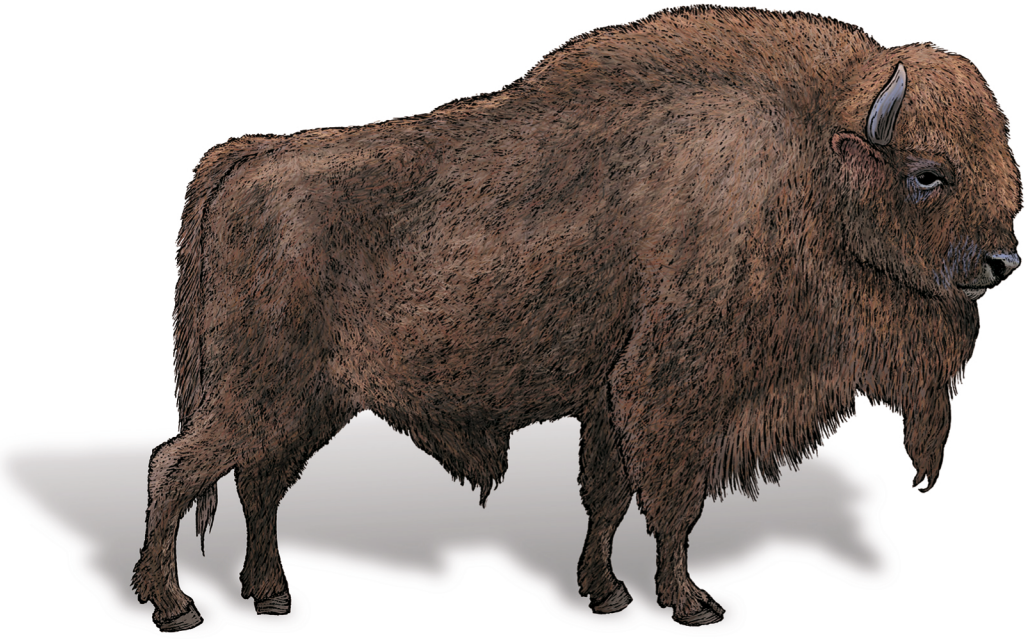
O'REILLY®

2. Auflage
Aktualisiert
und erweitert

Das DevOps Handbuch

Teams, Tools und Infrastrukturen erfolgreich umgestalten



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

2. AUFLAGE

Das DevOps-Handbuch

Teams, Tools und Infrastrukturen erfolgreich umgestalten

Gene Kim, Jez Humble, Patrick Debois, John Willis, Nicole Forsgren

Deutsche Übersetzung von Thomas Demmig

O'REILLY®

Gene Kim, Jez Humble, Patrick Debois, John Willis, Nicole Forsgren

Lektorat: Alexandra Follenius
Übersetzung: Thomas Demmig
Korrektorat: Sibylle Feldmann, *www.richtiger-text.de*
Satz: III-Satz, *www.drei-satz.de*
Herstellung: Stefanie Weidner, Frank Heidt
Umschlaggestaltung: Karen Montgomery, Michael Oréal, *www.oreal.de*
Druck und Bindung: mediaprint solutions 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 978-3-96009-199-8
PDF 978-3-96010-696-8
ePub 978-3-96010-697-5
mobi 978-3-96010-698-2

2. Auflage 2022

Wieblinger Weg 17
69123 Heidelberg

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.

Hinwbeis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *kommentar@oreilly.de*.

5 4 3 2 1 0

Inhalt

Anmerkung des Verlegers zur zweiten Auflage

Einfluss der ersten Auflage

Seit der ersten Veröffentlichung des *DevOps-Handbuchs* haben Daten aus den *State of DevOps Reports* und andere Forschungsergebnisse immer wieder gezeigt, dass DevOps den Time to Value für Firmen verbessert sowie die Produktivität und das Wohlbefinden der Mitarbeiter steigert. DevOps hilft auch dabei, flexible und agile Unternehmen zu schaffen, die sich an außergewöhnliche Veränderungen anpassen können, wie das in der seit 2020 andauernden COVID-19-Pandemie zu sehen war.

