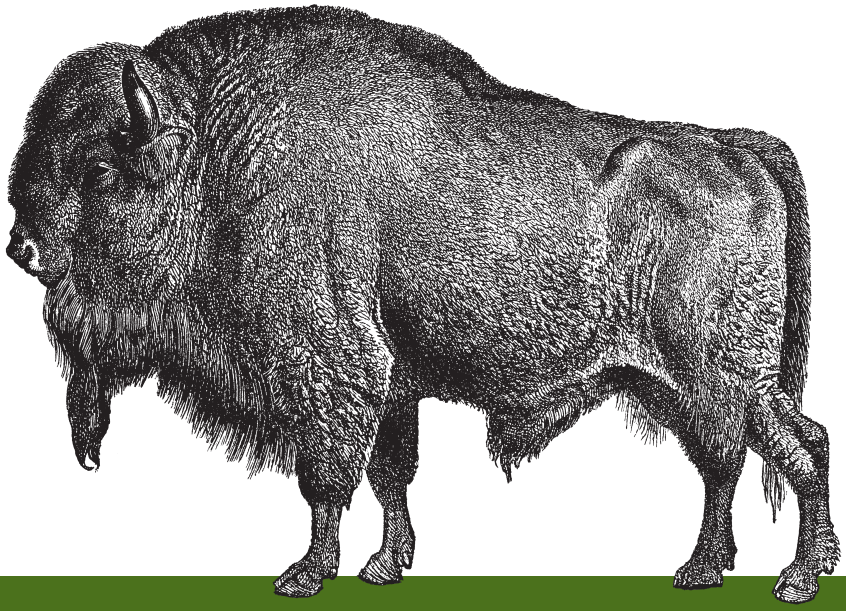
O'REILLY®



Das DevOps Handbuch

TEAMS, TOOLS UND INFRASTRUKTUREN
ERFOLGREICH UMGESTALTEN

Gene Kim, Jez Humble,
Patrick Debois & John Willis
Übersetzung von Thomas Demmig

Das DevOps-Handbuch

Teams, Tools und Infrastrukturen erfolgreich umgestalten

Gene Kim, Jez Humble, Patrick Debois & John Willis

Deutsche Übersetzung von Thomas Demmig

O'REILLY®

Gene Kim, Jez Humble, Patrick Debois & John Willis

Lektorat: Alexandra Follenius
Übersetzung: Thomas Demmig
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Satz: III-satz, www.drei-satz.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Print 978-3-96009-047-2
PDF 978-3-96010-123-9
ePub 978-3-96010-124-6
mobi 978-3-96010-125-3

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.
O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

1. Auflage 2017

5 4 3 2 1 0

Inhalt

Einleitung

Aha!

Das *DevOps-Handbuch* ist nicht von heute auf morgen entstanden – es begann im Februar 2011 mit wöchentlichen Skype-Telefonaten zwischen den Autoren und der Vision, dem damals noch unvollendeten Buch *Projekt Phoenix: Der Roman über IT und DevOps – Neue Erfolgsstrategien für Ihre Firma* ein präskriptives Handbuch zur Seite zu stellen.

Über fünf Jahre und mehr als zweitausend Arbeitsstunden später ist das *DevOps-Handbuch* endlich fertig. Ein außerordentlich langwieriger Prozess, der aber sehr lohnenswert und lehrreich war und den Rahmen viel weiter spannte, als wir uns das ursprünglich vorgestellt hatten. Alle Autoren eint, dass sie an DevOps als außerordentlich wichtiges Konzept glauben, weil sie in ihrem Berufsleben einen »Aha-Moment« hatten, den unsere Leser sicherlich interessiert.

Gene Kim. Ich hatte die Möglichkeit, seit 1999 erfolgreiche Technologiefirmen untersuchen zu können, und eine der ersten Erkenntnisse war, dass die bereichsübergreifende Zusammenarbeit der unterschiedlichen Gruppen – IT Operations, Information Security und Development – für den Erfolg entscheidend ist. Ich erinnere mich vor allem an das erste Mal, als ich die Abwärtsspirale beobachten konnte, die entstand, weil diese Gruppen unterschiedliche Ziele verfolgten.

In 2006 hatte ich die Gelegenheit, für eine Woche eine Gruppe zu begleiten, die die outgesourcten IT Operations eines großen Flugbuchungsdienstleisters betreute. Mir wurden die negativen Folgen der umfangreichen jährlichen Software-Releases beschrieben, die jedes Mal zu großem Chaos und Unterbrechungen für den Outsourcer, aber auch für die Kunden führen würden. Es gäbe wegen der für den Kunden bemerkbaren Ausfälle Strafen aufgrund nicht eingehaltener SLAs (*Service Level Agreement*), der Gewinn bräche ein, und daher würden die talentiertesten und erfahrensten Mitarbeiter entlassen werden. Es müssten mehr ungeplante Aufgaben erledigt und überall Feuer gelöscht werden. Die verbleibende Crew hätte noch weniger Zeit, den stetig wachsenden Service Request Backlog der Kunden abzuarbeiten. Die Verträge würden überhaupt nur durch den massiven Einsatz des mittleren Managements aufrechterhalten werden, und alle hätten Angst, dass sie in drei Jahren bei einer Neuausschreibung sowieso nicht mehr zum Zuge kämen.

Das Gefühl der Hoffnungslosigkeit und Sinnlosigkeit der Arbeit war für mich der Startschuss für meinen moralischen Kreuzzug. Development schien immer als strategisch wichtig angesehen zu werden, während IT Operations nur taktisch betrachtet wurde – etwas, das man gern wegdelegierte oder gleich ganz auslagerte, nur um es fünf Jahre später in einer noch schlechteren Verfassung wiederzubekommen.

Während all der Jahre war den meisten von uns bewusst, dass es einen besseren Weg geben musste. Ich erinnere mich an die Vorträge der Velocity Conference im Jahr 2009, in denen die erstaunlichen Ergebnisse vorgestellt wurden, die aus der Architektur, den Praktiken und kulturellen Normen entstanden, die wir als DevOps kennen. Ich war total begeistert, weil hier der bessere Weg, nach dem wir alle suchten, deutlich aufgezeigt wurde. Dieses Wissen zu verbreiten, war Teil meiner persönlichen Motivation, am *Projekt Phoenix* mitzuarbeiten. Sie können sich vorstellen, wie glücklich ich darüber war, dass viele Menschen durch das Buch ihre eigenen Aha-Momente hatten.

