2020

FIRMTEC SOLUTIONS AG

# Ganzheitliches Performance-Engineering

## SCHNELLE, STABILE UND SKALIERBARE SOFTWARE-PRODUKTE

## EINSATZMÖGLICHKEITEN UND NUTZEN AUS DER PRAXIS

DR. ANDREAS ELSENER, HEINZ GROB

1. Auflage November 2020

# Inhaltsverzeichnis

***"If the user can't use it, it doesn't work."***
*Susan M. Dray – Expertin für Mensch-Computer-Interaktion*

# Zusammenfassung

IT-User erwarten schnelle und stabile IT-Systeme, um Anwendungsfälle reibungslos zu nutzen. Produktverantwortliche erwarten attraktive Systeme mit einer hohen Kundenzufriedenheit. IT-Serviceverantwortliche erwarten eine hohe Verfügbarkeit der Systeme mit einer guten Wirtschaftlichkeit, ohne unnötige Risiken.

All diese Eigenschaften sind eng mit Performance oder mit der systeminternen Charakteristik der Skalierbarkeit verbunden. Nur Systeme, die mit zusätzlichen Ressourcen effektiv mehr leisten, also skalierbar sind, können nachhaltig mit schwankenden Lastszenarien umgehen und steigende Geschwindigkeitsanforderungen erfüllen. Aufgrund der zunehmenden Komplexität und der erhöhten Abhängigkeiten zwischen Teilsystemen und Komponenten fehlt es Software immer häufiger an dieser Skalierbarkeit.

Ein nichtskalierbares System verhält sich unter Last analog zu einer Autobahn mit hohem Verkehrsaufkommen: Zuerst wird das System spürbar langsamer und ab einem gewissen Punkt ist es wie eine stehende Kolonne. Im schlimmsten Fall kollabiert das System, wird instabil und stürzt ab. Die wahre Herausforderung ist, Software von Anfang an skalierbar zu bauen. Eine schlechte Alternative, die leider häufig gewählt wird, ist die kostspielige Überdimensionierung der Plattform und der Infrastruktur, um für alle Fälle vorbereitet zu sein. Im Falle von statischen, nichtskalierbaren Systemen wie einer Autobahn ist das gerechtfertigt. Im Falle von IT Systemen ist es nachhaltiger, wenn die Ursachen für die fehlende Skalierbarkeit gefunden und beseitigt werden.

Performance-Tests knapp vor der Einführung reichen bei weitem nicht, um Performance-Probleme zu beheben. Performance-Monitoring alleine führt zu einer rein reaktiven Performance-Kultur. Darum braucht es Performance-Engineering.

Performance-Engineering ist die zentrale Disziplin, die sich nachhaltig und ganzheitlich um Geschwindigkeit, Stabilität und Skalierbarkeit von IT-Systemen kümmert. Es besteht aus einem Set von gut aufeinander abgestimmten Massnahmen, verteilt über den ganzen Software-Lebenszyklus.

## Nutzen von Performance-Engineering

***Doppelter Gewinn:*** *Performance-Engineering hilft Erträge zu steigern und gleichzeitig Kosten zu reduzieren. In diesem Sinne zahlt es sich doppelt aus.*

***Erfolg- und Effizienzsteigerung:*** *Schnelle, stabile und skalierbare Applikationen bringen eine hohe Anwenderzufriedenheit und entsprechend gute Bewertungen. Das hat einen direkten Einfluss auf Umsatzzahlen[1]. Erwiesenermassen arbeiten auch Mitarbeiter auf schnellen IT-Systemen selber schneller und sind damit effizienter[2].*

***Reputationsschäden vermeiden:*** *Performance-Tests mit katastrophalen Ergebnissen kurz vor der Einführung können zu Projektabbrüchen oder grossen Verzögerungen führen. Im schlimmsten Fall führt die Einführung der Software zum Debakel, was in der heutigen Zeit ein gefundenes Fressen für die einschlägige Presse ist[3]. Darum sollte die Zeit im Projekt effektiv für ein umsichtiges Performance-Engineering genutzt werden.*

***Kostenreduktion:*** *Performance-Engineering macht Applikationen ökonomischer, indem Infrastruktur-, Lizenz- und Personalkosten gespart werden.*

[1] http://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales
[2] https://jlelliotton.blogspot.com/p/the-economic-value-of-rapid-response.html
[3] http://www.inside-it.ch/articles/39864

# Lesehilfe

Das Dokument geht in der Einleitung auf zentrale Fragen ums Perfomance-Engineering ein: «Warum Performance-Engineering?», «Was ist Performance?», «Wie arbeitet ein Performance-Engineer?» und natürlich auch «Wer profitiert von Performance-Engineering?». Dieses Kapitel richtet sich an Leser, die in der Vergangenheit weniger Bezug zu Performance-Engineering hatten.

Anschliessend werden exemplarische Einsatzgebiete für Performance-Engineering mit Beispielen aus der Praxis beleuchtet. Der Fokus liegt auf typischen Herausforderungen und wie sie gelöst werden können und was dabei für ein Nutzen entsteht. Die Idee ist, dass die relevanten Einsatzgebiete selektiv gelesen werden.

Das Kapitel «Performance-Engineering in der Praxis» vertieft das Thema Performance-Engineering für den interessierten Leser. Dabei wird auch auf organisatorische Themen wie beispielweise eine Performance-Engineering-Governance eingegangen.

Zum Schluss wird noch auf die FirmTec-spezifische Sicht eingegangen: Das Kapitel zeigt den FirmTec-Ansatz, die Angebote, sowie die Autoren dieses Werkes. Zu allerletzt werden wichtige Begriffe in einem kurzen Glossar zusammengefasst.

# Einleitung

## Warum Performance-Engineering?

### Allgemeine Situation

Immer mehr interagieren Kunden direkt mit den Systemen von Anbietern, weil die Digitalisierung unaufhaltsam voranschreitet. Immer mehr IT-Systeme werden miteinander integriert. Damit steigen die Abhängigkeiten und die Komplexität der IT-Landschaften enorm. In einer solchen Welt haben kleine Ursachen eine immer grössere Wirkung.

Mit der Zunahme der Digitalisierung gibt es eine immer grössere Anzahl von Anwendern und Benutzern von IT-Systemen. In einer stetig mehr integrierten Welt sind das nicht alleine losgelöste Endkundensysteme. Die Kunden greifen über verschiedene Zwischensysteme auf die Kernsysteme der Unternehmen oder der Verwaltung zu. Daraus resultiert, dass die oben erwähnten kleinen Ursachen nicht nur eine lokale, firmeninterne Wirkung haben, sondern auch immer mehr Kunden direkt betreffen und im schlimmsten Fall sogar eine globale Wirkung haben.

### Herausforderung Kundenwert

Für die zukünftigen Umsatzzahlen und die Attraktivität einer Unternehmung ist der empfundene Kundenwert von Applikationen zu einer zentralen Grösse geworden. Was heisst der Begriff Kundenwert? Er definiert sich als Kundenerlebnis dividiert durch die Erwartung des Kunden.

$$Empfundener\ Kundenwert = \frac{Erlebnis}{Erwartung}$$

Das heisst, dass eine Verbesserung oder Verschlechterung des Kundenerlebnis direkt auf den Kundenwert einer Applikation wirkt. Wenn Mitbewerber ihre Angebote oder Software verbessern, steigen die Erwartungen ans eigene Produkt. Bei einem gleichbleibenden Kundenerlebnis nimmt der empfunden Kundenwert ab, was die Formel mit der Erwartung im Nenner ausdrückt: Grössere Erwartung heisst kleiner Kundenwert. Fazit daraus ist, dass das Kundenerlebnis und der dadurch empfundene Kundenwert laufend optimiert werden müssen, weil sich die Mitbewerber ebenso verhalten.

### Performance als Fitness oder Gesundheit

Die Performance eines IT-Systems oder einer Applikation kann als Fitness oder Gesundheit betrachtet werden. Im gesunden Zustand läuft die Applikation reibungslos und ist entsprechend ressourceneffizient. Erste Performance-Probleme können schon im unbelasteten Zustand auftreten. Spannender ist auf jeden Fall die Untersuchung der Performance im betriebsnahen, also belasteten Zustand einer Applikation. Analog zu einem Belastungs-EKG für unseren Körper können so wunde Punkte schneller identifiziert werden.

# Was ist Performance?

Performance von IT-Systemen wird in der deutschen Sprache häufig mit dem Begriff «Effizienz» übersetzt. Der Effizienzbegriff beschreibt gemäss DIN-ISO-Norm 9241-11, *«das Verhältnis zu dem Ausmass, in dem Benutzer spezifische Ziele erreichen, gegenüber den eingesetzten Mittel».* Dieser Begriff hilft in der Praxis aber leider zu wenig. Praxisnahe kann der Begriff Performance auf die folgenden drei Hauptaspekte reduziert werden: Geschwindigkeit, Stabilität und Skalierbarkeit. Oder analog auf Englisch «Speed, Stability and Scalability». Diese drei Begriffe sind auch als die drei «S» im Bezug Performance bekannt.

## Geschwindigkeit

Ein performantes IT-System muss seinen Benutzern in einer angemessenen Zeit die Daten zur Verfügung stellen, die gesucht werden bzw. die Verarbeitungen durchführen, die beauftragt wurden. Es geht also um Antwortzeiten, Verarbeitungszeiten oder Durchlaufzeiten. Ebenfalls können Reaktionszeiten auf Klicks gemeint sein oder Renderraten beim Aufbau von Darstellungen und bei der Anzeige von bewegten Bildern. Die Geschwindigkeit wird meist aus der Sicht des Anwenders oder der involvierten Prozesse betrachtet.

## Stabilität

Ein performantes IT-System muss sich stabil verhalten. Dies gilt insbesondere, wenn es einer gewissen Belastung ausgesetzt ist. Die Belastung kann beispielsweise von der Menge gleichzeitig agierender Benutzer oder von vielen technischen Abfragen oder einer entsprechend grossen Batchverarbeitung kommen. In allen Fällen wird erwartet, dass ein System der Last standhält und sich für alle Benutzer «stabil» verhält. Hier ist auch die Benutzer- oder Prozesssicht entscheidend.

## Skalierbarkeit

Im Falle der Skalierbarkeit geht es darum, dass ein System unter Hinzunahme von weiteren Ressourcen auch effektiv mehr leisten kann. Grundsätzlich sind Engpässe in der Skalierbarkeit die Ursachen für erhöhte Antwortzeiten, also eine tiefere Verarbeitungsgeschwindigkeit und nicht zuletzt auch der limitierende Faktor für die Stabilität. Ein wichtiger Faktor für bessere oder schlechtere Skalierbarkeit ist der Anteil des sequentiell ausgeführten Codes[4]. Dieser sollte so klein wie möglich sein. Die Skalierbarkeit ist die systeminterne Sicht auf die Performance.

[4] Amdahlsches Gesetz: https://de.wikipedia.org/wiki/Amdahlsches_Gesetz

# Wie arbeitet ein Performance-Engineer?

Hauptziel eines Performance-Engineers ist das aktive Verbessern der Performance statt eines reaktiven Eingreifens bei akuten Problemen. Wenn vorher die Rede war, dass Performance-Engineering Medizin für Applikationen ist, dann ist die Arbeit eines Performance-Engineers mit der Arbeit von Ärzten wie folgt vergleichbar.

## Prävention

Mit gezielten Aktivitäten sollen Performance-Probleme aktiv vermieden werden. Das beginnt bei der Sensibilisierung von Anforderungsstellern, Entwicklern und Betriebsmitarbeitern. Weiter werden organisatorische, methodische und technologische Mittel zur Vermeidung von Performance-Problemen geschaffen. Konkret werden zeitgerecht Performance-Tests durchgeführt, um allfällige Probleme weit vor der Einführung zu antizipieren.

## Symptome

Eine grundlegende Tätigkeit eines Performance-Engineers ist die Quantifizierung des erlebten Benutzerverhaltens. Es geht um aussagekräftige Messungen von Antwortzeiten, aber auch um die Vermessung von Applikationen in gewissen Belastungszuständen. Dies erfolgt in Entwicklungsphasen oder auch als Monitoring im produktiven Betrieb. Bereits mit wenigen Hilfsmitteln können von aussen Performance-Engpässe symptomatisch beschrieben werden.

## Anamnese

Die professionelle Erfragung von Performance-relevanten Informationen hilft bei der Einordnung von aktuellen Benutzerwahrnehmungen in Bezug auf die Möglichkeiten eines IT-Systems. Ein wichtiger Schritt ist das Studium der Architektur des betroffenen Systems oder Systemverbunds.

## Diagnose

Typischerweise reicht die reine Aussensicht nicht, um Performance-Probleme zu beheben. Um eine Diagnose zu stellen braucht es Performance-Analysen. Die Diagnose soll möglichst punktgenau die Ursache eines allfälligen Performance-Engpasses benennen. Das könnte unter anderem eine Datenbankabfrage, ein ausgeschöpfter Ressourcen-Pool oder ein Stück sequentieller Code sein.

