund der vorangegangenen technischen, methodischen und logistischen Betrachtungen von Performancetests im unternehmerischen Umfeld beschäftigen wir uns in diesem vorletzten Kapitel mit deren organisatorischen Aspekten. Hierzu beginnen wir mit einer Beschreibung der Probleme, die von den involvierten Parteien und von uns als Folge einer unzureichenden Organisation am häufigsten gesehen werden. Manche von ihnen wurden bereits angesprochen. Anschließend wenden wir uns deren tieferen Ursachen zu, um auf dieser Basis einige grundlegende organisatorische Verbesserungsmaßnahmen vorzuschlagen.

Problembeschreibung

Aufgrund der zahlreichen Projekte und Reviews zum Thema Performance-Qualitätssicherung, die wir in großen Unternehmen in den letzten zehn Jahren durchgeführt haben, lassen sich die von den involvierten Parteien immer wieder genannten Kritikpunkte wie folgt zusammenfassen:

Kritik des Fachbereiches

Im Zuge der Beauftragung von Performancetest-Projekten vermisst der Fachbereich vor allem eine hinreichende Transparenz in Bezug auf Mehrwert, Kosten und Aufwände, die damit für ihn verbunden sind. Zudem fühlt er sich schlecht beraten, was die Empfehlung von zielführenden Performancetest-Szenarien am Projektanfang und die Interpretation der Testresultate am Projektende betrifft.

Kritik des Testteams

Das Testteam identifiziert im wesentlichen drei Probleme: erstens sind die benötigten Leistungen der anderen Parteien mit massiven Verzögerungen und/oder Qualitätsmängeln behaftet. Zweitens stehen zu wenige und zu kurze Zeitfenster für aussagekräftige Performancetests zur Verfügung. Und drittens ist das Testteam angesichts der vielen, gleichzeitig zu bearbeitenden Testprojekte überlastet. Aufgrund all dessen sieht sich das Testteam zu einer Arbeitshaltung genötigt, bei der die Erreichung hoher Testdurchsätze im Vordergrund steht, zu Lasten der Qualität.

Kritik der Systementwicklung und des Systembetriebes

Hauptkritikpunkt beider Parteien ist, dass die vom Testteam benötigten Leistungen nicht eingeplante Zusatzaufwände sind, die neben dem eigentlichen Tagesgeschäft erbracht werden müssen. Weiterhin wird bedauert, dass Performancetests ausschließlich im Rahmen von Abnahmeprüfungen stattfinden, obwohl sie auch bei entwicklungs- und betriebsrelevanten Fragestellungen wertvolle Erkenntnisse liefern könnten.

Eine detailliertere und nach anderen Kategorien geordnete Betrachtung der Problemlage liefert folgendes Bild:

Allgemeine Defizite

- In Projekten wird die Performance-Qualitätssicherung hinsichtlich Aufwände und Testzeiträume zu wenig und zu spät berücksichtigt. Interferenzen mit parallel stattfindenden Aktivitäten werden nicht vollständig aufgelöst.
- Diskussionen bezüglich der Kundenziele und deren Transformation in operationale Testziele/Testszenarien inklusive Erwartungen/Performance Requirements finden nicht statt.
- Performancetests und insbesondere die dafür zu erbringenden Leistungen werden als lästig, unnötig oder nicht zielführend empfunden.
- Die involvierten Parteien besitzen kein eindeutiges und allseits verbindliches Aufgaben- und Rollenverständnis.

Technische und logistische Defizite

- Die Ausstattungsmerkmale der bereitgestellten Umgebungen entsprechen nicht den Anforderungen. Zudem werden Abhängigkeiten der Testumgebung von anderen Plattformen nicht ausreichend unterdrückt.
- Testdatenbestände sind in ihren qualitativen und quantitativen Ausprägungen unzureichend.
- Das Zurücksetzen der Testumgebung in ihren Ausgangszustand ist nicht möglich.
- Das Monitoring der Test- und Simulationsumgebungen entspricht nicht den Vereinbarungen: falsche Loglevels, falsche Messintervalle, falsche oder unvollständige Metriken etc.