»Ich denke, 2020 hat gezeigt, was Technologie in Zeiten außerordentlicher Krisen leisten kann«, schreibt Gene Kim in seinem Artikel »State of DevOps: 2020 and Beyond«. »Die Krise diente als Katalysator für rapide Veränderungen. Und ich bin dankbar, dass wir sie bewältigen konnten.«[1]

Eine der Ideen von DevOps und dem *DevOps-Handbuch* ist, dass es die Arbeitspferde und nicht die Einhörner der Firmen und der Technologie-Welt vorstellt – und auch dafür geschrieben ist. DevOps war nie – und ist es auch weiterhin nicht – nur für Technologie-Riesen wie den FAANGs (Facebook, Amazon, Apple, Netflix und Google) oder Start-ups effektiv. Dieses Buch und die DevOps-Community im Ganzen haben immer wieder gezeigt, dass DevOps-Praktiken und -Prozesse selbst von Altsystemen durchzogene Organisationen in alten »Arbeitspferd«-Unternehmen in flexible Technologie-Organisationen verwandeln können.

Heute wird deutlicher als je zuvor, dass jede Firma eine Technologie-Firma und jeder Leader ein Technologie-Leader ist. Technologie kann nicht nur nicht länger ignoriert oder in den Keller verbannt werden – sie ist auch als vitaler Teil der gesamten Strategiemaßnahmen der Firma zu berücksichtigen.

1 Kim, »State of DevOps: 2020 and Beyond«

Änderungen zur zweiten Auflage

In dieser erweiterten Auflage des *DevOps-Handbuchs* haben die Autorin und die Autoren den Haupttext dort aktualisiert, wo neue Forschungen, Ergebnisse und Erfahrungen unser Verständnis von DevOps und seinen Einsatz in der Branche weiterentwickelt und neu geformt haben. Zudem freuen wir uns darüber, die renommierte Forscherin Dr. Nicole Forsgren als Koautorin gewonnen zu haben, die dabei half, den Text durch neue Untersuchungen und weitere Metriken zu aktualisieren.

Continuous Learning

Wir haben ein paar zusätzliche Einsichten und Ressourcen mit aufgenommen, die uns erst nach dem Erscheinen der ersten Auflage aufgefallen sind oder die erst danach veröffentlicht wurden. Diese »Continuous Learning«-Abschnitte sind im Buch wie hier hervorgehoben und enthalten neue Daten und zusätzliche Ressourcen, Werkzeuge und Techniken, die Sie auf Ihrer DevOps-Reise einsetzen können.

Außerdem haben wir das Buch durch zusätzliche Fallstudien ergänzt, um zu zeigen, wie weit sich DevOps in allen Branchen verbreitet hat – insbesondere wie es über die IT-Abteilungen hinaus bis in die C-Ebene vorgedrungen ist. Zusätzlich haben wir am Ende jeder Fallstudie ein oder zwei wichtige Schlussfolgerungen angefügt, die die wichtigsten (aber nicht die einzigen) Erkenntnisse daraus zusammenfassen. Und schließlich haben wir die Zusammenfassung am Ende jedes Buchteils durch neue Ressourcen ergänzt, mit denen Sie Ihre Reise fortführen können.

Wie geht es mit DevOps und dem Softwarezeitalter weiter?

Wenn uns die letzten fünf Jahre etwas gelehrt haben, dann, dass Technologie immer wichtiger wird und dass wir viel erreichen können, wenn IT und Business offen und ehrlich miteinander reden – so wie DevOps es ermöglicht.

Vielleicht wird das am deutlichsten durch die rapiden Veränderungen, die aufgrund der COVID-19-Pandemie seit dem Jahr 2020 erforderlich sind. Mithilfe von DevOps konnten Organisationen Technologie nutzen, um internen und externen Kunden während Zeiten großer Verwerfungen Services anzubieten. Diese großen, komplexen Organisationen, die für ihre Unfähigkeit bekannt waren, sich zu verändern und sich schnell anzupassen, hatten plötzlich keine andere Wahl mehr.