Jez Humble. Mein Aha-Moment mit DevOps erlebte ich bei einem Start-up im Jahr 2000 – in meinem ersten Job, den ich nach dem Studium annahm. Eine Zeit lang war ich einer von zwei Technikmitarbeitern. Ich machte alles: Netzwerk, Programmieren, Support, Systemverwaltung. Wir deployten Software in die Produktivumgebung, indem wir sie per FTP direkt von unseren Workstations hochluden.

2004 erhielt ich dann einen Job bei ThoughtWorks – einer Consulting-Firma, in der ich als Erstes in einem Projekt zusammen mit ungefähr 70 Leuten arbeiten sollte. Ich gehörte zu einem Team aus acht Entwicklern, deren einzige Aufgabe das Deployen unserer Software in eine der Produktivumgebung ähnliche Umgebung war. Zu Beginn war das sehr stressig. Aber im Laufe der Zeit schafften wir es, von einem manuellen Deployment, das zwei Wochen brauchte, zu einem automatisierten zu gelangen, das nur eine Stunde dauerte und bei dem der Wechsel zwischen alter und neuer Umgebung in Millisekunden vonstattenging – während der normalen Arbeitszeit und mithilfe des Blue-Green-Deployment-Musters.

Dieses Projekt war Inspirationsquelle für viele der Ideen in *Continuous Delivery* (Addison-Wesley, 2010) und in diesem Buch. Mich und viele andere, die in diesem Bereich unterwegs sind, treibt das Wissen an, dass es – unabhängig von den gegebenen Randbedingungen – immer besser geht und dass ich den Menschen auf ihrer Reise dorthin helfen möchte.

Patrick Debois. Bei mir waren es mehrere Momente. 2007 arbeitete ich in einem Data-Center-Migration-Projekt mit ein paar Agile-Teams zusammen. Ich war neidisch, dass sie so produktiv waren und in kurzer Zeit so viel erledigen konnten.

In meinem nächsten Projekt begann ich, in Operations mit Kanban zu experimentieren, und ich beobachtete, wie sich die Teamdynamik änderte. Später stellte ich auf der Agile Toronto Conference 2008 mein IEEE-Paper dazu vor, hatte aber das Gefühl, dass es in der Agile-Community nicht groß wahrgenommen wurde. Wir

setzten eine Agile-System-Administration-Gruppe auf, aber ich achtete nicht auf die menschliche Seite des Ganzen.

Nachdem ich auf der Velocity Conference 2009 die Präsentation »10 Deploys per Day« von John Allspaw und Paul Hammond gesehen hatte, war ich davon überzeugt, dass auch andere wie ich dachten. Also entschied ich mich dazu, die ersten DevOpsDays zu organisieren, und prägte damit unabsichtlich den Begriff DevOps.

Die Energie auf diesem Event war einmalig und ansteckend. Als die Menschen anfingen, sich bei mir dafür zu bedanken, dass ihr Leben nun besser geworden sei, verstand ich den Einfluss, den das Ganze hatte. Seitdem habe ich nicht mehr damit aufgehört, DevOps anzupreisen.

John Willis. 2008 hatte ich gerade eine Consulting-Firma verkauft, die auf Beratung im Umfeld von großen Legacy-IT-Operations rund um Configuration Management und Monitoring spezialisiert war (Tivoli). Ich traf Luke Kanies (den Begründer von Puppet Labs), der auf einer Open-Source-Konferenz von O'Reilly einen Vortrag über *Configuration Management* (CM) hielt.

Zuerst lungerte ich nur hinten im Vortragsraum herum, schlug die Zeit tot und dachte: »Was kann mir dieser Zwanzigjährige schon über Configuration Management erzählen?« Schließlich hatte ich schon mein ganzes Leben bei ein paar der größten Unternehmen der Welt gearbeitet und ihnen dabei geholfen, die Architektur für ihr CM und andere Operations-Management-Lösungen aufzubauen. Aber nach fünf Minuten setzte ich mich ganz nach vorne und erkannte, dass alles, was ich in den letzten 20 Jahren gemacht hatte, falsch gewesen war. Luke beschrieb das, was ich heute als CM der zweiten Generation bezeichne.

Nach seiner Session hatte ich die Gelegenheit, mich mit ihm auf einen Kaffee zusammenzusetzen. Von dem, was wir heute als *Infrastructure as Code* bezeichnen, war ich vollkommen überzeugt. Aber bei unserem Gespräch ging Luke noch weiter und erläuterte mir seine Ideen. Er glaubte, dass sich Operations mehr wie Softwareentwicklung verhalten müsse. Sie müssten ihre Konfigurationen unter Versionsverwaltung stellen und für ihren Workflow CI/CD-Delivery-Muster übernehmen. Ich war damals noch der Oldschool-IT-Operations-Mensch und antwortete wohl mit so etwas wie: »Diese Idee wird so wenig Erfolg haben wie Led Zeppelin mit ihrem Folk-Kram.« (Ich lag so was von falsch!)

Etwa ein Jahr später in2009 besuchte ich auf einer anderen O'Reilly-Konferenz (Velocity) einen Vortrag von Andrew Clay Shafer über Agile Infrastructure. Dabei zeigte Andrew dieses berühmte Bild einer Mauer zwischen Entwicklern und Operations, bei der die Arbeit immer nur hinübergeworfen wird. Dieses Bild bezeichnete er als »Wall of Confusion«. Die Ideen in diesem Vortrag kodifizierten das, was Luke mir ein Jahr zuvor zu erklären versuchte, und mir ging ein Licht auf. Im selben Jahr war ich der einzige Amerikaner, der zu den ersten DevOpsDays in Gent eingeladen wurde. Als dieses Event vorbei war, floss DevOps durch meine Adern.