Methodische Defizite

- Simulative Lastprofile werden aus der nutzerbezogenen Perspektive heraus „realitätsnah“ definiert und sind überkomplex. Ex-Post-Prüfungen der Realitätsnähe finden nicht statt.
- Bei der Wahl der Testumgebung stehen nicht die relevanten Fragen im Vordergrund, sondern die Nähe zur Produktionsumgebung.
- Statistische Analysen der Messdaten beschränken sich auf Mittelwerte. Andere Lage-, Konzentrations-, Streu- und Genauigkeitsmaße bleiben außen vor.
- Betrachtungen der Testresultate hinsichtlich Konsistenz, Plausibilität und Messartefakte werden nicht angestellt. Ebenso fehlt der Vergleich mit Performance-Modellvorhersagen.
- Insgesamt werden die drei Kardinalforderungen *Reproduzierbarkeit*, *Interpretierbarkeit* und *Zielgerichtetheit* nicht ausreichend berücksichtigt.

Problemanalyse

Aus unserer Sicht lassen sich die meisten Probleme auf vier primäre Faktoren zurückführen, in deren Zusammenhang eine weitere, bislang unberücksichtigte Instanz zu sehen ist, nämlich das *Management*:

Faktor Kompetenz

Häufig sind beim Testteam Kompetenzdefizite insbesondere bezüglich der methodischen Aspekte von Performancetests festzustellen. Dies zeigt sich unter anderem in einem mangelnden Verständnis der allgemeinen und spezifischen Performance-Zusammenhänge von IT-

Systemen. Somit kann das Testteam keine dezidierte Haltung in Bezug auf Testziele, Testgestaltung und Testinterpretation entwickeln, geschweige denn gegenüber den anderen Parteien selbstsicher vertreten. Dies führt dazu, dass das Testteam grundsätzlich sehr passiv auftritt und in die Rollen des unkritischen Erfüllungsgehilfen (gegenüber dem Kunden) und Bittstellers (gegenüber den anderen Parteien) verfällt. Dieser Mangel an geistiger Führerschaft wird sehr wohl registriert und ist auch mitverantwortlich für die fehlende Motivation der anderen Parteien zur engagierten Mitarbeit.

Faktor Kommunikation

Im allgemeinen haben Performance Engineers in Unternehmen einen schweren Stand, weil ihre Arbeitsleistung schwer einzuschätzen ist und allenfalls dann goutiert wird, wenn gravierende Performance-Mängel identifiziert und beseitigt wurden. Hinzu kommt, dass insbesondere das Management die Komplexität und Aufwände einer verlässlichen Performance-Qualitätssicherung oftmals unterschätzt. Auch in dieser Hinsicht agiert das Testteam zu passiv, indem es seine Arbeitsvoraussetzungen, Arbeitsweise und Arbeitserfolge nicht ausreichend herausstellt und kommuniziert.

Faktor Standardisierung

Ein weit verbreitetes Problem ist, dass aus gedanklicher Bequemlichkeit oder aus Gründen der Flexibilität gerne darauf verzichtet wird, die Leistungen von Unternehmenseinheiten vollständig zu standardisieren. Die konkrete Folge in unserem Fall ist, dass die im Rahmen von Performancetests zu erbringenden Informations- und Bereitstellungsleistungen nicht angemessen eingeplant und budgetiert werden können. Unter anderem deshalb fällt es auch dem Testteam schwer, dem Kunden eindeutige und belastbare Leistungs-, Aufwands- und Kostenzusagen zu machen.

Faktor Steuerung

Der Mangel an Kompetenz, Kommunikation und Standardisierung ist auch ein Grund dafür, dass in vielen Unternehmen keine adäquaten Mechanismen zur Steuerung von Performancetest-Projekten existieren. Insbesondere fehlen geeignete Ablauf-, Aufgaben- und Rollenkonzepte sowie eine parteiübergreifende, mit der nötigen Autorität ausgestattete Kontroll- und Eskalationsinstanz.