## Therapie

Ein Performance-Engineer eröffnet bei Performance-Problemen nicht einfach ein Fehlerticket im Bugtrackingsystem, sondern er kümmert sich aktiv um die Behebung des Engpasses. Gemeinsam mit Solution-Architekten und Software-Entwicklern schlägt er geeignete Massnahmen zur Verbesserung der Performance vor und treibt die Umsetzung voran.

## Prognose

Mit Performance-Engineering werden die Geschwindigkeit, Stabilität und Skalierbarkeit von IT-Systemen nachhaltig verbessert. Das hat einen Einfluss auf die Betriebskosten einer Applikation, weil sie mit weniger Infrastruktur betrieben werden kann und sie weniger manuelle Eingriffe, wie beispielsweise Restarts, benötigt.

# Wer profitiert von Performance-Engineering?

Verschiedene Rollen in Unternehmungen von Fach und IT können von der Zusammenarbeit mit einem Performance-Engineer profitieren. In allen Fällen geht es um ein Geben und Nehmen. Das heisst, ein Performance-Engineer ist umgekehrt auf die Zusammenarbeit mit verschiedenen Stakeholdern angewiesen.

## Product-Owner / Fachvertreter

Das Fach trägt üblicherweise die Produktverantwortung. Entsprechend müssen die Fachvertreter auch realistische Anforderungen an die Performance stellen. Dies ist meist ein Subset der sogenannten nichtfunktionalen Anforderungen. Dazu gehören neben der Leistung und Effizienz auch die Zuverlässigkeit, die Benutzbarkeit, die Skalierbarkeit und weitere Anforderungen.

Im Idealfall verantwortet der Product-Owner auch Kosten für die Weiterentwicklung und den Betrieb. Entsprechend ist eine effizient betreibbare Software ein grosser Gewinn. Ein Product-Owner profitiert aber von Performance-Engineering in jedem Fall durch Ausbleiben von Verzögerungen in der Weiterentwicklung und durch bessere Kundenfeedbacks.

## IT-Management / Service-Management / Betriebsorganisation

Das IT-Management muss üblicherweise die Verfügbarkeit der IT-Services gewährleisten. Dies in einem engen Kostenkorsett. Dazu müssen sinnvolle Entscheidungen für eine ausreichende Betreibbarkeit der Applikationen gefällt werden. Ein wichtiger Punkt dabei ist das Performance-Monitoring.

IT-Management, Service-Manager und Betreiber profitieren von höheren Verfügbarkeitswerten, wenn Applikationen stabiler laufen. Ausfallrisiken können mit Performance-Engineering massiv gesenkt werden. Im Falle von effizient laufender Software, kann der Infrastrukturressourceneinsatz und teilweise auch der Lizenzeinsatz massiv reduziert werden. Das kann einem stark beanspruchten IT-Budget die notwendige Entlastung geben.

## Architekten / Lösungsarchitekten

Architekten müssen die Informationen von Anforderungssteller und Entscheidern für umsetzenden Stellen wie Software-Entwicklung aufbereiten. Sie konzipieren das Lösungsdesign und fällen richtungsweisende Entscheide, wie die Technologiewahl. Der Performance-Engineer baut das Messsetup und das Monitoring basierend auf diesen Entscheidungen auf. Darum sind Architekten zentrale Ansprechpartner für den Performance-Engineer.

Architekten profitieren vom realitätsnahen und aussagekräftigen Feedback des Performance-Engineers, weil sie damit ihre Aussagen und Entscheide belastbarer machen können.

## Projektleiter / Scrum-Master

Projektleiter und Scrum-Master sind Enabler, die ein Vorhaben zum Erfolg führen können, indem sie die wichtigsten Steine aus dem Weg räumen. Für das Performance-Engineering sind sie wichtig, weil sie dem Thema Performance das notwendige Gehör oder Gewicht verschaffen können.

Gleichzeitig profitieren sie davon, dass Zeitpläne nicht durch unerwartete Performance-Engpässe kurz vor der Einführung über den Haufen geworfen werden.

## Entwickler

Software-Entwickler setzen die Anforderungen um. Dabei hat häufig das Funktionale einen viel höheren Stellenwert als das Nichtfunktionale. Der Performance-Engineer bildet hier den Gegenpol, welcher primär auf nichtfunktionale Aspekte achtet. Entwickler kennen aber den Code und können diesen bei Bedarf auch für bessere Analysefähigkeiten zielgerichtet erweitern. Zum Beispiel durch das Hinzufügen von weiteren Logausgaben.

Software Entwickler profitieren, weil sie meist nur wenig Möglichkeiten haben, ihren Code parallel zu testen. Das kann selbst in funktionalen Bereichen zu einer Verbesserung der Qualität führen.

## Test- und Releaseverantwortliche

Test- und Releaseverantwortliche müssen am Schluss ein Set von Anforderungen beziehungsweise Änderungen an einer Software beurteilen und abnehmen oder als abgenommen gemeldet bekommen. Dabei haben auch sie häufig einen funktionalen Fokus. Für Performance-Engineers sind sie relevante Ansprechpartner, weil die Testinfrastruktur, insbesondere die Testumgebungen und die Testdaten, mit den funktionalen Testern geteilt werden müssen. In vielen Fällen sind Tester auch Lotsen, um die Kernfunktionalitäten schneller kennenzulernen.

Wenn es nach der Einführung zu Performance-Problemen kommt, stehen sowohl die Release- wie Testverantwortlichen schlecht da. Sie profitieren darum durch die breitere Abdeckung und minimierte Risiken.

# Einsatzgebiete

## Webapplikationen

### Herausforderung

Webapplikationen werden von einer meist unbekannten und schwankenden Anzahl von Anwendern bedient. Folglich müssen sie Anfragen aller Benutzer skalierbar und vor allem schnell genug verarbeiten können. Dafür müssen ausreichende Ressourcen (Speicher und Rechenkapazität) zur Verfügung stehen. Softwaretechnisch sind typische Schwachpunkte der Datenbankzugriff oder eine unzureichende Parallelisierung der Business-Logik. Weiter ist die Integration mit Umsystemen kritisch, insbesondere, bei synchronen Datenzugriffen.

Wenn früher eher die Bandbreite die Performance limitierte, sind es heute immer aufwändigere, JavaScript-lastige Clients mit komplexer, interner Logik, welche Performance-Probleme verursachen können. Heikel ist hier insbesondere die Speicherverwaltung, wobei Speicherlöcher auftreten können. Das Verhalten des Clients in Bezug auf den Seitenaufbau hat auch eine direkte Wirkung auf das Kundenerlebnis.

Wichtig ist, dass Webapplikationen selten als stand-alone betrachtet werden können, sondern in den Unternehmenskontext stark integriert werden (siehe Abbildung 1).

### Lösung

Webapplikationen können typischerweise sehr gut über die verwendeten Protokolle bedient werden. Damit lässt sich die Last von hunderten bis tausenden von Benutzern effizient simulieren. So können das Geschwindigkeitsverhalten unter Last, sowie die eigentliche Belastungsgrenze einfach ermittelt werden. Wichtig ist, allfällige Engpässe zeitnah mit entsprechenden Analyse-Werkzeugen zu untersuchen. Damit kann die Ursache für Engpässe meist mit geringem Aufwand gefunden werden. Falls Absprünge in Umsysteme kritisch sind, können diese auch auf analoge weise entkoppelt getestet und so ein allfälliger Engpass einfacher lokalisiert werden.

Mit der Auslagerung von Logik in die Clients wird vermehrt die Analyse der Laufzeit im Client wichtig. Falls mehrere Clients zu simulieren sind, gibt es spannende Ansätze über UI-Test-Werkzeuge, die über Performance-Test-Werkzeuge parallel ausgeführt und orchestriert werden. Dieser Ansatz ist insbesondere wichtig, wenn die Javascirpt-intensiven Clients keinen anderen Zugang auf Webtransaktionen ermöglichen. Andernfalls reicht es, wenige Clients gleichzeitig zu simulieren oder im einfachsten Fall sogar das Performanceverhalten im Client nur an einer Instanz exemplarisch zu untersuchen. Realistischer kann diese Analyse in Kombination mit einem Backend erfolgen, welches gleichzeitig über Webtransaktionen oder API-Aufrufe belastet wird. Mit spezifischen Werkzeugen können auch der Seitenaufbau und die geladenen Inhalte bezüglich Performance analysiert werden.

Zur langfristigen Sicherstellung einer hinreichenden Performance, empfiehlt sich aktive Überwachung einer Webapplikation. Dafür gibt es unterschiedliche Lösungen, die von der Enduser-Experience über Business-Transaktionen bis hin zum Ressourcenverbrauch alles oder auch nur Teilaspekte abdecken.

### Nutzen

***Optimales Ranking in Suchmaschinen:*** *Eine hohe Skalierbarkeit ist der Treiber für die Geschwindigkeit und Stabilität im Webumfeld. Von der Geschwindigkeit und Stabilität hängt die Attraktivität von Weblösungen direkt ab. Das Wissen auch Suchmaschinen. Darum werden performantere Seiten in den Suchergebnissen bevorzugt.*[5]

***Performance als Visitenkarte:*** *Firmen mit stabilen und schnellen Kundenkanälen profitieren von einer professionellen Wirkung beim Kunden.*

***Schlanker Infrastruktur-Footprint:*** *Mit Performance-Optimierungen kann der Infrastruktur-Footprint nachhaltig verkleinert werden. Das hat insbesondere bei kostenintensiven Technologien ein grosses Einsparpotential*

[5] https://developers.google.com/web/updates/2018/07/search-ads-speed

# Portale und Apps bei Gesundheitsversicherer

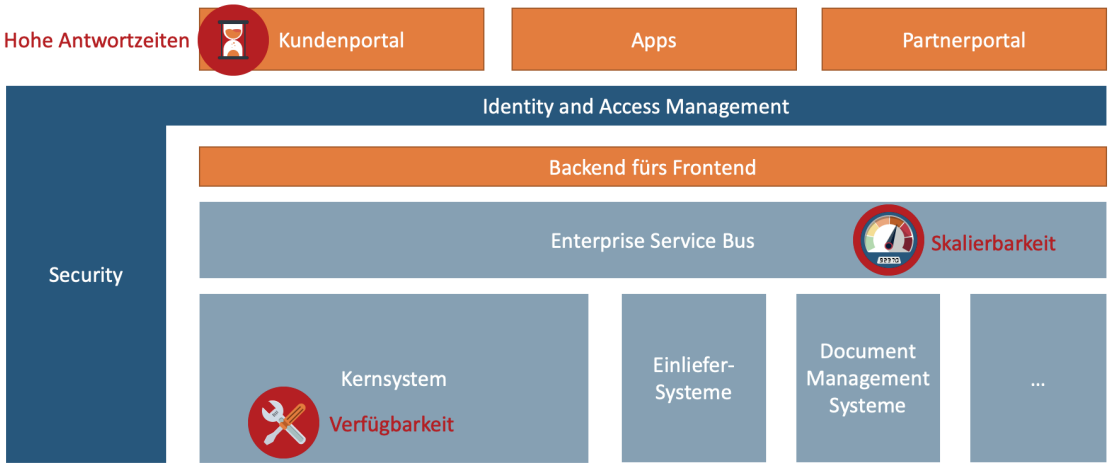


*Abbildung 1: Die Beispiel-Webapplikation bietet verschiedene Kanäle für Kunden und Partner (effektiv Webportale und native, mobile Apps), welche auf einem Micro-Service-Backend aufsetzen (orange). Die Applikation ist mit den bestehenden Enterprise-Applikationen (grau) integriert. Weiter gibt es die Security (blau): Zum Beispiel Identity and Access Management, welches mit der Security der Enterprise-Applikationen harmonieren muss. Im konkreten Beispiel wurden verschiedene Schwierigkeiten gelöst: Hohe Antwortzeiten aus Kundensicht, die fehlende Skalierbarkeit des Enterprise Services Buses und die schlechte Verfügbarkeit des Kernsystems, die auch zu einer schlechten Verfügbarkeit aus Endkundensicht führte.*

***«FirmTec hat umsichtig getestet und beraten,***
***und als Sparring-Partner auf Augenhöhe***
***unsere Kundenkanäle nachhaltig verbessert.»***
*Lösungsarchitekt fürs Kundenportal eines Gesundheitsversicherers*

# Mobile Apps

## Herausforderung

Mobile-Apps verhalten sich ähnlich wie Webapplikationen, wobei meist nur dynamische Daten zum Beispiel über sogenannte REST-Schnittstellen bezogen werden. Im mobilen Umfeld spielt die Bandbreite beziehungsweise die Verbindungsqualität eine grössere Rolle. Wenn viele User mit einer schlechten Anbindung auf Services zugreifen, können Poolressourcen rasch ausgeschöpft sein, was zu Engpässen und Performance-Problemen führt.

Die Business-Logik selbst ist in den meisten Applikationen klein. In der Vergangenheit lief dies meist entkoppelt vom Kernsystem der Unternehmung. In diesem Bereich nimmt die Komplexität spürbar zu. Beispielsweise gibt es immer mehr Apps, die direkt mit Kernsystemen integriert sind und dabei Datenzugriffe über teilweise komplexe Integrationslösungen durchführen. Folglich steigen die Anforderungen an den gesamten Verbund bezüglich Geschwindigkeit aber auch Verfügbarkeit.