Alle Autoren dieses Buchs hatten ganz klar die gleiche Erscheinung, auch wenn sie aus verschiedenen Richtungen kamen. Aber es gibt eine unglaubliche Menge an Belegen dafür, dass die beschriebenen Probleme so gut wie überall auftauchen und dass sich die Lösungen, die mit DevOps verbunden sind, nahezu universell einsetzen lassen.

Dieses Buch will beschreiben, wie Sie die Transformationen durch DevOps, an denen wir beteiligt waren oder die wir beobachten konnten, selbst umsetzen können, es will aber auch mit vielen Mythen aufräumen, die vorzugeben versuchen, warum DevOps in bestimmten Situationen nicht funktionieren wird. Ein paar der bekanntesten Mythen, die wir über DevOps gehört haben, sind:

Mythos: DevOps ist nur für Start-ups

DevOps-Praktiken entstanden zwar in Internet-»Unicorn«-Firmen wie Google, Amazon, Netflix und Etsy, aber alle waren an einem das Geschäft gefährdenden Punkt angelangt – aufgrund von Problemen, die eher mit den klassischen »Horses«-Firmen verbunden sind: hochriskante Code-Releases, die zu katastrophalen Fehlschlägen zu werden drohten, fehlende Möglichkeiten, Features schnell genug veröffentlichen zu können, um so der Konkurrenz zuvorzukommen, Compliance-Sorgen, die Unfähigkeit, skalieren zu können, riesiges Misstrauen zwischen Entwicklung und Operations usw.

Aber alle diese Firmen schafften es, ihre Architektur, die technischen Praktiken und ihre Kultur so umzustellen, dass diese erstaunlichen Ergebnisse entstanden, die wir mit DevOps verbinden. Wie der Information Security Executive Dr. Branden Williams gern scherzt: »Reden wir nicht mehr über DevOps-Einhörner oder -Pferde, sondern nur noch über Vollblüter oder Pferde, die zum Abdecker müssen.«

Mythos: DevOps ersetzt Agile

DevOps-Prinzipien und -Praktiken sind kompatibel zu Agile, und viele haben das Gefühl, dass es sich bei DevOps um eine logische Fortsetzung der Agile-Reise handelt, die 2001 ihren Anfang nahm. Agile dient häufig als effektiver Türöffner für DevOps, weil es sich auf kleine Teams fokussiert, die den Kunden kontinuierlich qualitativ hochwertigen Code liefern.

Viele DevOps-Praktiken entwickeln sich daraus, dass wir unsere Arbeit über das Ziel des »potenziell auslieferbaren Codes« am Ende jeder Iteration hinaus fortführen und dafür sorgen, dass er sich stets in einem auslieferbaren Zustand befindet, den Entwickler täglich einchecken und mit dem wir unsere Features in produktivähnlichen Umgebungen präsentieren.

Mythos: DevOps ist nicht kompatibel zu ITIL

Viele sehen DevOps als Rückschritt gegenüber ITIL oder ITSM (*IT Service Management*), das ursprünglich 1989 veröffentlicht wurde. ITIL hat viele Generationen

von Ops-Mitarbeitern beeinflusst, einen der Autoren eingeschlossen, und ist eine stetig weiterentwickelte Praxisbibliothek, mit der die Prozesse und Vorgehensweisen erstklassiger IT Operations kodifiziert werden sollen – einschließlich Servicestrategie, Design und Support.

DevOps-Praktiken können kompatibel zum ITIL-Prozess gestaltet werden. Aber um die mit DevOps in Verbindung gebrachten kürzeren Durchlaufzeiten und höheren Deployment-Frequenzen zu unterstützen, müssen viele Teile der ITIL-Prozesse vollständig automatisiert werden, was eine Reihe von Problemen rund um das Konfigurations- und Release-Management löst (zum Beispiel werden so die Konfigurationsmanagement-Datenbank und die Softwarebibliotheken aktuell gehalten). Und weil zu DevOps auch das schnelle Erkennen von und Reagieren auf Serviceprobleme gehören, bleiben die ITIL-Bereiche Servicedesign sowie Störungs- und Problemmanagement so wichtig wie zuvor.

Mythos: DevOps ist nicht kompatibel zu Information Security und Compliance

Das Fehlen klassischer Steuerelemente (zum Beispiel der Segregation of Duty, eines Änderungsgenehmigungsprozesses oder manueller Security Reviews am Ende des Projekts) mag Experten für Information Security und Compliance in Panik versetzen.

Aber das heißt nicht, dass DevOps-Organisationen keine effektiven Steuerelemente besitzen. Statt Security- und Compliance-Aktivitäten nur am Projektende durchzuführen, lassen sich diese auf jeder Ebene der täglichen Arbeit im Softwareentwicklungs-Lifecycle einbinden, was zu mehr Qualität, Sicherheit und Regelkonformität führt.

Mythos: DevOps eliminiert IT Operations – »NoOps«

Viele missverstehen DevOps als vollständiges Eliminieren der Funktionen von IT Operations. Aber das ist nur sehr selten der Fall. Die Arbeit in IT Operations mag sich ändern, aber sie bleibt so wichtig wie zuvor. IT Operations arbeitet jetzt viel früher im Software-Lifecycle mit der Entwicklung zusammen, während Letztere wiederum auch dann noch mit IT Operations zu tun hat, wenn der Code schon längst produktiv gegangen ist.

Anstatt dass IT Operations manuell Aufgaben erledigt, die aus dem Ticketsystem purzeln, ermöglicht diese Gruppe der Entwicklung, produktiver zu werden, indem sie APIs und Self-Service-Plattformen anbietet, mit denen Umgebungen erstellt werden können. Außerdem kann Code getestet und deployt, die Produktivumgebung überwacht und angezeigt werden und vieles mehr. Dadurch nähert sich IT Operations mehr der Entwicklung an (genauso wie QA und Infosec) und ist an der Produktentwicklung beteiligt, wobei das Produkt hier eine Plattform ist, auf der Entwickler sicher, schnell und zuverlässig ihre IT-Services testen, deployen und in der Produktivumgebung laufen lassen können.