American Airlines konnte sich die laufende DevOps-Transformation ebenfalls zunutze machen, um schnell große Gewinne einzufahren, wie Sie in den Kapiteln 1 und 5 lesen werden.

Dr. Chris Strear berichtet in Kapitel 2 von seinen Erfahrungen mit der Theory of Constraints, um die Abläufe in Krankenhäusern zu optimieren.

Im Jahr 2020 konnte Nationwide Building Society – die weltgrößte Bausparkasse – dank ihrer aktuellen DevOps-Transformation innerhalb von Wochen auf Kundenanforderungen reagieren, statt wie sonst üblich in Jahren zu agieren. Mehr über ihre Erfahrungen lesen Sie in Kapitel 8.

Bei der erfolgreichen Transformation hin zu zukunftsträchtigen Arbeitsweisen spielt die Technologie eine wichtige Rolle, aber Business Leadership muss die Führung übernehmen. Den Flaschenhals bilden heutzutage nicht mehr technische Vorgehensweisen (auch wenn es diese Engpässe weiterhin gibt), die größte Herausforderung – damit auch die schwierigste Anforderung – besteht darin, die Firmenleitung mit ins Boot zu holen. Transformation muss gemeinsam durch Business und Technologie umgesetzt werden, und die hier vorgestellten Theorien können diese Veränderung unterstützen.

Das Unternehmen kann nicht mehr länger in binären Denkmustern agieren – Topdown oder rein technologiegetrieben. Wir müssen eine echte Zusammenarbeit erreichen. Neunzig Prozent dieser Arbeit dreht sich darum, die richtigen Menschen mit einzubeziehen und engagiert auf einen sinnvollen Weg zu schicken. Beginnen Sie damit, kann uns die sich daraus ergebende Motivation in die Zukunft tragen.

IT Revolution
Portland, OR Juni 2021

Vorwort zur zweiten Auflage

Seit der Veröffentlichung der ersten Auflage des DevOps-Handbuchs sind fünf Jahre vergangen. Es hat sich viel verändert, aber es ist auch vieles gleich geblieben. Manche unserer Werkzeuge und Technologien sind aus der Mode gekommen, andere überhaupt nicht mehr vorhanden, aber das sollte Sie als Leserin oder Leser nicht abschrecken. Auch wenn sich die technologische Landschaft verändert hat, bleiben die diesem Buch zugrunde liegenden Prinzipien genauso relevant wie immer.

Tatsächlich ist der Bedarf an DevOps heutzutage sogar noch gewachsen, wenn Organisationen schneller, sicherer und zuverlässiger Werte an ihre Kunden und Anwenderinnen liefern müssen. Dazu müssen sie ihre internen Prozesse transformieren und mithilfe von Technologie Wert bieten – unter Rückgriff auf DevOps-Praktiken. Das gilt für Organisationen auf der ganzen Welt und in allen Branchen.

In den letzten Jahren habe ich Untersuchungen für die jährlichen *State of DevOps Reports* geleitet (zuerst mit Puppet, später mit DORA und Google) – zusammen mit den Koautoren Jez Humble und Gene Kim. Die Forschung hat gezeigt, dass viele der in diesem Buch beschriebenen Praktiken die Ergebnisse wie Geschwindigkeit und Stabilität von Software-Releases verbessern – ebenso wie sie zur Leistung von Organisationen beitragen, unter anderem zu Profitabilität, Produktivität, Kundenzufriedenheit, Effektivität und Effizienz.

Für die zweite Auflage des DevOps-Handbuchs haben wir den Text mit aktualisierten Daten überarbeitet, die auf neuesten Forschungen und Best Practices basieren, und neue Fallstudien mit aufgenommen, um noch mehr Geschichten davon zu erzählen, wie Transformationen »an der Basis« aussehen. Vielen Dank, dass Sie uns auf dieser Reise durch die fortwährende Verbesserung begleiten.