Die Entwicklung von Apps passiert häufig losgelöst von der Backend-Entwicklung. Zum Teil ist das noch unternehmensintern zum Teil aber auch extern. App-Entwickler kümmern sich wenig um die Backend-Performance, auch wenn das System am Schluss nur komplett integriert zu betreiben ist (Abbildung 2), weil das nicht in ihrem Verantwortungsbereich liegt. Entsprechend braucht es Vermittler, die sich End-to-End über alle Applikationsschichten um eine gute Performance kümmern.

## Lösung

Einschränkungen in der Bandbreite, zum Beispiel im Falle von mobilen Clients mit schlechter Anbindung, können über entsprechende Lastgenerierungswerkzeuge simuliert werden. Mit extremen Szenarien kann die Grenze eines Applikationsservers bezüglich Ressourcen-Pools effizient ermittelt werden.

Die Entwicklung von neuen Mobile-App-Plattformen und -Werkzeugen ist sehr schnell. Darin werden verschiedene Hilfsmittel zur Analyse und Diagnose für die selbergebauten Apps angeboten. Im komplexeren Falle von hochintegrierten Systemen ist die Überwachung trotzdem anspruchsvoll. Ein zentrales Instrument sind Korrelationen, die den übergreifenden Systemzugriff nachvollziehbar machen. Antwortzeiten können so entsprechend auf die Teilsysteme hinuntergebrochen werden. Damit wird am besten so früh wie möglich begonnen: Zum Beispiel schon im Design oder in den entsprechenden Teilsystemtests.

## Nutzen

***Kundenrating:*** *Im Mobile-Umfeld sind die Kundenbewertungen in App- und Play-Store für den Erfolg von Apps entscheidend. Entsprechend braucht jedes App eine hinreichende Performance, weil sonst das Score im Kundenranking sofort sinkt.*

***Die Konkurrenz schläft nicht:*** *Wenn man Release-Notes bei der Aktualisierung von Apps anschaut, fällt auf, dass immer öfter lediglich Performance- oder Stabilitätsverbesserungen gemacht werden. Mit aktivem Performance-Engineering kann die Konkurrenz punkto erlebtem Kundenwert geschlagen werden*[6]*.*

***Schnelle Backends:*** *Performance-Engineering kümmert sich auch um notwendige Verbesserungen am Backend-System, auch wenn oder gerade, weil diese häufig nicht im Scope der App-Entwickler liegen.*

[6] Siehe Herausforderung Kundenwert (Seite 6)

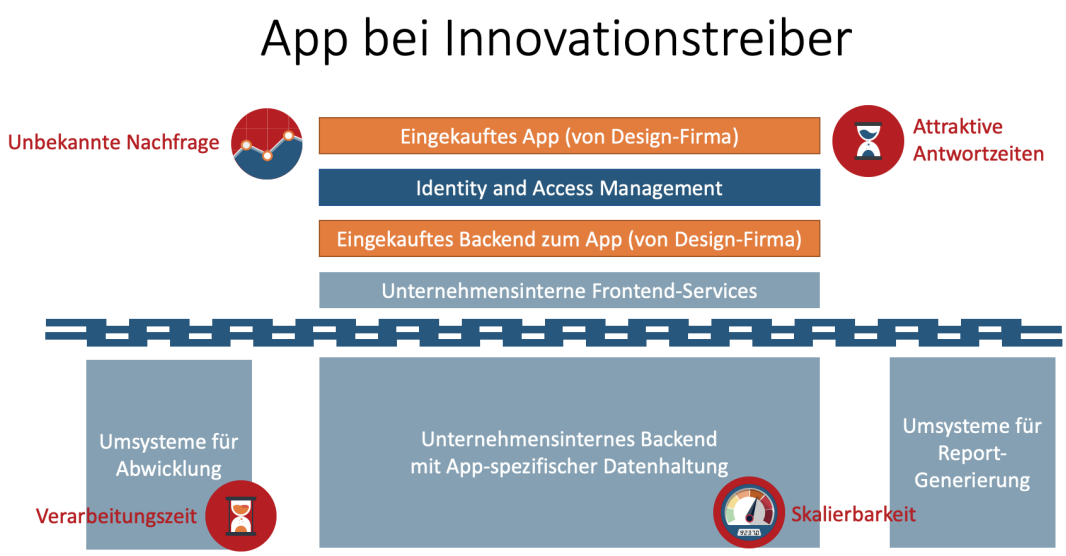


*Abbildung 2: Das Beispiel zeigt eine App für ein innovatives Businessprodukt, das mit einem Drittanbieter aus dem Design-Umfeld entwickelt wurde. Die Schwierigkeit war hier, dass diese Apps in die bestehende Unternehmenslandschaft integriert werden musste. Dabei mussten die Antwortzeiten der App stets im attraktiven Bereich bleiben, weil sonst Neukundschaft bereits beim Laden der Daten abspringen könnte. Erschwerend kam dazu, dass keine Daten über die erwartete Nutzung oder Nachfrage bezüglich der App vorhanden waren, weil es um ein komplett neues Geschäftsmodell ging. Natürlich mussten zu Erfüllung von attraktiven Antwortzeiten nicht nur die neuen, externen Komponenten gut laufen, sondern auch die internen Systeme mussten bezüglich Verarbeitungszeiten und Skalierbarkeit hinreichend optimiert werden.*

***«FirmTec hat das Prio-1-Projekt des CEO im Bereich Performance erfolgreich ins Ziel geführt!»***
*Projektverantwortlicher eines Innovations-Apps im Finanzumfeld*

# Client-Server-Applikationen

## Herausforderung

Webapplikationen, mobile Apps und viele weitere IT-Systeme verwenden heute eine Client-Server-Architektur. Hier soll allerdings spezifisch über Rich-Client-Applikationen gesprochen werden. Damit ist bereits gesagt, dass ein erheblicher Teil der Business-Logik dezentral auf Clients verteilt ist. Diese Logik kann Performance-Einschränkungen auf der Client-Seite mit sich bringen.

Ein erheblicher Einflussfaktor auf die Performance kann die Laufzeitumgebung des Clients beziehungsweise die Art und Weise, wie auf den Client zugegriffen wird, haben (Abbildung 3). Immer mehr Firmen installieren Clients nicht mehr lokal auf Endgeräten, sondern bedienen diese auf zentralen Infrastrukturen und greifen via Virtual Desktop oder Remote-Desktop darauf zu. Diese Art von Virtualisierung kann einen starken Einfluss aufs Benutzererlebnis haben, speziell wenn das Klicken oder Tippen stark verzögert dargestellt wird.

Natürlich ist auch der Server nicht unkritisch, weil sich eine grössere Anzahl Clients je Server verbinden. Das heisst, der Server inklusive der verwendeten Plattformen wie Applikationsserver und Datenbanken müssen die Last bewältigen können. Je nach Technologie ist die Verbindung des Clients zum Server, bzw. die Lastverteilstrategie (Load-Balancing) ein heikler Punkt.

## Lösung

Clients sind isoliert zu untersuchen. Wichtig ist, dass dies in einer realistischen Umgebung passiert, zum Beispiel auf einer typischen Anwender-Workstation mit dem entsprechenden Setup an Ressourcen, oder anderen Softwareprodukten. Mit betriebssystemnahen Werkzeugen und teilweise auch Profiling-Werkzeugen können detailliertere Analyse erfolgen.

Serverseitig kann über Simulation der Client-Requests die Belastung verschiedener Knoten nachgestellt werden. Teilweise bestehen jedoch Schwierigkeiten, wenn proprietäre Schnittstellen oder Socket-Verbindungen im Spiel sind. Dann empfiehlt sich eine Zusammenarbeit mit den Entwicklern oder gegebenenfalls mit dem Hersteller. Alternativ können auch Service-Endpunkte, falls vorhanden, für die Lastgenerierung verwendet werden. Eine eher umständliche Möglichkeit ist Lastgenerierung über eine Farm von Clients. Umständlich, weil dies ein sehr Infrastruktur-aufwendiger, wartungsintensiver und damit teurer Ansatz ist.

Vom Vorgehen einer Lastgenerierung über gleichzeitig aktive, reale Benutzer wird abgeraten, weil dies in den allermeisten Fällen wenig aussagekräftig und das Ergebnis nicht reproduzierbar ist. Anstatt ein System über User zu belasten, kann mit geeigneten Monitoring-Strategien im produktiven Betrieb Rückschluss aufs Performance-Verhalten gezogen werden. Dies erzeugt nach eingesetzter Technologie der Applikation jedoch mehr oder weniger Aufwand.

## Nutzen

***Effizientere Geschäftsprozessabwicklung:*** *Performance-Verbesserungen an Client-Server-Applikationen helfen den Firmen-internen Anwendern beziehungsweise den B2B-Partnern, die auf diesen Systemen arbeiten. Damit wird eine direkte Effizienzsteigerung in der Abwicklung von Geschäftsprozessen erreicht.*

***Weniger Infrastrukturkosten für Clients:*** *Mit optimal funktionierenden Clients und Server können Upgrades der Arbeitsplatz-Hardware oder auch der Server-Infrastruktur in längeren Zyklen erfolgen.*

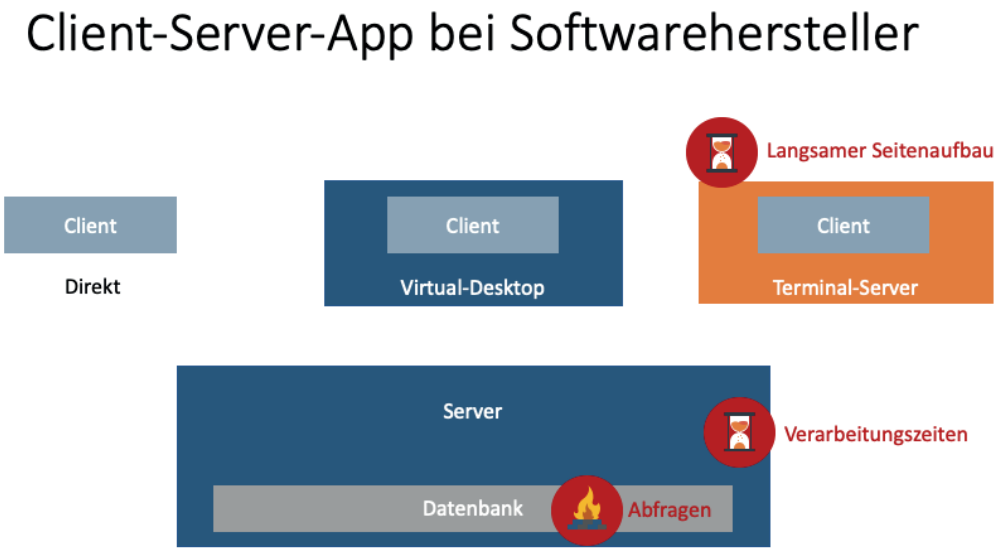


*Abbildung 3: Im gezeigten Beispiel geht es um ein ERP-System, das bei der Kundschaft verschieden genutzt wurde: Gewisse Kunden installierten den Client direkt auf dem Benutzersystem, andere Kunden verwendeten eine Virtual-Desktop-Infrastruktur, um auf den Client zu zugreifen und wieder andere verwendeten zu diesem Zweck Remote-Desktop. Im letzten Fall waren erhebliche Verzögerungen bei Interaktionen mit dem Client spürbar, weil die zentralen Terminal-Server zu wenig Grafikunterstützung boten. Bei der Analyse dieser Performance-Probleme tauchten weitere Mankos bei den Verarbeitungszeiten auf, die vor allem von aufwändigen Datenbankabfragen herrührten.*

***«FirmTec half uns bei der Verbesserung unserer Systemanforderungen, damit wir gegenüber unseren Kunden zuverlässiger auftreten können.»***
*Entwicklungsverantwortlicher eines ERP-Systems bei einem Softwarehersteller*

# Kernsystem-Lösungen

## Herausforderung

Kernsysteme, zum Beispiel ERP-Lösungen oder CRM-Systeme, sind die modernen Arbeitspferde. Heerscharen von Mitarbeitern bewirtschaften damit die zentralen Businessprozesse ihrer Unternehmung. Entsprechend gibt es abhängig von der Firmengrösse teilweise eine geringe bis hin zu einer enormen Anzahl Benutzer, welche sich häufig über den ganzen Tag gleichzeitig auf diesen Systemen bewegen. Das heisst, dass die Last und das zu verarbeitende Datenvolumen zentrale Herausforderungen sind, die ein Kernsystem beherrschen muss.

Weil Kernsysteme so zentral sind, werden sie auch immer weiter integriert. Entsprechend steigen die Transaktionszahlen und damit die Anforderungen an Performance und Verfügbarkeit. Dies kann Betriebsstabilisierungsstrategien mit regelmässigen Restarts verunmöglichen.

Parametrierungen ermöglichen die spezifische Erweiterung der Businessfunktionalität in einem konfigurierbaren Framework des Herstellers. Parametrierbare Standardsoftware birgt aber besondere Performance-Risiken. Diese konfigurativen Änderungen können selten explizit getestet werden. Damit sind unvorteilhafte Systemzugriffe mit teilweise gravierendem Performance-Impact vorprogrammiert.