Mythos: DevOps ist nur »Infrastructure as Code« oder Automatisierung

Viele der DevOps-Muster in diesem Buch erfordern durchaus eine Automatisierung, aber DevOps benötigt auch kulturelle Normen und eine Architektur, die es der gesamten IT-Werschöpfungskette erlaubt, die gemeinsamen Ziele zu erreichen. Das geht weit über Automatisierung hinaus. Wie Christopher Little – Technology Executive und einer der ersten Chronisten von DevOps – schrieb: »Bei DevOps geht es nicht um Automatisierung, so wie es in der Astronomie nicht um Teleskope geht.«

Mythos: DevOps geht nur mit Open-Source-Software

Auch wenn viele DevOps-Erfolgsgeschichten in Firmen geschrieben werden, die Software wie den LAMP-Stack nutzen (Linux, Apache, MySQL, PHP), können die Ergebnisse unabhängig von der verwendeten Technologie erreicht werden. Es gibt genauso positive Berichte über Anwendungen, die in Microsoft.NET, COBOL oder Mainframe-Assembler-Code geschrieben wurden, ebenso in der SAP-Welt und sogar in Embedded Systems (zum Beispiel HP LaserJet-Firmware).

Den Aha-Moment verbreiten

Alle Autoren dieses Buchs wurden von den erstaunlichen Innovationen inspiriert, die in der DevOps-Community stattfinden, und von den Ergebnissen, die daraus entstehen: Es bilden sich zuverlässige Arbeitssysteme, und kleine Teams können schnell und unabhängig Code entwickeln und validieren, der sich dann zuverlässig an die Kunden ausliefern lässt. Wir glauben, dass es sich bei DevOps um eine Manifestation dynamischer, lernfähiger Firmen handelt, die fortlaufend ihre auf viel Vertrauen basierenden Normen weiterentwickeln. Und genau solche Organisationen werden weiterhin innovativ und im Markt erfolgreich sein.

Wir hoffen wirklich, dass das *DevOps-Handbuch* für viele Menschen eine wertvolle Quelle sein wird:

- ein Ratgeber für das Planen und Umsetzen der DevOps-Transformationen,
- eine Ressource für eine Reihe von Fallstudien, die man untersuchen und von denen man lernen kann,
- eine Geschichte von DevOps,
- ein Werkzeug, mit dem man ein Bündnis zwischen Product Ownern, Architektur, Entwicklung, QA, IT Operations und Information Security schmieden kann, um gemeinsame Ziele zu erreichen,
- ein Weg, um Unterstützung im höheren Management für DevOps-Initiativen zu erhalten,
- ein moralischer Imperativ, um die Art und Weise zu ändern, wie wir Technologiefirmen managen, um effektiver und effizienter zu werden,

- eine Möglichkeit, fröhlichere und menschlichere Arbeitsumgebungen zu schaffen, sowie
- eine Hilfe zum lebenslangen Lernen.

Damit wird es für den Einzelnen nicht nur einfacher, höher gesteckte Ziele zu erreichen, auch die Firma profitiert davon.

Vorwort

In der Vergangenheit gab es in vielen Bereichen des Ingenieurwesens auf die eine oder andere Weise merklichen Fortschritt und ein wachsendes Verständnis der eigenen Arbeit. Es gibt in vielen Ingenieurdisziplinen universitäre Weiterbildung und Organisationen, die Expertenhilfe bieten (Bauwesen, Maschinenbau, Elektrotechnik, Nukleartechnik usw). Tatsächlich braucht die moderne Gesellschaft auch alle Arten von Ingenieuren, um die Vorteile der interdisziplinären Zusammenarbeit zu erkennen und diese umzusetzen.

Denken Sie an die Konstruktion eines modernen Autos. Wo endet die Arbeit eines Maschinenbauers, und wo beginnt die eines Elektrotechnikers? Wo (und wie und wann) sollte jemand mit Aerodynamik-Kenntnissen (der sicherlich eine ziemlich genaue Vorstellung davon hat, wie die Fenster aussehen sollten, wie groß sie zu sein haben und wo sie sitzen sollten) mit einem Experten für die Sitzergonomik zusammenarbeiten? Was ist mit dem chemischen Einfluss des Treibstoffs und des Öls auf die Materialien von Motor und Schläuchen über die gesamte Lebenszeit des Autos? Es gibt zum Design eines Autos noch viel mehr Fragen, die man stellen kann, aber das Ergebnis ist immer das gleiche: Erfolg stellt sich in modernen technischen Projekten nur ein, wenn mehrere Sichtweisen und unterschiedliches Expertenwissen zusammenkommen.

Damit sich ein technischer Bereich weiterentwickelt, muss ein Punkt erreicht werden, an dem man sich Gedanken über den bisherigen Weg macht, diesen von mehreren Seiten beleuchtet und das Ganze so zusammenfasst, dass es für die Vorstellungen der Community zur Zukunft hilfreich ist.

Dieses Buch bietet diese Zusammenfassung und sollte als grundlegende Sammlung von Sichtweisen auf das (zugegebenermaßen sich noch entwickelnde und wachsende) Feld von Softwareentwicklung und Operations gesehen werden.

Egal in welcher Branche Sie arbeiten oder welches Produkt oder welchen Service Ihre Firma bereitstellt – diese Art des Denkens ist für das Überleben jedes Business und Technology Leader außerordentlich wichtig.

John Allspaw, CTO, Etsy
Brooklyn, NY, August 2016

Stellen Sie sich eine Welt vor, in der Dev und Ops zu DevOps werden

Eine Einführung in das DevOps-Handbuch

Stellen Sie sich eine Welt vor, in der Product Owner, Entwicklung, QA, IT Operations und Infosec zusammenarbeiten – nicht nur, um sich gegenseitig zu helfen, sondern auch, um sicherzustellen, dass die gesamte Firma Erfolg hat. Indem für ein gemeinsames Ziel gearbeitet wird, ermöglicht man den schnellen Fluss geplanter Arbeit in die Produktivumgebung (zum Beispiel durch zehn, Hunderte oder Tausende Code-Deploys pro Tag), während gleichzeitig herausragende Stabilität, Zuverlässigkeit, Verfügbarkeit und Sicherheit gewährleistet werden.