Organisatorische Empfehlungen

Angesichts dieses Befundes erscheinen uns folgende organisatorische Verbesserungsmaßnahmen sinnvoll, die insgesamt auf die Etablierung geeigneter Performancetest-*Prozesse* hinauslaufen. Sie alle setzen eine kollegiale und konstruktive Arbeitsatmosphäre mit einer angstfreien Lern- und Diskussionskultur voraus:

Kompetenz

Zur Beseitigung der Kompetenzdefizite des Testteams schlagen wir Fortbildungskurse mit folgenden Themenschwerpunkten vor:

- Konstruktive und analytische Performance-Qualitätssicherung
- Performance-Modellierung von IT-Systemen
- Technische, methodische und experimentelle Aspekte von Performancetests

- Analyse und Interpretation von Performancetests
- Simulations-, Monitoring- und Analyseprodukte
- Performance-Aspekte der kundenrelevanten IT-Infrastrukturen
- Projekt- und Prozessmanagement

Zudem sollte dem Testteam genügend Freiraum gegeben werden, das erlernte Wissen praktisch zu erproben und kontinuierlich fortzuentwickeln. Schließlich ist bei der Auswahl von Performance Engineers darauf zu achten, dass sie über ein ausreichendes technisches, methodisches, experimentelles, analytisches, kommunikatives und koordinatives Potential verfügen.

Insgesamt wird auf diese Weise das Testteam dazu befähigt, fundierte Best Practice-Ansätze für Performancetest-Projekte zu entwickeln und diese nach außen selbstbewusst zu vertreten.

Kommunikation

Beim Faktor Kommunikation geht es zum einen um ein parteiübergreifendes Verständnis von den Voraussetzungen, Abläufen und Zielen von Performancetest-Projekten, um eine engagierte und qualitativ hochwertige Zusammenarbeit zu gewährleisten. Zum anderen müssen die zugehörigen Kosten-Nutzen-Relationen gegenüber dem Management plausibel kommuniziert werden, damit von seiner Seite genügend Unterstützung in Form von Wertschätzung, Personal und Budgets erfolgt.

Neben einem möglichst häufig stattfindenden informellen Austausch bieten sich hierzu zwei gesonderte Gesprächsformate an:

- Workshops, in denen Testteam, Systementwicklung und Systembetrieb allgemeine und projektspezifische Performancetest-Aspekte gemeinsam erörtern.
- Statusmeetings, in denen das Management über aktuelle Performance-Qualitätssicherungsaktivitäten, deren Erfolge, Probleme und Aufwände informiert wird.

Standardisierung

Idealerweise sollte jede Partei über einen eigenen Leistungskatalog verfügen, der sämtliche performancetest-relevante Arbeitspakete inklusive Voraussetzungen, Ergebnisse, Aufwände und Kosten umfasst. Dies bietet den Vorteil, dass Performancetest-Projekte insgesamt und in Teilen besser geplant, durchgeführt, kontrolliert und budgetiert werden können. Insbesondere ermöglicht es dem Testteam, ein kundenorientiertes Performancetest-Portfolio aufzubauen.

Auf dem Weg dorthin ist zunächst zu klären, welche Kundenwünsche durch Performancetests befriedigt werden sollen und wie die kundenrelevanten IT-Infrastrukturen beschaffen sind. Hieraus lassen sich bestimmte Arten von Performancetests mit jeweils ähnlichen operationalen Testzielen und Testvoraussetzungen ableiten, die den Grundstock des Performancetest-Portfolios bilden. Anschließend müssen die jeweiligen Abläufe und logistischen Voraussetzungen der einzelnen Performancetest-Arten akribisch durchdekliniert werden, beispielsweise im Rahmen der oben genannten Workshops und durch detaillierte Ausarbeitung der Supportmatrizen in Abbildung 5. Im Rahmen einer solchen Systematik können schließlich das Performancetest-Portfolio sowie die zugehörigen Leistungskataloge konsequent entwickelt werden.

Steuerung

Der Faktor Steuerung umfasst die Planung, Durchführung, Überwachung und Optimierung von Performancetest-Projekten, dergestalt, dass eine möglichst hohe Qualität durch möglichst wenig Aufwand erzielt wird (*Effektivitäts- und Effizienzprinzip*). Dabei unterliegen selbstverständlich alle Steuerungsaktivitäten einem fortlaufenden Lern- und Reifungsprozess (siehe Abbildung 6). Jenseits der etablierten Steuerungsinstrumente, die in diversen nationalen und internationalen Projektmanagement-Standards (GPM, PRINCE2, HERMES, DIN 69901, ISO 21500 etc.) dokumentiert sind, scheinen uns im vorliegenden Kontext drei Punkte von besonderer Bedeutung zu sein:

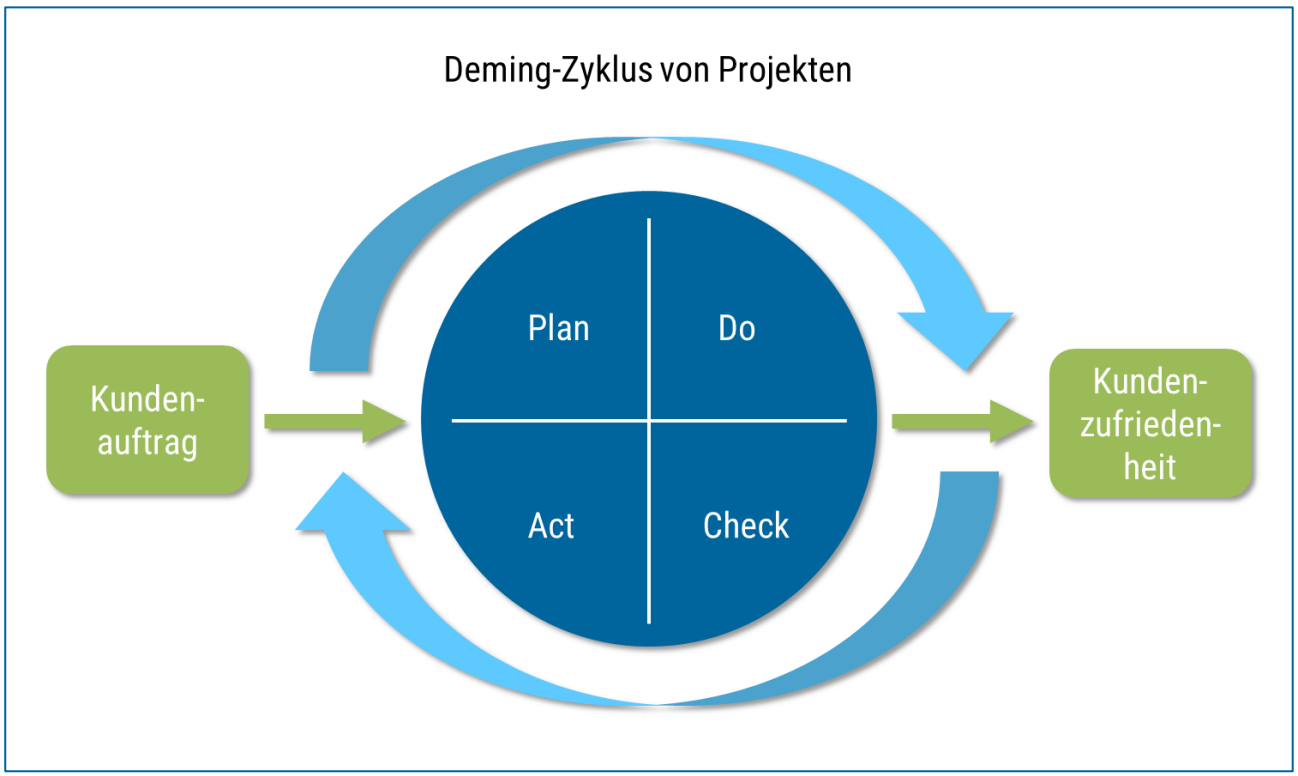

Abbildung 6: Allgemeines Schema der iterativen Qualitätsverbesserung von Projekten, wie es in einschlägigen DIN- und ISO/IEC-Normenfamilien zum Ausdruck kommt. Jeder Iterationsschritt besteht aus einem *Plan-Do-Check-Act-Zyklus* (*Deming-Zyklus*), in welchem ein Projekt zunächst geplant (*Plan*), dann ausgeführt (*Do*), dann geprüft (*Check*) und schließlich verbessert (*Act*) wird.

Zunächst sollte für jedes Performancetest-Projekt ein chronologischer Ablaufplan ausgearbeitet werden, in welchem sämtliche Projektschritte inklusive ihrer (erwarteten) Voraussetzungen, Ergebnisse, Start- und Endzeitpunkte dokumentiert sind. Hierbei sind auch etwaige Aktivitätskonflikte aufzulösen, bei denen zur selben Zeit auf dieselben Ressourcen zugegriffen wird.