## Lösung

Der zentrale Punkt im Performance-Engineering von Kernsystemen ist die Benutzerperspektive. Es gibt üblicherweise unzählige Aktionen, Verarbeitungen und Prozesse mit langen Laufzeiten. Wichtig ist, dass diejenigen identifiziert werden, welche Einschränkungen für die Anwender mit sich bringen.

In der Untersuchung des Verhaltens unter Last ist die Schwierigkeit, eine repräsentative Tagesbelastung auf dem System zu simulieren. Typischerweise sind Service-Endpunkte effektiver und effizienter zu bedienen als Benutzeroberflächen. Entsprechend empfiehlt sich ein verteilter Ansatz: Lastgenerierung auf Services und Vermessung des Clients (End-User-Experience) im belasteten Zustand mit einzelnen simulierten Benutzern (siehe Abbildung 4). Im Falle eines hohen Integrationsgrads sind aber auch die systemübergreifenden Businessprozesse relevant, weil erst in der End-to-End-Sicht, das Performance-Verhalten effektiv beurteilt werden kann.

Für den Betrieb eines Kernsystems ist die kontinuierliche und aktive Überwachung essentiell, weil jeder potentielle Ausfall mit erheblichen Kosten verbunden sein kann. In diesem Sinne ist das Antizipieren von Anomalien, um proaktiv zu handeln, sehr viel Wert. Mit der Überwachung können auch Abweichungen in Trends, zum Beispiel durch veränderte Parametrierungen und Konfigurationen festgestellt werden.

## Nutzen

***Versions- und Konfigurationsänderungen risikoreduziert einführen:*** *Umfassendes Performance-Engineering reduziert das Risiko für performance-bedingte Betriebsunterbrüche nach der Einführung signifikant.*


***Tiefe Fluktuationsrate bei Mitarbeitern:*** *Schnelle und stabile Kernsysteme führen zu zufriedenen Anwendern. Dies hat einen erwiesenen, positiven Einfluss auf die Fluktuationsraten und auf die Produktivität in einer Grossunternehmung.*

***Betriebskosten senken:*** *Ohne Performance-Engineering wird der stabile Betrieb von Kernsystemen oft über eine riesige Infrastruktur sichergestellt. Mit Performance-Engineering können die Betriebskosten auf ein optimales Niveau gebracht werden.*

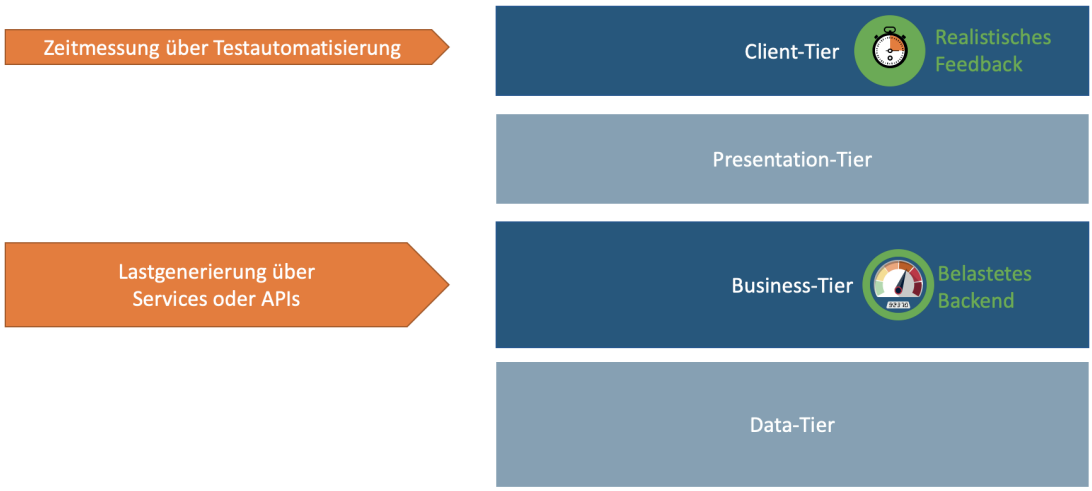


*Abbildung 4: Das Beispiel zeigt die Lastgenerierung für ein branchenspezifisches ERP-System. Schwierigkeit hier ist, dass die Bedienung des Clients aus dem Lasttests relativ aufwändig ist, weil er über eine proprietäre Schnittstellentechnologie mit dem Presentation-Tier kommuniziert. Entsprechend bleiben nur UI-Tests für die Bedienung des Clients. Diese wurden aber ausschliesslich für die Zeitmessung durch einige wenige simulierte Benutzer beschränkt. Die Hauptlast wird über Service-Schnittstellen des Business-Tiers generiert. In der Kombination lässt sich der Effekt der Last auf den Client sehr gut aufzeigen.*

***«Mit dem Ansatz von FirmTec haben wir die Performance und die Kosten von Releases im Griff»***
*Testverantwortlicher eines Outsourcing-Provider für ERP-Systeme*

# Integrationsplattformen

## Herausforderung

Integrationsplattformen sind das Rückgrat von IT-Landschaften in grossen Unternehmen. Entsprechend ist die Anzahl partizipierender Systeme und ausgetauschter Transaktionen hoch. Instabilitäten oder Performance-Engpässe im Integrationssystemen haben weitreichende Auswirkungen und behindern viele Mitarbeiter und zum Teil auch Kunden.

Je nach System werden verschiedene Integrationspattern verwendet: Zwei Hauptmuster sind die synchrone und asynchrone Kommunikation. Im synchronen Fall stehen mehr die zeitlichen Aspekte und die Ressourcenauslastung im Fokus. Bei asynchroner Kommunikation muss gewährleistet sein, dass im Fall einer hohen Last die entsprechenden Pufferkapazität vorhanden ist und damit keine Anomalien, wie Nachrichtenverluste, auftauchen.

Integrationsplattformen können auf kommerziellen Produkten aufgebaut sein. Entsprechend fehlen die Transparenz für den Quellcode und damit die Möglichkeiten für die Überwachung. Unabhängig davon, ob eine Integrationsplattform selber entwickelt oder eingekauft wird, muss sie die nichtfunktionalen Anforderungen der Unternehmung erfüllen. Dies kann bereits früh in der Evaluations- oder Einführungsphase getestet werden (Abbildung 5).

## Lösung

Integrationsplattformen können über die angebundenen Systeme oder auch direkt getestet werden. Im Fall von Endsystemen ergibt dies ein realistischeres Bild bezüglich den Businessprozessen. Im anderen Fall kann die Integrationsplattform für sich detailliert analysiert und optimiert werden.

Um Performanceaussagen zur Integrationsplattform zu verbessern, braucht es auch entsprechende Überwachungs- und Analysewerkzeuge. Für Businessprozesse sind diese mit Vorteil applikationsübergreifend aufgebaut. Transaktionen sollten dann über durchgehende Korrelations-IDs identifiziert werden können. Performance-Monitoring-Lösungen bieten hier entsprechende Unterstützung an. Von Eigenimplementierungen ist abzuraten, weil sich dies in Bezug auf Entwicklungs- und Wartungskosten nicht auszahlt.

## Nutzen

***Skalierbarkeit der Unternehmung steigern:*** *Robuste und zuverlässige Integrationsplattformen sind die Dreh- und Angelpunkte für die Skalierbarkeit der gesamten Unternehmung aus IT-Sicht.*

***Ein Nährboden für schnelle Projekte:*** *Wenn die Integrationsplattform für sich schon optimiert ist, können später Projekte mit Integrationsanforderungen viel schneller abgewickelt werden.*

***Umfassende Sicht auf Businessprozesse:*** *Ist die Integrationsplattform gut ins Monitoring eingebunden, ermöglicht dies eine umfassende Sicht auf diverse Businessprozesse und reduziert entsprechend den anschliessenden Analyseaufwand.*

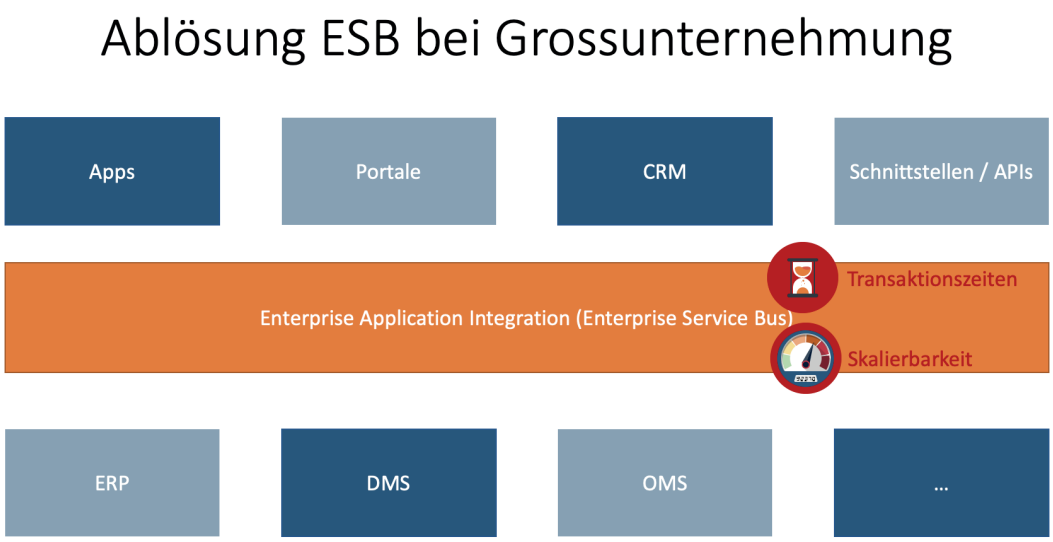


*Abbildung 5: Beim Neuaufbau oder Wechsel der Integrationsplattform ist zu gewährleisten, dass diese Plattform hinreichend skalierbar ist, um die aktuellen und zukünftigen Lastszenarien zu bewältigen. Weiter ist es essentiell, dass die Plattform selbst möglichst einen kleinen Overhead auf die Transaktionszeiten verursacht. Im konkreten Fall war die abzulösende Plattform durchaus anfällig bezüglich Engpässe in der Skalierbarkeit, was zu hängenden Transkationen führte. Darum wurden bei der Ablösung schon im Proof-of-Concept Performance-Untersuchungen durchgeführt.*

***«Beim Plattformwechsel hat uns FirmTec unkompliziert und zeitgerecht Entscheidungshilfen aus Performance-Sicht geliefert.»***
*Plattformverantwortlicher für Enterprise-Integration einer Grossunternehmung*

# E-Commerce-Systeme

## Herausforderung

E-Commerce-Systeme oder eben Verkaufslösungen werden im Normalfall direkt durch den Kunden bedient. Ziel ist es, dem Kunden möglichst viele Produkte möglichst attraktiv zu präsentieren, um zum Kaufen zu animieren und dadurch einen möglichst hohen Umsatz zu generieren.

Nirgends ist wohl die Konkurrenz direkter spürbar als im Online-Handel: Entsprechend negativ ist, wenn die Anzahl der Kunden auf dem System, die Menge an Produkten im Katalog oder das Design, zum Beispiel die Qualität der Bilder, einen Einfluss auf die Geschwindigkeit oder die Stabilität des Systems haben. Diese Effekte werden von grossen Anbietern schon lange untersucht und quantifiziert. Verschiedene Untersuchungen haben ermittelt, dass Verbesserung der Antwortzeiten der Webseiten im 100-Millisekunden-Bereich eine Steigerung des Umsatzes im Prozentbereich bewirken.

Neben vielen Bildern, Videos und weiteren verkaufssteigernden Effekten gibt es einen weiteren Performance-Bremse: In modernen E-Commerce-Systemen wird nämlich der Kunde genau überwacht, damit er später möglichst spezifisch beworben werden kann. Dies führt zu einem immer schwierigeren Spagat zwischen Attraktivität, aufwändigem Design und umfangreichen Analysemöglichkeiten auf der einen Seite und der optimalen Performance und Effizienz auf der anderen Seite (siehe Abbildung 6).

## Lösung

Die solide Basis eines E-Commerce-System ist das Backend: Daten müssen im Nu an Clients geschickt werden können. Suchergebnisse müssen sofort und unabhängig von der Benutzermengen zur Verfügung stehen. Dies ist mit Performance-Tests über die üblichen Webschnittstellen hinlänglich zu prüfen.

Die obenerwähnten Performance-Fresser liegen aber immer mehr auf Frontendseite. Folglich muss im E-Commerce-Systemen nicht nur das Backend, wie bei allen Webapplikationen untersucht werden, sondern immer mehr auch die Frontend-Performance. Auch dafür gibt es spezialisierte Werkzeuge, die Analysen der verwendeten Webseitenbestandteile, des Seitenaufbaus oder der Inhalte ermöglichen. Mittels dieser Werkzeuge können direkt Empfehlungen abgeleitet werden.

Am Schluss zählt nur das stabile und schnelle Funktionieren des Gesamtsystems: Entsprechend müssen kombinierte Tests in einer entsprechenden realistischen Testumgebung durchgeführt und die produktiven Systeme lückenlos überwacht werden, um bei etwaigen Problemen sofort eingreifen zu können.