In dieser Welt testen funktionsübergreifende Teams rigoros ihre Hypothesen darüber, welche Features die Benutzer am meisten erfreuen und das Unternehmen voranbringen. Sie kümmern sich nicht nur um das Implementieren von Benutzerfeatures, sondern sie stellen auch aktiv sicher, dass ihre Arbeit geräuschlos und wiederholt die gesamte Wertkette durchläuft, ohne in IT Operations oder bei internen oder externen Kunden Chaos und Unterbrechungen zu verursachen.

QA, IT Operations und Infosec arbeiten parallel daran, Reibungsverluste zwischen den Teams zu verringern und Systeme aufzubauen, die es den Entwicklern ermöglichen, produktiver zu sein und bessere Ergebnisse zu erhalten. Indem das Wissen von QA, IT Operations und Infosec mit dem der Delivery-Teams verbunden und automatisierte Self-Service-Tools und -Plattformen geschaffen werden, können die Teams ihre tägliche Arbeit erledigen, ohne von anderen Teams abhängig zu sein.

Damit können Firmen eine zuverlässige Arbeitsumgebung aufbauen, in der kleine Teams schnell und unabhängig voneinander Code entwickeln, testen und deployen und diesen Wert den Kunden schnell, sicher und zuverlässig bereitstellen. So maximieren Firmen die Produktivität der Entwickler, ermöglichen ein unternehmensweites Lernen, sorgen für eine hohe Mitarbeiterzufriedenheit und sind im Markt erfolgreich.

Das sind Ergebnisse, die durch DevOps zustande kommen können. Die meisten von uns leben aber in einer anderen Welt. Häufig ist das System, nach dem wir arbeiten, kaputt, was zu ausgesprochen schlechten Ergebnissen führt, die unser Potenzial bei Weitem nicht widerspiegeln. In unserer Welt sind Entwicklung und

IT Operations Kontrahenten, Tests und Aktivitäten von Infosec geschehen erst am Ende eines Projekts – zu spät, um gefundene Probleme noch zu beheben –, und fast alle kritischen Aufgaben erfordern einen zu großen manuellen Aufwand und zu viele Übergaben, was dazu führt, dass wir ständig warten. So etwas führt nicht nur zu ausgesprochen langen Durchlaufzeiten, wenn wir Aufgaben zu erledigen haben, dazu ist die Qualität unserer Arbeit – insbesondere das Deployen in die Produktion – problematisch und chaotisch, was negative Auswirkungen auf unsere Kunden und unser Geschäft hat.

Letztendlich erreichen wir unsere Ziele nicht, und die gesamte Firma ist mit der Leistung der IT unzufrieden, was zu Budgetkürzungen und frustrierten Mitarbeitern führt, die das Gefühl haben, den Prozess und seine Ergebnisse sowieso nicht ändern zu können.[1] Die Lösung? Wir müssen die Art und Weise, wie wir arbeiten, ändern – und DevOps zeigt uns den besten Weg dorthin.

Um das Potenzial der DevOps-Revolution besser zu verstehen, wollen wir uns die Umwälzungen in der Produktfertigung der 1980er-Jahre anschauen. Durch die Übernahmen von Lean-Prinzipien und -Praktiken verbesserten Firmen drastisch die Produktivität ihrer Fabriken, die Verfügbarkeit für den Kunden, die Produktqualität und die Kundenzufriedenheit, wodurch sie im Markt mehr Erfolg hatten.

Vor der Revolution betrugen die durchschnittlichen Herstellungs- und Lieferzeiten einer Fabrik sechs Wochen, wobei weniger als 70 % der Bestellungen pünktlich beim Kunden ankamen.[2] Im Jahr 2005 – nach einer weiten Verbreitung von Lean-Praktiken – verringerte sich die durchschnittliche Zeitdauer auf weniger als drei Wochen, und mehr als 95 % der Lieferungen waren pünktlich. Firmen, die keine Lean-Praktiken umsetzten, verloren Marktanteile, und viele mussten ganz aufgeben.

Genauso wurde die Latte bei Technologieprodukten und -Dienstleistungen ebenfalls höher gelegt – was früher gut genug war, reicht jetzt nicht mehr aus. In den letzten 40 Jahren sind die Kosten und Zeiträume, die zum Entwickeln und Deployen von geschäftsentscheidenden Fähigkeiten und Features notwendig waren, kontinuierlich um Größenordnungen gesunken. In den 1970er- und 1980er-Jahren brauchten die meisten Features ein bis fünf Jahre für die Entwicklung und Umsetzung, und sie kosteten oft zweistellige Millionenbeträge.

In den 2000ern verringerte sich die Zeit aufgrund der Fortschritte in der Technologie und der Übernahme von agilen Prinzipien und Praktiken auf Wochen oder Monate, während das Deployen in die Produktivumgebung immer noch ebenfalls Wochen oder Monate brauchte – häufig mit katastrophalen Ergebnissen.

1 Das ist nur ein kleiner Ausschnitt der Probleme, die in klassischen IT-Organisationen zu finden sind.

2 Eliyahu M. Goldratt, *Beyond the Goal: Eliyahu Goldratt Speaks on the Theory of Constraints (Your Coach in a Box)* (Prince Frederick, Maryland: Gildan Media, 2005), Audiobook

In den 2010er-Jahren – mit der Einführung von DevOps und immer mehr Verbreitung findender Hardware, Software und mittlerweile der Cloud – können Features (und sogar ganze Start-ups) in Wochen entwickelt werden und innerhalb von Stunden oder Minuten in Produktion gehen. Für solche Firmen wurde das Deployment endlich zur Routine mit nur wenigen Risiken. Sie können Experimente durchführen, um Geschäftsideen auszuprobieren, herausfinden, welche Ideen dem Kunden und der Firma als Ganzem den meisten Nutzen bringen, und diese dann in Features umsetzen, die sich schnell und zuverlässig wieder in der Produktion einsetzen lassen.