Weiterhin sollte es ein tragfähiges Aufgaben- und Rollenkonzept geben, das wenigstens folgenden Basismechanismus abdeckt:

- Jede Partei besitzt einen Ansprechpartner nach außen (*Single Point of Contact*), der für die Erbringung der jeweiligen Projektleistungen gemäß der definierten Abläufe verantwortlich ist.

- Es existiert ein übergeordneter, vom Management mit genügend Rückendeckung und Durchgriffsrechten ausgestatteter Projektleiter, der sämtliche Abläufe auf ihren plangemäßen Vollzug prüft und notfalls korrigierend eingreift. Zudem steht er allen Parteien als Eskalationsinstanz zur Verfügung.

Schließlich sollten geeignete effektivitäts- und effizienzbezogene Projektkennzahlen etabliert werden, um darüber die Projektqualität eindeutig bestimmen und in weiteren Projektzyklen iterativ verbessern zu können. In diesem Zusammenhang ist der Begriff der *Projektreife* instruktiv, der beispielsweise im Rahmen von CMMI (*Capability Maturity Model Integration*) folgendermaßen definiert wird:

- *Reifegrad 1: Initial*
  Arbeitsabläufe werden ad hoc durchgeführt.
- *Reifegrad 2: Geführt*
  Arbeitsabläufe werden unter Anleitung ausgeführt; ähnliche Projekte können erfolgreich wiederholt werden.
- *Reifegrad 3: Definiert*
  Arbeitsabläufe werden in standardisierter Weise geplant und ausgeführt.
- *Reifegrad 4: Quantitativ geführt*
  Es werden quantitative Ziele für die Projektqualität etabliert und als Kriterien für das Projektmanagement verwendet.
- *Reifegrad 5: Optimiert*
  Arbeitsabläufe werden auf der Grundlage von quantitativen Qualitätszielen ständig analysiert und verbessert.

5 Zusammenfassung

Als eine spezielle Form von physikalischen (und erkenntnisfördernden) Experimenten müssen Performancetests die drei Kardinalforderungen *Reproduzierbarkeit*, *Interpretierbarkeit* und *Zielgerichtetheit* erfüllen. Im arbeitsteiligen unternehmerischen Umfeld kommt als vierte Forderung die der praktischen *Umsetzbarkeit* hinzu. Vor diesem Hintergrund wurden in diesem Artikel einige wichtige Gesichtspunkte von *geschlossenen Performancetests gegen interaktive Systeme* aus den Perspektiven Technik, Methodik, Logistik und Organisation überblicksartig beleuchtet.

Technische Aspekte

Der generelle experimentelle Aufbau solcher Tests besteht (i) aus der Testumgebung, also dem zu testenden System selbst, (ii) aus einer Simulationsumgebung, mit deren Hilfe die Testumgebung in automatisierter Weise kontrolliert unter Last gesetzt wird, und (iii) aus einer Monitoring-Umgebung, mittels derer das Leistungsverhalten der Test- und Simulationsumgebungen gemessen wird.

Für die Planung von sinnvollen Performancetests ist es zunächst von entscheidender Bedeutung, sämtliche experimentellen Freiheitsgrade der drei Umgebungen inklusive Lastgeneration

und Testdaten zu identifizieren und deren (vermutete) Einflüsse auf die Testresultate möglichst genau zu bestimmen.

Methodische Aspekte

In methodischer Hinsicht stellen sich vor allem zwei Herausforderungen, nämlich (i) das experimentelle Setup von Performancetests in Übereinstimmung mit den drei Kardinalforderungen zu definieren und (ii) die Testergebnisse statistisch und physikalisch korrekt zu interpretieren. Notwendige Voraussetzungen hierfür sind umfangreiche experimentelle Erfahrungen, ein fundiertes Wissen um die allgemeinen und spezifischen Performance-Zusammenhänge von IT-Systemen sowie tiefergehende Kenntnisse in Statistik und Stochastik.