## Nutzen

***Risikoreduktion:*** *Die Einführung eines verteilten Systems ist üblicherweise komplex und risikobehaftet. Performance-Engineering reduziert die Risiken für Stabilitätsprobleme und das Nichterreichen der erwarteten Verarbeitungsmengen oder -Zeiten nachhaltig.*

***Wirtschaftlichkeit:*** *Verteilte Systeme werden meist mit einer längeren Kostenperspektive aufgebaut, entsprechend müssen sie auch wirtschaftlich betrieben werden können. Dies gilt für die eingesetzte Hardware und Lizenzen, aber auch für allfällige betriebsrelevante Eingriffe, die kumuliert viel kosten.*

***Neutrale Performance-Beurteilung:*** *Wenn verteilte Systeme firmenübergreifend aufgebaut werden, ist die Performance entsprechend firmenübergreifend sicherzustellen, was schnell zu einem Schwarz-Peter-Spiel führt. Mit einem unabhängigen Performance-Engineering kann dies unterbunden werden.*

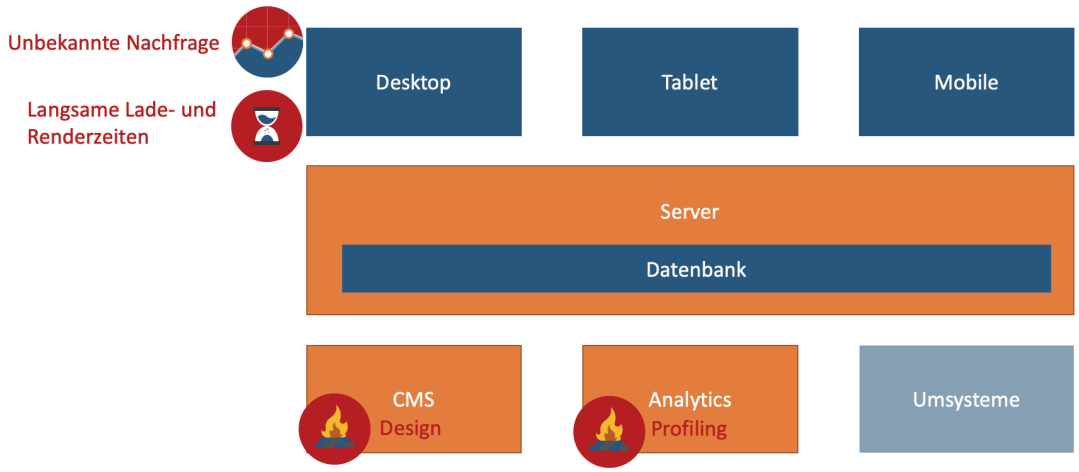


*Abbildung 6: Im konkreten Fall war das E-Commerce-System eines führenden Bekleidungsunternehmen in Europa messbar schlechter als die Konkurrenz: Antwortzeiten, insbesondere Lade- und Seitenaufbauzeiten waren langsamer als bei den Mitbewerbern. Gleichzeitig konnten nur ansatzweise Aussagen zur Auslastung des Shops gemacht werden, obwohl stark auf Analytics mit verschiedensten Werkzeugen gesetzt wurde. Effektiv waren die verschiedenen Analytics-Werkzeuge, die zur Profilierung der Kundschaft genutzt wurden, ein erheblicher Teil des Problems. Die Hauptursache lag aber in den überrissenen Erwartungen an Ästhetik und Design der Oberfläche, die alle Performance-Anforderungen in den Hintergrund drängten.*

***«Mit der Unterstützung von FirmTec ist unser Webshop der Konkurrenz davongezogen!»***
*Projektleiter – Redesign Webshop eines Detailhändlers*

# Verteilte Systeme

## Herausforderung

Immer mehr Business-to-Business-Prozesse, aber auch unternehmensinterne Prozesse werden über diverse Applikationen, sogenannte Services verteilt. Eine Schwierigkeit sind die verschiedenen Service-Owner beziehungsweise Applikationsverantwortlichen, die mit divergierenden Interessen für ihren jeweiligen Service einstehen. In vielen Fällen fehlt es an klaren Anforderungen an die übergeordneten Prozesse und entsprechend an Anforderungen an die involvierten Services oder Applikationen.

Aus technologischer Sicht ist die Heterogenität eine grosse Herausforderung: Legacy-Technologien, eingekaufte Softwareprodukte respektive Services und Eigenentwicklungen können bunt gemischt sein. Eine Schwierigkeit ist ein solch heterogenes System zu überwachen.

Für die Innovation sind hier meist grössere Vorhaben wie komplette IT-Programme am Werk. Darum gibt es auch organisatorisch viele Abhängigkeiten. Oft gibt es kaum eine Phase, wo das Gesamtsystem getestet werden kann. Abbildung 7 zeigt ein Beispielprojekt aus dem Bankenumfeld, in welchem mit Outsourcing gearbeitet wurde.

## Lösung

In einem ersten Schritt müssen Erwartungen oder Anforderungen an die (Kern-)Prozesse formuliert werden. Dabei geht es aus Sicht der Performance vor allem um nichtfunktionale Anforderungen: Wichtige Metriken sind Verarbeitungszeiten und das erwartete Volumen. Allenfalls muss dies in verschiedenen Produkt- oder Serviceklassen definiert werden.

Für die Kernprozesse sind entsprechende Lasttreiber zu erstellen. Um aber effektiv vorwärtszukommen, hilft es, das System in kleinere Teilsysteme zu zerlegen und diese zuerst separat zu testen. Zur Aufhebung von Abhängigkeiten können Simulationsansätze (zum Beispiel Service-Virtualisierung) wertvolle Dienste leisten. Nach den Tests der individuellen Systeme braucht es unbedingt Gesamtsystemtests, um das verteilte System erfolgreich einzuführen.

Für alle Arten von Test braucht es ein systematisches Monitoring. Klar definierte Messpunkte müssen Informationen möglichst in ein zentrales System einliefern, woraus dann Rückschlüsse zu Verarbeitungszeiten und -Volumen erfolgen. Die Log- und Monitoring-Lösungen sind selber auf die bevorstehenden Lastszenarien zu prüfen. Lösungen, welche vor der Einführung aufgebaut werden, sind in die Betriebsorganisation zu überführen, um hier möglichst von Anfang an eine klare Sicht aufs System zu erhalten.

## Nutzen

***Risikoreduktion:*** *Die Einführung eines verteilten Systems ist üblicherweise komplex und risikobehaftet. Performance-Engineering reduziert die Risiken für Stabilitätsprobleme und das Nichterreichen der erwarteten Verarbeitungsmengen oder -Zeiten nachhaltig.*

***Wirtschaftlichkeit:*** *Verteilte Systeme werden meist mit einer längeren Kostenperspektive aufgebaut, entsprechend müssen sie auch wirtschaftlich betrieben werden können. Dies gilt für die eingesetzte Hardware und Lizenzen, aber auch für allfällige betriebsrelevante Eingriffe, die kumuliert viel kosten.*

***Neutrale Performance-Beurteilung:*** *Wenn verteilte Systeme firmenübergreifend aufgebaut werden, ist die Performance entsprechend firmenübergreifend sicherzustellen, was schnell zu einem Schwarz-Peter-Spiel führt. Mit einem unabhängigen Performance-Engineering kann dies unterbunden werden.*

# Verarbeitungssystem bei Grossbank

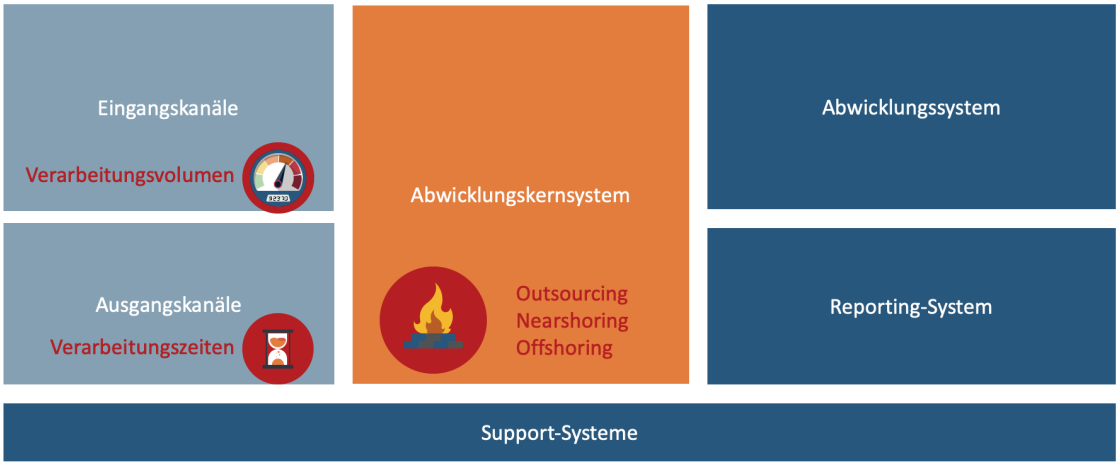


*Abbildung 7: Das Beispiel zeigt ein verteiltes System aus dem Bankenumfeld. Hierbei ging es um Outsourcing eines zentralen Businessprozess. Herausfordernd war, dass nur der halbe Prozess externalisiert wurde und die andere Hälfte weiterhin intern lief. Folglich war gute Kommunikation zwischen den Parteien unerlässlich. In allen Fällen musste das geforderte Verarbeitungsvolumen in einer definierten Zeit abgewickelt werden können. Später kam als weitere Anforderung dazu, dass gewisse Produkte auch schnelle Verarbeitungszeiten aus End-to-End-Sicht erfüllen müssen. Im Performance-Engineering wurde dieses System Schritt für Schritt untersucht – Infrastruktur, Teilsysteme und später das Gesamtsystem.*

***«Das System läuft ab dem ersten Betriebstag stabil und schnell! – Es brauchte keine Taskforcemeetings der Betriebsorganisation»***
*Servicemanager der Gesamtlösung einer Grossbank*

# Cloud-Applikationen

## Herausforderung

Die grosse Herausforderung im Cloudumfeld ist die Hyperskalierung. Cloud-Technologie besitzt die Eigenschaft, dass sie bedarfsgestützt mehr oder weniger Ressourcen einsetzen kann. Im besten Fall geschieht diese Skalierung voll automatisch und elastisch. Um von diesen Vorteilen profitieren zu können, muss die Software entsprechend skalierbar sein, ansonsten entstehen hier nur Kosten durch aufwändigere Infrastrukturservices ohne Nutzen beim Anwender.

Es gibt diverse Arten von Cloud-Applikation. Ein wichtiges Unterscheidungskriterium ist der Entwicklungsansatz: Eine Cloud-Native-Applikation wird durch die Integration verschiedener Cloud-Services erzeugt. Im Gegensatz dazu gibt es konventionell entwickelte Software, die auf eine Cloudtechnologie migriert wird. Im Falle einer Cloud-Native-Applikation ist zu erwarten, dass die verwendeten Cloud-Services eine hinreichend gute Skalierung bringen und somit nicht zwingend Performance-Test vorzunehmen sind. Im Falle von konventionell-entwickelter Software ist dies jedoch unbedingt zu prüfen.

Im Kontext von Unternehmen ist das gemischte Workload-Szenario von Off- und On-Premises-Applikationen oder -Services komplex und teilweise risikobehaftet, speziell wenn der übergreifende Datenzugriff eine höhere Latenzzeit mit sich bringt.

## Lösung

Cloud-Applikationen können analog zu traditionellen On-Premises-Applikationen mit Performance-Engineering untersucht werden. Unterschiedlich ist das elastische Skalieren, was speziell untersucht werden muss. Weiter ist zu berücksichtigen, dass künstliche Kapazitätsgrenzen aus beispielsweise Kostenüberlegungen zu unerwarteten Performance-Engpässen führen können. Diese müssen hinreichend gut untersucht und verstanden werden. Auch die Effekte vom automatischen zuschalten weiterer Ressourcen sollten im Testsystem ausgiebig untersucht werden, um Schwellenwerte sinnvoll und ökonomisch zu definieren.

Toolmässig bietet die Cloud viele Werkzeuge, die in der traditionellen Welt separat aufgebaut werden müssen. Das betrifft beispielsweise Monitoring-Services aber auch, Bausteine für automatisierte Delivery-Pipelines oder die Lastgenerierung.

Um Probleme mit langen Wartezeiten wegen Datenzugriffen zu reduzieren, können Caches an sinnvollen Stellen aufgebaut werden. Dabei werden Daten dupliziert und zum Teil sogar zwischen den verschiedenen Frontend-Services redundant gehalten (Abbildung 8).