Tabelle 1: Der sich stetig beschleunigende Trend hin zu einem schnelleren, billigeren und risikoärmeren Ausliefern von Software (Quelle: Adrian Cockcroft, »Velocity and Volume (or Speed Wins)«, Vortrag auf der FlowCon, San Francisco, November 2013)

	1970er bis 1980er	1990er	2000er bis heute
Ära	Mainframes	Client/Server	Kommodisierung und Cloud
Repräsentative Technologie	COBOL, DB2 auf MVS usw.	C++, Oracle, Solaris usw.	Java, MySQL, Red Hat, Ruby on Rails, PHP usw.
Zyklusdauer	1–5 Jahre	3–12 Monate	2–12 Wochen
Kosten	1.000.000–100.000.000 Dollar	100.000–10.000.000 Dollar	10.000–1.000.000 Dollar
Riskant für	gesamte Firma	Produktlinie oder -bereich	Produktfeature
Folgen bei Fehler	Bankrott, Verkauf der Firma, massive Entlassungen	Umsatzziele nicht erreicht, Jobverlust des CIO	vernachlässigbar

Heutzutage deployen Unternehmen, die die Prinzipien und Praktiken von DevOps übernehmen, oft Hunderte oder Tausende Änderungen pro Tag. In einer Zeit, in der die Time to Market kurz sein und man fortlaufend experimentieren muss, sind Firmen, die diese Ergebnisse nicht erreichen können, dazu verdammt, Marktanteile zu verlieren und eventuell sogar ganz unterzugehen – so wie die Fertigungsfirmen, die keine Lean-Prinzipien übernahmen.

Egal in was für einer Branche wir uns messen: Heutzutage hängt es von der Technologie-Wertkette ab, wie wir Kunden gewinnen und ihnen etwas bieten können. Oder wie es Jeffrey Immelt, CEO von General Electric, kurz und prägnant zusammenfasst: »Jede Branche und jede Firma, die es nicht schafft, ihr Kerngeschäft mit Software zu unterstützen, wird untergehen.« [3] Jeffrey Snover, Technical Fellow bei Microsoft, sagt: »Früher haben Firmen Wert erzeugt, indem sie Atome verschoben. Heute erzeugen sie Wert, indem sie Bits verschieben.«[4]

3 Jeff Immelt, »GE CEO Jeff Immelt: Let's Finally End the Debate over Whether We Are in a Tech Bubble«, *Business Insider*, 9. Dezember 2015, *http://www.businessinsider.com/ceo-of-ge-lets-finally-end-the-debate-over-whether-we-are-in-a-tech-bubble-2015-12*

4 »Weekly Top 10: Your DevOps Flavor,« *Electric Cloud*, 1. April 2016, *http://electric-cloud.com/blog/2016/04/weekly-top-10-devops-flavor/*

Man kann die Größe dieses Problems gar nicht hoch genug ansetzen – jede Firma ist betroffen, unabhängig von Branche, Größe oder ob sie kommerziell oder gemeinnützig ist. Mehr denn je wird durch die Art und Weise, wie Technologiearbeit verwaltet und umgesetzt wird, bestimmt, ob unsere Firmen im Markt wachsen können und ob sie überhaupt überleben werden. In vielen Fällen müssen wir uns Prinzipien und Praktiken zu eigen machen, die sich deutlich von denen unterscheiden, mit denen wir in den letzten Jahrzehnten erfolgreich waren (siehe Anhang A).

Nachdem wir die Dringlichkeit des durch DevOps lösbaren Problems dargestellt haben, wollen wir uns nun genauer dessen Symptome anschauen, herausfinden, warum es auftritt und warum es sich – ohne massive Eingriffe – mit der Zeit verschlimmert.

Das Problem: Etwas in Ihrer Firma muss verbessert werden (denn sonst würden Sie dieses Buch nicht lesen)

Die meisten Firmen schaffen es nicht, Änderungen an der Produktivumgebung in Minuten oder Stunden zu deployen, der Zeitrahmen beträgt eher Wochen oder Monate. Und sie können auch keine Hunderte oder Tausende Änderungen pro Tag übernehmen, stattdessen hadern sie schon mit monatlichen oder gar vierteljährlichen Deployments. Zudem sind Deployments für die Produktivumgebung nie Routine, es kommt immer wieder zu Ausfällen, chronischem Feuerlöschen und unvermeidlichen Heldentaten.

In einer Zeit, in der man für einen Vorteil gegenüber dem Wettbewerber schnell am Markt sein, gute Serviceleistungen erbringen und immer wieder experimentieren muss, haben diese Firmen einen deutlichen Wettbewerbsnachteil. Das liegt zu großen Teilen an ihrer Unfähigkeit, einen zentralen chronischen Konflikt innerhalb ihrer Technologieorganisation aufzulösen.

Der zentrale, chronische Konflikt

In so gut wie jeder IT-Organisation gibt es einen inhärenten Konflikt zwischen Entwicklung und IT Operations, der zu einer Abwärtsspirale bei der Time to Market für neue Produkte und Features und der Qualität führt. Ausfälle treten häufiger auf, und – am schlimmsten – die technischen Schulden steigen stetig.

Der Begriff »technische Schulden« wurde erstmalig von Ward Cunningham geprägt. Analog zu den finanziellen Schulden beschreiben die technischen Schulden, wie von uns getroffene Entscheidungen zu Problemen führen, die sich mit der Zeit immer schwieriger beheben lassen und die uns zur Verfügung stehenden Optionen nach und nach verringern – auch wenn wir vernünftig damit umgehen, zahlen wir doch die Zinsen.

Ein Faktor, der dazu beiträgt, sind die häufig konkurrierenden Ziele von Entwicklung und IT Operations. IT-Organisationen sind für viele Dinge verantwortlich, unter anderem auch für die folgenden zwei Ziele, die gleichzeitig verfolgt werden müssen:

- Auf die sich sehr schnell verändernde Wettbewerbssituation reagieren.
- Stabile, zuverlässige und sichere Services für den Kunden bieten.

Meist ist die Entwicklung dafür verantwortlich, auf Änderungen im Markt zu reagieren und Features so schnell wie möglich in die Produktivumgebung zu deployen. IT Operations wiederum ist dafür verantwortlich, den Kunden IT-Services zu bieten, die stabil, zuverlässig und sicher sind, womit es für jemanden, der potenziell kritische Änderungen an der Produktivumgebung vornehmen will, schwierig bis unmöglich wird, dies umzusetzen. In dieser Konstellation haben Entwicklung und IT Operations diametral entgegengesetzte Ziele und Motivationen.