Konkrete Punkte, die es in diesem Zusammenhang zu berücksichtigen gilt, sind:

- Jeder Performancetest verlangt eine konkrete Haltung bezüglich der zu erwartenden Ergebnisse, mit denen die faktischen Testresultate konfrontiert werden können.
- „Realitätsnahe" Tests beinhalten in der Regel eine hohe Komplexität, die insbesondere zu Lasten der Kardinalforderung nach *Interpretierbarkeit* geht. Das experimentelle Setup von Performancetests sollte deshalb so einfach wie möglich gehalten werden und vornehmlich den jeweiligen Testzielen Rechnung tragen.
- In der Analyse eines jeden Performancetests ist zunächst zu klären, ob er in allen Belangen plangemäß durchgeführt wurde. Unabhängig von den Warnhinweisen der verwendeten Werkzeuge sollten hierzu vor allem die Messdaten auf Konsistenz, Plausibilität und Messartefakte geprüft werden.
- Mittelwerte (und andere statistische Lagewerte) sind nutzlos, solange ihre Genauigkeit bzw. *Reproduzierbarkeit* nicht eingeschätzt werden kann. Die entscheidende Kenngröße in diesem Zusammenhang ist das *Konfidenzintervall*.

Logistische Aspekte

Performancetest-Projekte in Unternehmen sind keine alleinige Angelegenheit des Testteams, sondern bedürfen der Mitwirkung des Fachbereiches, der Systementwicklung und des Systembetriebes in Form von Informations- und Bereitstellungsleistungen. Dies betrifft unter anderem die Festlegung der Testziele, die Definition und Einrichtung der experimentellen Setups sowie die Interpretation der Testresultate.

Um eine möglichst hohe Qualität der Mitarbeit zu gewährleisten, sollte das Testteam in allen Projektphasen die geistige Führerschaft übernehmen, indem es die benötigten Leistungen umfassend motiviert, klar beschreibt, aktiv einfordert und gewissenhaft prüft.

Organisatorische Aspekte

Die Komplexität und der arbeitsteilige Charakter von unternehmerischen Performancetest-Projekten erfordern ein hohes Maß an Organisation. Diesbezügliche, in der Praxis häufig anzutreffende Unzulänglichkeiten lassen sich größtenteils auf Defizite in puncto (i) Testkompetenz, (ii) Kommunikation der involvierten Parteien, (iii) Standardisierung der Projektleistungen und (iv) Steuerung der Projekte zurückführen.

Folgende organisatorische Maßnahmen erscheinen uns daher empfehlenswert:

- Fortbildungen des Testteams in diversen praktischen und theoretischen Disziplinen der Performance-Qualitätssicherung.
- Workshops und Statusmeetings zur Entwicklung einer parteiübergreifenden Sicht auf die unternehmensrelevanten Aspekte der Performance-Qualitätssicherung.
- Detaillierte Ausarbeitung aller performancetest-relevanten Arbeitspakete in Form von parteispezifischen Leistungskatalogen; Aufbau eines kundenorientierten Performancetest-Portfolios.
- Erstellung von chronologischen Projektablaufplänen; Schaffung klarer Projektverantwortlichkeiten und Eskalationswege; Etablierung von qualitäts- und aufwandsbezogenen Projektkennzahlen.

Dr. rer. nat. Armin Wachter, geb. 1968, hat an der Universität Wuppertal Physik studiert und anschließend am John von Neumann-Institut für Computing (NIC) des Forschungszentrums Jülich promoviert. Schwerpunkt seiner wissenschaftlichen Arbeit ist die numerische Bestimmung von Interquarkkräften auf massiv parallelen Hochleistungsrechnern (Gittereichtheorie).

Armin Wachter ist Autor mehrerer Lehrbücher in Theoretischer Physik. Seit 1997 arbeitet er im Bereich IT, wo er sich hauptsächlich mit dem Thema Performance Quality Management befasst. Im Jahre 2006 hat er zusammen mit Maik Karbon die Wachter & Karbon IT-Consulting gegründet.

Wachter & Karbon IT-Consulting GmbH & Co. KG
Hermülheimerstraße 10
50321 Brühl

Telefon: +49 2232 - 3109 36
Telefax: +49 2232 - 3109 35
Internet: www.wachter-karbon.com
E-Mail: info@wachter-karbon.com