## Nutzen

***Die hohe Skalierbarkeit von Cloud effektiv nutzen:*** *Cloud ist per Default hochskalierbar. Am Schluss müssen aber die auf der Cloud laufenden Applikationen skalierbar sein, um einen Nutzen davon zu haben. Performance-Engineering bringt genau diese Skalierbarkeit.*

***Keine Überraschungen durch Kostentransparenz:*** *Cloud schafft eine hohe Kostentransparenz auch für Infrastruktur und Basis-Services. Mit Performance-Engineering können unangenehme Überraschungen beim Umstieg auf die Cloud verhindert werden.*

***Minimierung von Lastpeaks:*** *Mit simulierten Lastszenarien aus dem Performance-Engineering kann das Lastvolumen auf der Cloud optimal gesteuert und damit Peak-Kosten vermieden werden.*

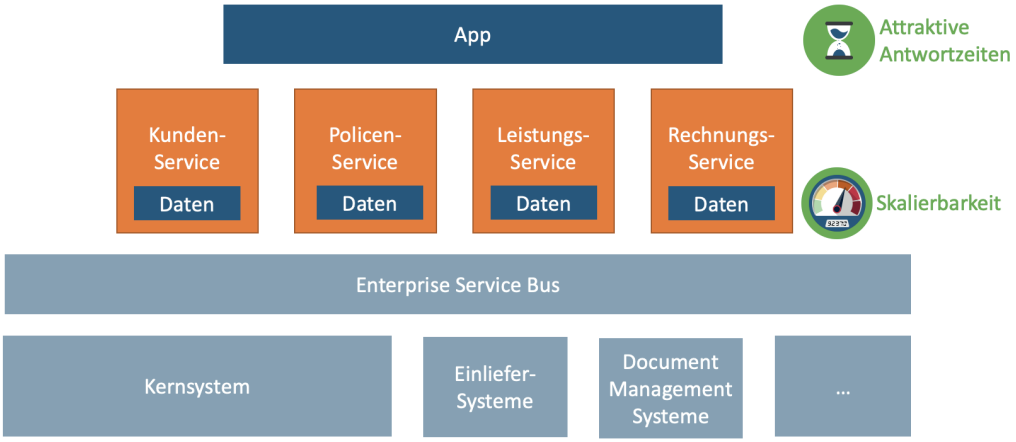


*Abbildung 8: Als Teilergebnis des Performance-Engineerings wurde der Aufbau eines skalierbaren Backends für die Apps erreicht. Mit der Skalierbarkeit wurde die Möglichkeit für einen hohen Profit von Cloud-Technologien geschaffen. Für den Benutzer entstanden zusätzlich attraktive Antwortzeiten, weil die Daten neu in Frontend-nahen Services gecacht werden.*

***«Dank einer hinreichenden Skalierbarkeit können wir voll von den Vorzügen einer elastischen Cloud profitieren.»***
*Applikationsverantwortlicher eines Gesundheitsversicherers*

# Blockchain-Applikationen

## Herausforderung

Applikationen, die auf Blockchain-Technologie aufbauen, leiden derzeit noch unter dem Fakt, dass Blockchain-Transaktionen per se sehr zeitaufwändig sind. Entsprechend sind die Applikationen so zu entwerfen, dass die Benutzerinteraktionen nach Möglichkeit nicht durch die Blockchain-Transaktionen gebremst werden.

Es besteht das Risiko, dass wegen der Blockchain als limitierender Prozess, die Aufmerksamkeit für andere Performance-Risiken nachlässt (Abbildung 9).

## Lösung

Unterschiedliche Blockchain-Technologien und Versionen können ein massiv unterschiedliches Performance-Verhalten anbieten. Entsprechend kann in Zusammenarbeit mit Performance-Engineering die Möglichkeiten der ausgewählten Blockchain-Technologie untersucht werden und allenfalls mit Alternativen faktenbasiert verglichen werden.

Das Verhalten der Gesamtapplikation muss geprüft werden, um nicht zusätzliche Performance-Risiken einzugehen. In diesem Zusammenhang kann auch das Verhalten der übrigen Applikation im Zusammenspiel mit der Blockchain optimiert werden.

Grundsätzlich hat die Blockchain-Technologie als Ganzes Potential für einen Performance-Boost. Hier ist aber aufgrund der Kritikalität der Sicherheit Vorsicht geboten.

## Nutzen

***Die optimale Blockchain-Technologie finden**: Im Blockchain-Umfeld ist die Technologie entscheidend bezüglich Energieeffizienz. Mit Performance-Engineering kann hier beim Betrieb von Test- und Produktivsystemen erheblich gespart werden.*

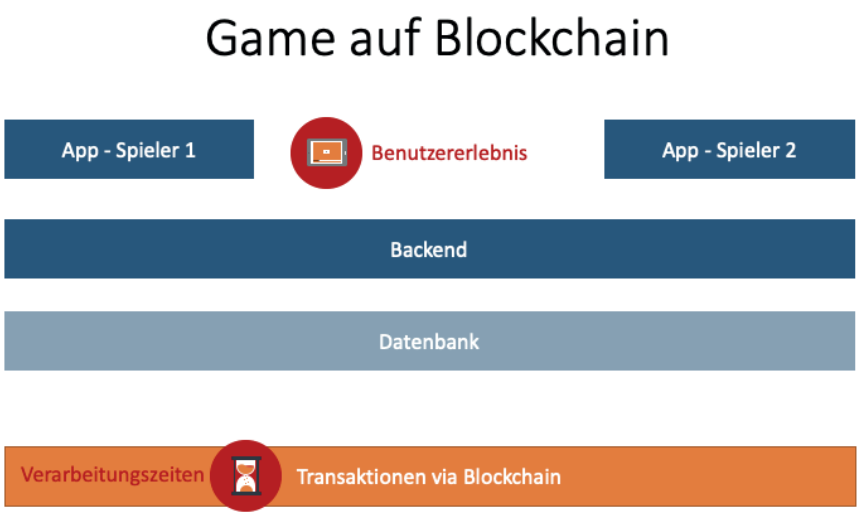


*Abbildung 9: Im Umfeld eines Startups wurde ein Spiel, das den Austausch von Bildern analog zu Fussball-Sticker ermöglicht, auf der Blockchain entwickelt. Die Transaktionszeiten sind klar der limitierende Faktor auf den Prozess. Mit Performance-Engineering konnte hier die für den Usecase optimale Blockchain-Technologie ausgewählt werden. Wichtig war auch, dass die übrigen Applikationskomponenten eine hinreichende Performance bringen, um gegenüber dem Spieler bei den Nicht-Blockchain-Transaktionen eine zufriedenstellende Antwortzeit anzubieten.*

***«FirmTec hat uns als Startup im Performance-Engineering ein erfolgreiches Abheben ermöglicht.»***
*Blockchain-Enthusiast und Firmengründer des Startups*

# Performance-Engineering in der Praxis

## Performance-Testing

### Last- und Performance-Tests

Performance-Testing hat eine lange Tradition. Teilweise wird auch von Last- und Performance-Testing gesprochen, um darauf aufmerksam zu machen, dass sowohl die Stabilität unter Last wie auch die Performanz, das heisst die Geschwindigkeit, untersucht werden. Hier werden die Begriffe Lasttest und Performance-Test synonym verwendet, weil sich beide um die Geschwindigkeit, Stabilität und Skalierbarkeit von Applikationen kümmern.

### Performance-Tests alleine reichen nicht

Typischerweise werden in einem Performance-Tests nichtfunktionale Anforderungen geprüft. Im schlimmsten Fall bleibt es bei der Auswertung der Anforderungen in dem Sinne, dass ein Bericht zu den erfüllten und nicht erfüllten Anforderungen, allenfalls mit Defects ergänzt, erstellt wird. Die Sicht ist zu wenig aussagekräftig, speziell wenn im Defect nur auf die Differenz zwischen Soll und Ist hingewiesen wird, ohne detailliertere Analyse der inneren Ursache.

### Funktionale und nichtfunktionale Tests

Im klassischen Verständnis ist das Prüfen von nichtfunktionalen Anforderungen immer zweitrangig gegenüber dem Testen der Funktionalität. Entsprechend dürfen in vielen Firmen Performance-Tests erst nach dem Abschluss der funktionalen Tests, zu Randzeiten oder am Wochenende durchgeführt werden. In der Realität ist es aber so, dass eine Applikation mit einer schlechten Performance meist kaum mehr zu bedienen oder sogar unbrauchbar wird. Darum ist die Bedeutung der nichtfunktionalen Tests zwingend zu erhöhen.

### Verschiedene Testarten

Im Sinne eines modernen Performance-Engineerings sollten diverse Performance-Tests mit verschiedenen Testzielen stattfinden: Damit wird ermittelt, wo Möglichkeiten und Grenzen der Infrastrukturkomponenten oder Plattformservices liegen. Dies kann unabhängig von den laufenden Projekten untersucht werden. Weiter können Performance-Tests für Teilsysteme frühzeitige Ansätze für Verbesserungen des Gesamtsystems bringen. Tests des Gesamtsystems sind zur Vervollständigung des Bildes wichtig, sollten aber nicht das einzige Mittel sein.

### Lastprofile

In allen Tests (Infrastruktur / Plattform, Teilsystem, Gesamtsystem) geht es hauptsächlich um die Quantifizierung des Performance-Verhalten, also eher um eine Symptomsicht. Dazu werden ein oder mehrere definierte Funktionalitäten untersucht. Dabei werden die Anfragen an das System unter Test normalerweise parallelisiert. Die Lastkurve über der Zeit wird Lastprofil genannt. Die Hauptansprüche an das Lastprofil sind eine hohe Aussagekraft und die Wiederholbarkeit, darum werden häufig modellhafte Lastprofile statt einer realistischen Belastung eingesetzt.

### Wartezeiten (auch Think-Times) und Pacing

Die Last kann auf verschiedene Weisen gesteigert werden: Wie oben beschrieben über Parallelisierung, aber auch indem Wartezeiten (auch englisch «Think-Times») zwischen den einzelnen Anfragen verkürzt werden. In gewissen Tools kann die Durchlaufzeit pro funktionalen Ablauf definiert werden. Dies wird üblicherweise Pacing genannt.

### Testdaten

Damit die Applikation unter Test aus der Reserve gelockt werden kann und nicht ständig aus dem Cache antwortet, werden üblicherweise Eingabewerte oder Suchargumente als Parameter gesetzt und entsprechend variiert. Die Variierung kann in einer definierten Reihenfolge oder auch zufällig passieren. Falls der Testablauf Daten verändert oder sogar löscht braucht es entsprechende Mengen von Testdaten. Im besten Fall werden Möglichkeiten geschaffen, wie Daten bereinigt beziehungsweise zurückgesetzt werden können, um den Test zu einem späteren Zeitpunkt unter möglichst gleichen Bedingungen zu wiederholen. Im Sinne der Wiederholbarkeit sind Zufallsgeneratoren, die für die Randomisierung von Parametern, Eingabewerten oder Suchargumenten verwendet

werden, mit Vorsicht einzusetzen. Sie steigern teilweise den Realismus und verbessern Nebeneffekte durch Caching, können aber auch zu schwer interpretierbaren oder nicht wiederholbaren Ergebnissen führen.

### Sicherheitsmechanismen

Applikationen haben immer mehr Sicherheitsmechanismen. Vom einfachen Zugriffsschutz mit Username und Passwort, über Security-Tokens bis hin zu komplexen Mehrfaktor-Zugriffsverfahren sind diverse Ansätze im Einsatz. Zur automatisierten Simulation einer entsprechenden Userlast muss mit den entsprechenden Sicherheitsmechansimen umgegangen werden können. Im Fall von Username-Passwort reicht es, wenn hinreichend viele Test-Accounts zur Verfügung stehen. Ein Zweifaktor-Mechanismus mit Username-Passwort und mTAN kann mit einem Set von virtuellen Telefonnummern wie echt getestet werden, weil nur so gewährleistet ist, dass das TAN eindeutig zugeordnet werden kann. Schwieriger wird es bei moderneren Verfahren wie PhotoTAN oder Captchas: Im schlimmsten Fall muss hier der Schutz der Testumgebung reduziert werden, indem beispielsweise ein Standard-Wert für das TAN beziehungsweise Captcha definiert wird, der dann vom simulierten Benutzer verwendet werden kann. Dabei wird allerdings das TAN noch realitätsnahe generiert, was für Performance-Messungen relevant ist. Wenn Zertifikate im Spiel sind, zum Beispiel bei Single-Sign-On braucht es auch eine entsprechende Menge an Zertifikaten, damit verschiedene User simuliert werden können.

# Application-Performance-Management

## Punktgenaue Diagnosen

In den allermeisten Fällen reicht die reine Symptomsicht nicht: Verantwortliche interessieren sich für die inneren Ursachen und erwarten eine möglichst punktgenaue Diagnose der Ursache eines Performance-Engpasses und eine adäquate Therapie, um das Problem zu beheben. Application-Performance-Management bietet hierzu Mittel, um Performance zu überwachen (Performance-Monitoring) sowie auch für unterschiedliche Performance-Analysen.

## Logfile-Analyse

Immer mehr Firmen setzen auf Logfile-Analyse und Data-Mining, wenn es um Performance-Monitoring geht. Der Ansatz ist, dass alle möglichen Systeme ihre Logs in ein zentrales Repository schreiben. Aus diesen können übergreifende Analysen erfolgen. Logs können für Performance-Analysen relevante Informationen wie Zeitstempel, Laufzeiten oder Systemzustände enthalten. Die Schwierigkeit ist allerdings, die Logs ohne Kenntnisse des Codes korrekt zu interpretieren. Darum ist Logfile-Analyse ein Blackbox-Ansatz, der eigentlich eine präzisere Symptom-Beschreibung ermöglicht, aber für die Diagnosen nicht in allen Fällen zielführend ist. Zudem werden unter Umständen nicht alle Probleme in den Logfiles gespeichert, was die Analyse von inneren Applikationsproblemen erschwert.