Dr. Eliyahu M. Goldratt, einer der Begründer der Manufacturing-Management-Bewegung, hat diese Art von Konstellation »den zentralen chronischen Konflikt« genannt[5] – wenn die Ziele und Motivationen der unterschiedlichen Silos verhindern, dass globale, firmenweite Ziele erreicht werden.[6]

Dieser Konflikt sorgt für eine Abwärtsspirale, die so mächtig ist, dass sie das Erreichen der gewünschten Geschäftsergebnisse verhindert – sowohl innerhalb als auch außerhalb der IT-Organisation. Diese chronischen Konflikte bringen die Mitarbeiter häufig in Situationen, aus denen schlechte Software und ungenügende Servicequalität entstehen, daraus folgend schlechte Ergebnisse beim Kunden und ein täglicher Kampf mit Workarounds, Brandbekämpfung und Heldentaten – sei es im Produktmanagement, in der Entwicklung, in QA, IT Operations oder Information Security (siehe Anhang B).

Abwärtsspirale in drei Akten

Die Abwärtsspirale in der IT besteht aus drei Akten, die den meisten IT-Erfahrenen bekannt vorkommen dürften.

Der erste Akt beginnt in IT Operations, in der wir das Ziel haben, Anwendungen und Infrastruktur am Laufen zu halten, sodass unsere Firma den Kunden Werte liefern kann. In der täglichen Arbeit entstehen viele unserer Probleme durch Anwendungen und Infrastruktur, die komplex, schlecht dokumentiert und unglaublich fragil ist. Das sind die technischen Schulden und die täglichen Workarounds, mit denen wir ständig leben müssen, wobei wir immer versprechen, dass wir da mal ordentlich aufräumen, wenn wir ein bisschen Zeit haben. Aber diese Zeit haben wir nie.

5 Goldratt, *Beyond the Goal*

6 Im Fertigungsumfeld existiert ein ähnlicher zentraler chronischer Konflikt: die Notwendigkeit, eine pünktliche Lieferung an die Kunden zu erreichen und gleichzeitig die Kosten im Griff zu behalten. Wie dieser zentrale chronische Konflikt gelöst wurde, ist in Anhang B beschrieben.

Besonders alarmierend ist, dass die fragilsten Bestandteile entweder unser wichtigsten und am meisten Umsatz erzeugenden Systeme oder unsere kritischsten Projekte unterstützen. Mit anderen Worten: Die Systeme, die am ehesten Probleme bereiten, sind auch unsere wichtigsten Systeme, und sie sind das Ziel unserer dringlichsten Änderungen. Schlagen diese Änderungen fehl, gefährdet das unsere wichtigsten Firmenversprechen, wie Verfügbarkeit für die Kunden, Umsatzziele, Sicherheit der Kundendaten, korrekte Finanzdaten usw.

Der zweite Akt beginnt, wenn jemand einen Ausgleich für das letzte gebrochene Versprechen bieten muss – sei es ein Produktmanager, der ein größeres, tolleres Feature verspricht, um die Kunden zu blenden, oder der Geschäftsführer, der ein noch höheres Umsatzziel setzt. Dann wird der Technologiebereich der Firma dazu verpflichtet, dieses neue Versprechen zu erfüllen – ohne zu wissen, was die Technologie leisten kann (oder auch nicht) oder welche Faktoren überhaupt dazu geführt haben, dass das erste Versprechen gebrochen wurde.

Die Entwicklungsabteilung wird also damit beauftragt, noch ein dringendes Projekt umzusetzen, für das natürlich wieder Herausforderungen gemeistert werden müssen und neueste Technologien erforderlich sind, um die versprochenen Release Dates zu halten. Das führt zu noch mehr technischen Schulden, die wir natürlich nur machen, weil wir sie ganz bestimmt abbauen werden, wenn wir ein bisschen mehr Zeit haben.

Damit ist die Bühne frei für den dritten und letzten Akt, in dem alles Schritt für Schritt noch ein bisschen schwieriger wird – jeder ist ein bisschen beschäftigter, die Arbeit dauert ein bisschen länger, die Kommunikation wird ein bisschen langsamer, und die Liste der Aufgaben wird ein bisschen länger. Unsere Arbeit ist mehr voneinander abhängig, kleinere Aktionen führen zu größeren Problemen, und wir bekommen mehr Angst und sind Änderungen gegenüber noch weniger aufgeschlossen. Für die Arbeit sind mehr Kommunikation, Koordination und Genehmigungen nötig, die Teams müssen ein bisschen länger darauf warten, dass andere ihre Arbeit erledigen, und die Qualität wird immer schlechter. Die Räder drehen sich langsamer, und es ist aufwendiger, sie überhaupt am Laufen zu halten (siehe Anhang C).

Es mag zwar als Betroffener schwierig sein, die Abwärtsspirale zu sehen, aber wenn man einen Schritt zurücktritt, ist sie offensichtlich. Wir sehen, dass die Produktiv-Deployments von Code immer länger brauchen, bis sie abgeschlossen sind – von Minuten über Stunden bis zu Tagen und Wochen. Und die Ergebnisse der Deployments werden stetig problematischer, was zu wachsenden für die Kunden bemerkbaren Ausfällen führt, für die Operations Heldentaten vollbringen und Brände löschen muss. Das wiederum verringert noch mehr die Chance, technische Schulden zurückzahlen zu können.

Im Ergebnis werden unsere Produkt-Delivery-Zyklen langsamer und langsamer, weniger Projekte können umgesetzt werden, und die, die man angeht, sind nicht sehr ambitioniert. Zudem wird das Feedback zur geleisteten Arbeit langsamer und seltener, insbesondere von den Kunden. Und egal wie sehr wir uns bemühen – alles scheint schlimmer zu werden. Wir können nicht mehr schnell auf die sich ändernde Konkurrenzsituation reagieren und sind nicht dazu in der Lage, unseren Kunden stabile, zuverlässige Services zu bieten. Schließlich haben wir im Markt verloren.

Immer wieder sehen wir, dass die gesamte Firma ins Stolpern gerät, wenn die IT Probleme hat. Steven J. Spear schrieb in seinem Buch *The High-Velocity Edge*: Egal ob die Schäden »sich langsam wie eine Seuche ausbreiten« oder schnell »wie ein heftiger Unfall entstehen ... die Zerstörung kann umfassend sein«.[7]

Warum tritt diese Abwärtsspirale überall auf?