## Profiling von Code

Ein klassisches Mittel für Performance-Analysen ist die Laufzeitanalyse von Software, kurz Profiling genannt. Dabei werden die Ausführungszeiten der Methoden minutiös protokolliert. Langläufer oder häufig wiederkehrende Aufrufe können einfach ermittelt werden. Profiling ist eine sehr effektive Methode für Software-Entwickler, um ihren eigenen Code zu verbessern. Dies ist der sogenannte Whitebox Ansatz, weil alle Details zum Code sichtbar werden. Für den Performance-Engineer kann dies aber limitierend sein, weil viel zu viele Informationen vorhanden sind. Bei ungünstig konfiguriertem Profiling kann zudem ein System, insbesondere eine produktive Applikation, starke Performance Einbussen erleiden.

## Verschiedene Ebenen

Performance-Engpässe tauchen auf verschiedenen Ebenen auf. Zum Beispiel bestehen Ressourcen-Limitationen in der Infrastruktur (Memory, CPU, Storage, Netzwerk), die meist einen spürbaren Effekt auf die Applikationen haben. Es gibt aber auch Einschränkungen auf den Plattformsystemen: Ein ausgeschöpfter Pool im Betriebssystem oder dem Applikationsserver, sowie überlastete Datenbankverbindungen. Viele Performance-Engpässe entstehen aber effektiv auf der Softwareebene: ungünstig formulierte Datenbankabfragen oder umständliche Aufrufhierarchien. Um Performance-Probleme schnell zu diagnostizieren braucht es also Analyse-Möglichkeiten auf allen Ebenen. Dies in einer möglichst durchgehenden Sicht, um den Effekt der Datenbankabfrage auf die Verbindung in Zusammenspiel mit der Auslastung des Arbeitsspeichers zu beurteilen.

## Businessprozesssicht – Korrelation

Application-Performance-Management kümmert sich nicht einfach um das Auswerten von einzelnen Metriken oder die Verfügbarkeit von einzelnen Applikationen. Application-Performance-Management muss sich um die Überwachung und Verbesserung von übergreifenden Businessprozessen kümmern. Die Schwierigkeit ist insbesondere der übergreifende Aspekt, weil Businessprozesse selten nur auf einer Applikation ablaufen. Erschwerend sind in den meisten Applikationslandschaften von grösseren Unternehmungen verschiedenste Technologien und Produkte im Spiel. Entsprechend sind die Transaktionen über verschiedene Applikationen und Technologien nachzuverfolgen. Dazu werden häufig sogenannte Korrelations-IDs vergeben, die dann von System zu System weitergereicht werden. Moderne Application-Performance-Management Werkzeuge bieten die Möglichkeit sogenannte Businesstransaktionen entweder manuell anhand bestimmter Kriterien zu konfigurieren oder sogar automatisch zu erkennen.

## Aktives Monitoring: Synthetisches Transaktionsmonitoring und Anomalie-Detection

Seit jeher setzt sich Performance-Engineering für eine Verbesserung der Geschwindigkeit, Stabilität und Skalierbarkeit für die Benutzer ein. Entsprechend steht auch die Benutzersicht im Zentrum des Performance-Monitoring. Mit synthetischen Aufrufen kann aktiv festgestellt werden, ob sich das Performance-Verhalten einer

Applikation für den Benutzer verändert bzw. verschlechtert. Im Monitoring (wörtlich Überwachung) geht es darum, zeitnah Abweichungen festzustellen und anschliessend die Verantwortlichen zu alarmieren. Immer mehr geht es hier um Anomalie-Detection: Dabei geht es ums Erkennen von statistischen Mustern, die auf ein anormales Verhalten hinweisen. Hierfür kommen Strategien von Machine-Learning und Big-Data zum Einsatz.

## Reduktion des Datenvolumen

Um im Monitoring Anomalien festzustellen, braucht es hinreichend viele Daten. Dazu helfen beispielsweise historische Daten. Das kann aber je nach Datenvolumen, das gespeichert wird, zu sehr viel Aufwand führen. Im Performance-Monitoring-Umfeld ist eine verbreitete Strategie Daten zu aggregieren, also die Datenspeicherung zu verdichten. Das heisst, dass vorhandene Werte über einen gewissen Zeitraum gemittelt werden und nur der Mittelwert längerfristig gespeichert wird. Typischerweise gibt es verschiedene Granularitäten der Aggregation: Ältere Daten sind stärker verdichtet als aktuellere Daten und die aktuellsten Daten sind überhaupt nicht aggregiert.

Alternativ werden Daten nur nach gewissen Kriterien geloggt, aufgezeichnet oder gespeichert. Im Falle von Logs gibt es sogenannte Log-Levels, die je nach Bedarf eingestellt werden können. Zu Entwicklungszwecken existiert der Loglevel «Debug». Hier werden besonders viele Daten geschrieben, was im Normalfall Einbussen auf die Performance und hohe Anforderungen an den Datenspeicher zur Folge hat, jedoch detaillierte Informationen liefert. Gewisse Analyse-Systeme entscheiden abhängig von Kriterien, was zu speichern ist und was nicht: Zum Beispiel, auffällig langsame Transaktionen. Der Nachteil hier kann sein, dass die Sicht auf durchschnittliche oder gute Transaktionen fehlt.

Beim Performance-Monitoring ist der Umgang mit Daten immer ein schmaler Grat zwischen besserer Einsicht und negativer Beeinträchtigung der Performance der Systeme. Es ist situativ das richtige Setup zu ermitteln und mit den Verantwortlichen zu konfigurieren.

# Performance-Engineering im Agilen Umfeld

## Agiles Manifest

- *«Individuen und Interaktionen* mehr als Prozesse und Werkzeuge»: Performance-Engineering benutzt zwar viele Werkzeuge, erfolgreiche Performance-Engineers zeichnen sich dennoch primär durch die gekonnte Kommunikation mit den Stakeholdern aus. Denn nur durch Individuen und Interaktionen können Verbesserungen effektiv umgesetzt werden.
- *«Funktionierende Software* mehr als umfassende Dokumentation»: Performance-Engineering kümmert sich um die nichtfunktionale Basis, auf der die Software dann erfolgreich funktionieren kann. Ein einfaches Dokumentieren des Performance-Verhaltens ist in keinem Fall ausreichend oder zielführend (siehe Performance-Testing): Die Systeme sind zeitnah zu verbessern.
- *«Zusammenarbeit mit dem Kunden* mehr als Vertragsverhandlung»: Im Zentrum von Performance-Engineering steht immer der Kunde. Mit dieser Optik werden Performance-Engineers zu Schlüsselpersonen für Product-Owner, wenn es um die nachhaltige Verbesserung eines Software-Produktes geht.
- *«Reagieren auf Veränderung* mehr als das Befolgen eines Plans»: Performance-Engineering zeichnet sich als lose Sammlung von Aktivitäten auf verschiedenen Ebenen aus. Es geht um die Balance zwischen Agieren und Reagieren: Entsprechend bietet Performance-Engineering die maximale Flexibilität, um auf Veränderungen bezüglich Geschwindigkeit, Stabilität und Skalierbarkeit einzugehen.

## Performance-Engineering ist DevOps

Performance-Engineering unterstützt den Software-Entwicklungsprozess, indem so früh wie möglich Feedback zu nichtfunktionalen Anforderungen und der effektiven Umsetzung geliefert wird. Gleichzeitig werden mit den Verbesserungen an der Software und Hilfsmitteln wie Performance-Monitoring auch betriebliche Risiken massiv reduziert. Performance-Engineering muss zwingend die Produktsicht über den ganzen Software-Lifecycle haben. Dazu ist der Performance-Engineer meist auch mit den Verantwortlichen von Fach, Projekt, Entwicklung und Betrieb im Kontakt.

## Continuous-Performance-Testing

Das agile Umfeld zeichnet sich durch iterative Vorgehensmodelle aus. Dabei werden Software-Produkte inkrementell weiterentwickelt. Es muss stets sichergestellt werden, dass die Erweiterungen keinen nachteiligen Effekt auf frühere Funktionalitäten haben. Das gilt auch im nichtfunktionalen Bereich, also fürs Performance-Verhalten. Entsprechend empfiehlt sich ein Continuous-Performance-Testing: Dabei werden relevante Business-Funktionalitäten laufend getestet. Allerdings wird nicht das absolute Ergebnis bewertet, sondern es werden Trendlinien aufgenommen. Abweichungen von der Norm können rasch festgestellt und einfach zur inkrementellen Code-Änderung zurückverfolgt werden.

# Performance-Engineering-Governance

Eine Governance für Performance-Engineering hat drei Grundlagen (siehe Abbildung 10):

1. Die IT-Governance der Unternehmung, worauf die Performance-Governance abgestützt sein muss
2. Ein starkes Alignement zwischen Fach und IT, welches für eine hinreichend gute Performance absolut zentral ist
3. Nichtfunktionale Anforderungen, welche als fachliche und technische Treiber fürs Thema Performance-Engineering fungieren.

Die Performance-Engineering-Governance selbst betrifft dann die folgenden sechs Handlungsfelder: Organisation, Prozesse, Kultur, Mitarbeiter, Infrastruktur und Informationen. Das Ergebnis einer konsequent umgesetzten Governance für IT-Organisationen sind minimierte IT Risiken, eine sichtlich höhere Benutzerzufriedenheit und optimale Betriebskosten.

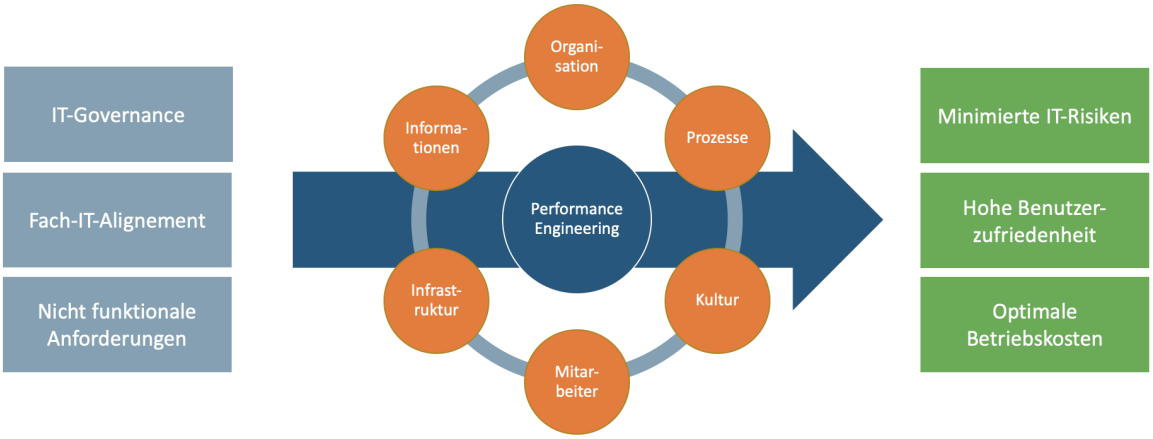


*Abbildung 10: Performance-Engineering Governance, die zu minimalen IT-Risiken, hoher Benutzerzufriedenheit und optimalen Betriebskosten führt.*

## Organisation

Performance-Engineering muss in der Unternehmung in einer geeigneten Form organisiert werden. Dies kann als zentrales Dienstleistungscenter im Sinne eines Teams oder Subteams aufgebaut werden oder auch als Circle of Excellence, wo die mit Performance assoziierten Engineers, lose miteinander zusammenarbeiten. Es muss klar sein, wer thematisch den Lead hat und wer der Ansprechpartner für Projekte und Produkte ist.

## Prozesse

Ein hoher Standardisierungsgrad ist die Grundlage für Effizienzsteigerungen und Automatisierung. Entsprechend sollen auch im Performance-Engineering-Umfeld die Prozesse klar geregelt sein. Dies gilt für Performance-Testing wie auch für Application-Performance-Management. Spezielle Beachtung bedürfen die Prozeduren für die Behebung von Performance-Incidents: Wer stellt fest, reagiert, teilt auf, analysiert und löst.

## Kultur

Performance und das Streben nach höherer Geschwindigkeit und Stabilität soll zu einem Kulturgut werden. Dazu gehört auch ein Umdenken bezüglich Verfügbarkeit von Applikationen. Schliesslich kann es nicht sein, dass die

Verfügbarkeit für Endkunden-Applikationen, nur zu Bürozeiten ausgewertet oder gewährleistet wird. Der Anspruch der Kundschaft geht immer mehr Richtung 7 x 24.

### Mitarbeiter

Performance-Engineering braucht entsprechende Mitarbeiter, welche die richtigen Skills mitbringen. Ebenso brauchen die eingesetzten Werkzeuge und Technologien Mitarbeiter, die sich darum kümmern können. Ein Punkt, bei welchem das Performance-Engineering in vielen Unternehmungen Lücken hat, ist die Stellvertretung insbesondere für die teilweise eher exotischen oder selbstentwickelten Werkzeuge. Standardisierung kann hier helfen.

### Infrastruktur

Fürs Performance-Testing und fürs Application-Performance-Management braucht es eine entsprechende Infrastruktur. Abgestimmt auf die Technologien, Usecases und Einsatzszenarien müssen Werkzeuge evaluiert und beschafft werden. Insbesondere im Application-Performance-Management-Umfeld muss das Werkzeug eingerichtet und längerfristig gewartet werden. Für einen möglichst grossen Nutzen sind erweiterbare Lösungen einfachen, konfigurierbaren Werkzeugen vorzuziehen. Weiter sollten die Erweiterungsartefakte, wenn möglich, als Code in Versionierungssystemen abgelegt werden können.

### Informationen

Performance-Engineering muss zwingend an Informationen bezüglich der relevanten Projekte, Produkte und Veränderungen kommen, um möglichst zeitgerecht unterstützen zu können. Gleichzeitig muss Performance-Engineering intern für Aufklärungsarbeit bezüglich dem Thema Performance sorgen. Weiter soll Erreichtes transparent gemacht werden, um intern ein besseres Standing zu erreichen.

# FirmTec-Performance-Engineering

## FirmTec-Ansatz

Der FirmTec-Ansatz ist ein umfassendes Performance-Engineering mit vielen Analogien zur Fertigung von komplexen Systemen, wie zum Beispiel Flugzeugen. Es erstreckt sich über den ganzen Lebenszyklus eines IT-Systems oder einer Applikation ganz im Gegensatz zu klassischen Performance-Tests, welche kurz vor einer Einführung erfolgen und bestenfalls zum Ergebnis kommen, dass mehr Hardware benötigt wird (siehe Abbildung 11).

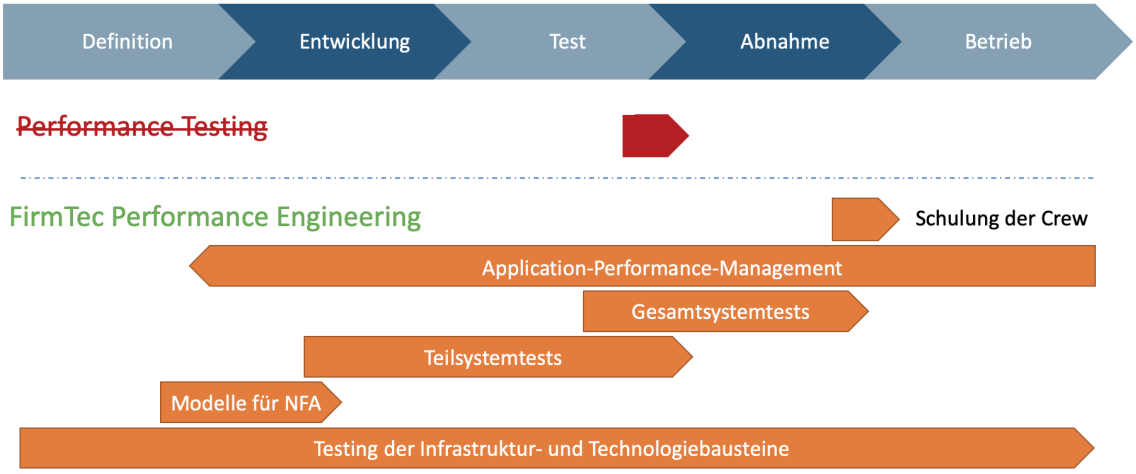


*Abbildung 11: Einbettung von Performance-Engineering im Software-Delivery-Lifecycle. Wichtig ist, dass Performance-Engineering nicht eine kurze Phase im Test oder Abnahmeprozess ist, wie alt hergebrachtes Performance-Testing, sondern sich über den kompletten Lifecycle erstreckt.*

### Testing der Infrastruktur- und Technologiebausteine

Die Grundlage für den erfolgreichen Bau eines Flugzeuges ist das richtige Material: Es braucht Schrauben und Spanten, die der späteren Belastung standhalten können. Analog braucht es im IT-Umfeld eine Infrastruktur und Technologiebausteine, die für die kommende Belastung fit und optimal vorbereitet sind. Diese Optimierungen müssen laufend erfolgen.

### Modelle für NFA

Immer wieder kommt es in der Realität vor, dass keine, unbrauchbare oder aus der Luft gegriffene nicht-funktionale Anforderungen vorliegen. Dies, weil der Zugang zu nichtfunktionalen Anforderungen für viele Fachleute zu technisch oder zu komplex ist. Darum bietet FirmTec Modelle an, um Ursache und Wirkungen von Last aufzuzeigen und sinnvolle Anforderungen zu formulieren. Dies entspricht der Strömungssimulation im Flugzeugbau: Sie visualisiert in einem theoretischen Modell, wo die besonders belasteten Punkte liegen.

### Teilsystemtests

Bevor ein Flugzeug zum ersten Testflug abhebt, werden alle Teile individuell und im Verbund getestet, weil das Risiko und die Kosten für Gesamtsystemtests von Anfang an, viel zu hoch wären. Analog sollten IT-Systeme auch Stück für Stück getestet werden, um Risiken von knappen Verfügbarkeiten auf den Testumgebungen und aufwändigen Datenbereinigungen zu reduzieren.

### Gesamtsystemtests

Gesamtsystemtests sind sehr wichtig, weil sie neue Schwachpunkte zum Vorschein bringen können, welche in Teilsystemtests nicht auftreten. Wichtig ist, dass die Datenkonstellationen und Umgebungskonfigurationen möglichst realistisch sind, also nicht nur der Schönwettersituation entsprechen. Schliesslich soll das neue Flugzeug ja auch bei Gegenwind, Seitenwind oder sogar Sturm fliegen und auch sicher wieder landen können.

### Application-Performance-Management

Application-Performance-Management wird in der Entwicklungs- und in der Testphase bereits aufgebaut, um es in allen Tests effektiv für Performance-Analysen benutzen zu können. Das Feedback aus dem Monitoring ist für Fachvertreter sehr relevant. Darum zeigt der Pfeil im Sinne eines Feedback-Mechanismus bewusst nach links. Dazu werden beispielsweise verständliche Dashboards gebaut, die das Performance-Verhalten aus fachlicher Sicht visualisieren oder entsprechende Reports entwickelt und regelmässig versandt. Das ist sozusagen das Cockpit des neuen Flugzeuges.

### Schulung der Crew

Am effizientesten ist es, wenn die Dashboards aus dem Testbetrieb auch im produktiven Betrieb Anwendung finden. Vor der Betriebsübergabe eines Systems, macht es Sinn, dass die zukünftige Betriebscrew, mit dem neuen IT-System und den Dashboards vertraut gemacht werden. Der Performance-Engineer hat die Möglichkeit, um für die Crew relevante Situationen nachzustellen, damit sie den Ernstfall trainieren können. Beim Flugzeug wäre das beispielsweise ein Triebwerkbrand im Simulator-Training. Im Falle der IT geht es mehr um Failovertests unter Last.

# FirmTec-Angebote

FirmTec ist ein Dienstleistungsanbieter mit starkem Fokus auf Performance-Engineering. Die Dienstleistungsangebote decken folgende Bereiche ab:

## Konzeption neuer Performance-Organisationen

Etablierung von Performance-Engineering-Kompetenzen in der Unternehmung inklusive nachhaltigem Aufbau der Organisation, von Methodik und Technologien.

## Toolevaluationen und -einführungen

Begleitung von unabhängigen Toolevaluationen und -einführung im Bereich Performance-Engineering, insbesondere für Performance-Testing oder Performance Monitoring, inklusive Studien des Bedarfs von massgeschneiderten Einführungskonzepten, aussagekräftigen «Proof-of-Concepts» oder Aufstellen eines überzeugenden Business Cases.

## Mitarbeit in Grossvorhaben

Projektbezogene Unterstützung für Performance-Engineering von der Erhebung nichtfunktionaler Anforderungen, über ganzheitliche Konzeption bis hin zur technischen Realisierung mit den vorhandenen oder neuen Werkzeugen. Bei Bedarf übernimmt FirmTec die Gesamtverantwortung für das Thema Performance.

## Mitarbeit in agilen BizDevOps-Produktteams

Produktfokussierte Unterstützung für Performance-Engineering von der Erhebung nichtfunktionaler Anforderungen, über die passende Konzeption bis hin zur technischen Realisierung mit den vorhandenen oder neuen Werkzeugen, inklusive der Einarbeitung von Performance-Engineering in Delivery Pipelines und dem Aufbau eines Continuous-Performance-Engineering.

## Mitarbeit in Performance-Kompetenz-Center

Tatkräftige Unterstützung für Performance-Engineering von der Erhebung nichtfunktionaler Anforderungen, über flexible Mitarbeit bei Konzepten bis hin zur technischen Realisierung mit den vorhandenen Werkzeugen in diversen Rollen. Zum Beispiel als Performance-Engineer, Performance-Architekt, Performance-(Test)-Manager, Performance-Tester.

## Umstellung bestehender Performance-Organisationen auf agile Arbeitsweisen

Bedarfsgerechte Konzeption und Transformation des Performance-Engineering auf eine agile Arbeitsweise, durch Herleitung des Zielbilds, der Umsetzung in Workshops und durch aktive Mitarbeit.

## Branchenkenntnisse

FirmTec hat Erfahrung im Performance-Engineering in verschiedenen Branchen wie Banking, Blockchain, E-Commerce, Gesundheitswesen, Immobilien, Industrie, IT-Service-Provider und Versicherungen.

# FirmTec-Ansprechpartner

## Dr. Andreas Elsener

Andreas Elsener hat an der ETH Zürich «Computational Science and Engineering» studiert und anschliessend am PSI und dem EPFL in Lausanne in Werkstoffwissenschaften promoviert. Gegenstand seiner Dissertation im High Performance Computing Umfeld waren Simulationen von mechanischen Lasttests mit den grössten Supercomputern der Schweiz zu dieser Zeit.

Nach dem Wechsel in die Privatwirtschaft ist Performance das Thema Nummer eins geblieben. Als Berater und als fachlicher Leiter des Competence Center Performance der Zürcher Kantonalbank vertiefte er sich weiter im Thema und kennt sowohl die Perspektive aus der internen Sicht einer Unternehmung, wie auch die unternehmensübergreifende, externe Sicht.

Seine Stärken liegen in der Analyse und in der Konzeption von neuen Performance-Engineering-Vorhaben und im Umsetzen von Performance-Engineering in Grossprojekten.

Kontakt: +41 79 525 10 42, andreas.elsener@firmtec.ch

## Heinz Grob

Heinz Grob hat Wirtschaftsinformatik studiert und von Anfang an eine starke Affinität für Performance entwickelt. Mit dem Thema Performance ist Heinz Grob in der zweiten Dekade als Selbständiger und als Partner von FirmTec Solutions AG unterwegs.

Heinz hat ein breites Praxiswissen und kennt verschiedenste Technologien und Werkzeugen im Performance-Umfeld.

Seine Stärken liegen in der Technik und im Umsetzen komplexer Performance-Engineering-Vorhaben.

Kontakt: +41 79 242 49 62, heinz.grob@firmtec.ch

# Glossar

| Begriff | Erklärung |
|---|---|
| API | Eigentlich Application-Programming-Interface: Im Kontext von Performance-Engineering geht es hier oft um Webservices. |
| Continuous-Performance-Engineering | Hauptsächlich kurze und fokussierte Performance-Tests, die regelmässig über die Software-Delivery-Pipeline angestossen werden. Ziel ist eine Trendmessung der Performance über die letzten Builds, um bei einer Verschlechterung sofort eingreifen zu können (Fail fast). |
| Geschwindigkeit | Das Laufzeitverhalten einer Applikation: Antwortzeiten, Verarbeitungszeiten oder Durchlaufzeiten |
| Kernsysteme | Zentrale Businessapplikationen einer Unternehmung wie Enterprise-Resource-Planning-Systeme (kurz ERP) oder Customer-Relationship-Management-Systeme (CRM) |
| Nichtfunktionale Anforderungen | Anforderungen an die Qualität, in welcher eine geforderte Funktionalität zu erbringen ist. Dazu gehören unter anderem die Zuverlässigkeit, die Effizienz, die Benutzbarkeit, sowie auch die Skalierbarkeit |
| Performance | Geschwindigkeit, Stabilität und Skalierbarkeit einer Applikation oder eines IT-Systems |
| Performance-Engineering | Die Disziplin, die sich nachhaltig und ganzheitlich um Geschwindigkeit, Stabilität und Skalierbarkeit von IT-Systemen kümmert |
| Skalierbarkeit | Die Möglichkeit unter Hinzunahme von weiteren Ressourcen auch effektiv mehr zu leisten |
| Stabilität | Verfügbarkeit unabhängig von der Belastung des Systems, solange die Last im definierten Rahmen bleibt. |
| Umsystem | System ausserhalb des eigentlichen Entwicklungs- oder Testfokus, zu dem eine direkte Schnittstelle besteht. |
| Verfügbarkeit | Zeiten, in welcher eine Applikation, ein IT-System oder ein Service, verwendet werden kann. Aus Sicht von Endanwendern im Web oder auf Apps geht die Erwartung immer mehr Richtung 7 x 24. |