Über zehn Jahre lang haben die Autoren dieses Buchs diese destruktive Spirale in unzähligen Firmen unterschiedlichsten Typs und verschiedenster Größe beobachten können. Besser als je zuvor verstehen wir, warum diese Spirale auftritt und warum DevOps-Prinzipien notwendig sind, um sie zu entschärfen. Zum einen hat jede IT-Organisation, wie schon beschrieben, zwei entgegengesetzte Ziele, zum anderen ist jede Firma eine Technologiefirma – ob sie es weiß oder nicht.

Wie Christopher Little, Software Executive und einer der ersten Chronisten von DevOps, schrieb: »Jede Firma ist eine Technologiefirma, egal was für ein Geschäft sie eigentlich betreibt. Eine Bank ist auch nur eine IT-Firma mit einer Banklizenz.«[8] [9]

Um sich davon zu überzeugen, dass dies tatsächlich der Fall ist, machen Sie sich klar, dass der Großteil der Investitionsvorhaben auch IT mit einbezieht. Wie es so schön heißt: »Es ist nahezu unmöglich, Geschäftsentscheidungen zu treffen, die nicht mindestens eine IT-Änderung nach sich ziehen.«

Im geschäftlichen und finanziellen Kontext sind Projekte kritisch, weil sie der primäre Auslöser für Änderungen in Firmen sind. Projekte sind das, was das Management normalerweise genehmigen muss, wofür es Budget einplant und für die es verantwortlich ist. Daher sind sie die Mechanismen, über die die Ziele und das Streben der Firma erreicht werden – um zu wachsen oder auch um zu schrumpfen.[10]

7 Steven J. Spear, *The High-Velocity Edge: How Market Leaders Leverage Operational Excellence to Beat the Competition* (New York, NY: McGraw Hill Education), Kindle Edition, Kap. 3

8 2013 hat die europäische Bank HSBC mehr Softwareentwickler beschäftigt als Google.

9 Chris Skinner, »Banks have bigger development shops than Microsoft,« Chris Skinner's Blog, accessed July 28, 2016, *http:// thefinanser.com/2011/09/banks-have-bigger-development-shops-than-microsoft.html/*

10 Wir wollen hier nicht diskutieren, ob Software als »Projekt« oder als »Produkt« finanziert werden soll. Darum wird es später im Buch noch gehen.

Projekte werden im Allgemeinen über Investitionen finanziert (also Fabriken, Material und große Projekte, und Ausgaben werden kapitalisiert, wenn die Amortisation Jahre brauchen wird), von denen mittlerweile 50 % Technologiebezug haben.[11] Das gilt selbst in »Lowtech-Industriezweigen«, die früher am wenigsten für Technologie ausgegeben haben, zum Beispiel im Energiebereich, der Metallverarbeitung, der Rohstoffgewinnung, im Bereich Automotive und der Baubranche. Mit anderen Worten: Führungskräfte hängen zum Erreichen ihrer Ziele viel mehr von einem effektiven IT-Management ab, als sie dachten.[12]

Die Kosten für Mensch und Wirtschaft

Wenn Mitarbeiter, insbesondere solche, die »nach« der Entwicklung kommen, jahrelang in dieser Abwärtsspirale gefangen sind, haben sie häufig das Gefühl, in einem System eingesperrt zu sein, das nur fehlschlagen kann, und sie sehen keine Möglichkeit, die Ergebnisse zu ändern. Auf diese Machtlosigkeit folgt schnell ein Burn-out, das mit den entsprechenden Gefühlen von Erschöpfung, Zynismus und sogar Hoffnungslosigkeit und Verzweiflung einhergeht.

Viele Psychologen sagen, dass das Erzeugen von Systemen, die Gefühle von Machtlosigkeit verursachen, eines der gefährlichsten Dinge ist, die wir Mitmenschen antun können – wir rauben anderen Menschen ihre Fähigkeit, ihre eigenen Ergebnisse zu steuern, und wir sorgen sogar für eine Kultur, in der die Menschen Angst haben, das Richtige zu tun, weil sie sich vor Bestrafung und Fehlschlägen fürchten oder sogar Sorge um ihre Existenzgrundlage haben. Das schafft die Grundlage für eine *erlernte Hilflosigkeit*, bei der den Mitarbeitern der Wille oder überhaupt die Fähigkeit verloren geht, sich so zu verhalten, dass ein gleiches Problem in Zukunft nicht mehr auftritt.

Für unsere Mitarbeiter bedeutet das viele Überstunden, Arbeit am Wochenende und eine verringerte Lebensqualität – nicht nur für die Mitarbeiter selbst, sondern für alle, die von ihnen abhängen, einschließlich Familie und Freunde. Es ist nicht überraschend, dass wir in solchen Situationen unsere besten Leute verlieren (abgesehen von denen, die das Gefühl haben, nicht gehen zu können, weil sie sich verantwortlich fühlen oder weil sie andere Verpflichtungen haben).

11 Nico Stehr und Reiner Grundmann, *Knowledge: Critical Concepts, Volume 3* (London: Routledge, 2005), 139

12 Dr. Vernon Richardson und seine Kollegen haben diese erstaunlichen Ergebnisse veröffentlicht. Sie untersuchten die Jahresabschlüsse von 184 börsennotierten Firmen und teilten diese in drei Gruppen ein: A) Firmen mit wesentlichen Schwächen mit IT-bezogenen Defiziten, B) Firmen mit wesentlichen Schwächen ohne IT-bezogene Definite und C) »saubere Firmen« ohne wesentliche Schwächen. Bei Firmen in Gruppe A wurde der CEO achtmal häufiger gewechselt als in Gruppe C, und der CFO wechselte viermal häufiger als in Gruppe C. IT ist also ganz klar entscheidender, als wir im Allgemeinen denken. – A. Masli, V. Richardson, M. Widenmier und R. Zmud, »Senior Executive's IT Management Responsibilities: Serious IT Deficiencies and CEO-CFO Turnover«, *MIS Quaterly* (elektronisch veröffentlicht, 21. Juni 2016)