Nicole Forsgren, PhD
Partner bei Microsoft Research 2021

Vorwort zur ersten Auflage

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

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 Technologie-Firmen 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* und in diesem Buch. Mich und viele andere, die in diesem Bereich unterwegs sind, treibt das Wissen an, dass wir es trotz aller Einschränkungen immer noch besser machen können 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 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.

DevOps-Mythen

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.«[1]

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.

1 Branden Williams, private Korrespondenz mit den Autoren, 2015.

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. Um die mit DevOps in Verbindung gebrachten kürzeren Durchlaufzeiten und höheren Deployment-Frequenzen zu unterstützen, müssen viele Teile von ITIL 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.«[2]

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 Owner, Architektur, Entwicklung, QA, IT Operations und Information Security schmieden kann, um gemeinsame Ziele zu erreichen,

2 Christopher Little, private Korrespondenz mit Gene Kim, 2010.

- ein Weg, um Unterstützung im höheren Management für DevOps-Initiativen zu erhalten,
- ein moralischer Imperativ, um die Art und Weise zu ändern, wie wir Technologie-Firmen 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.

Eine Einführung in das DevOps-Handbuch

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

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. 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.[2] Firmen, die keine Lean-Praktiken umsetzten, verloren Marktanteile, und viele mussten ganz aufgeben.

Genauso wurde die Latte bei Technologie-Produkten 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.

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 durchfüh-

1 Das ist nur ein kleiner Ausschnitt aus den Problemen, die in klassischen IT-Organisationen zu finden sind.

2 Goldratt, *Beyond the Goal*

ren, 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]

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 Technologie-Arbeit 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).

3 Jeff Immelt, »Let's Finally End the Debate«

4 »Weekly Top 10: Your DevOps Flavor,« *Electric Cloud*

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 Technologie-Organisation 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.

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

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.

Feature verspricht, um die Kunden zu blenden, oder der Geschäftsführer, der ein noch höheres Umsatzziel setzt. Dann wird der Technologie-Bereich 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]

7 Steven J. Spear, *The High-Velocity Edge*, Kap. 3

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 Technologie-Firma – ob sie es weiß oder nicht.

Wie Christopher Little, Software Executive und einer der ersten Chronisten von DevOps, schrieb: »Jede Firma ist eine Technologie-Firma, egal was für ein Geschäft sie eigentlich betreibt. Eine Bank ist auch nur eine IT-Firma mit einer Banklizenz.«[8, 9, 10]

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.[11]

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 % Technologie-Bezug haben.[12] 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.[13]

8 Christopher Little, private Korrespondenz mit Gene Kim, 2010.

9 2013 hat die europäische Bank HSBC mehr Softwareentwickler beschäftigt als Google.

10 Skinner, »Banks have bigger development shops than Microsoft«

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

12 Stehr und Grundmann, *Knowledge*, 139

13 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. – Masli et al., »Senior Executives' IT Management Responsibilities«

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. Diese Art von Kultur schafft die Grundlage für eine erlernte Hilflosigkeit, bei der den Mitarbeitenden 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).

Zusätzlich zu den menschlichen Leiden bei dieser Art von Arbeit sind auch die Opportunitätskosten für den Wert, den wir erschaffen können, außerordentlich hoch – die Autoren glauben, dass wir etwa 2,6 Billionen Dollar Wertschöpfung pro Jahr verlieren, was aktuell der jährlichen Wirtschaftsleistung von Frankreich entspricht, der sechstgrößten Wirtschaftsmacht der Welt.

Denken Sie mal über folgende Berechnungen nach: Sowohl IDC als auch Gartner haben geschätzt, dass im Jahr 2011 ungefähr 5 % des weltweiten Bruttoinlandsprodukts (3,1 Billionen Dollar) für IT ausgegeben wurden (Hardware, Service und Telekommunikation). Wenn wir annehmen, dass 50 % von diesen 3,1 Billionen Dollar für Betriebskosten und das Betreuen bestehender Systeme genutzt werden und dass ein Drittel dieser 50 % für dringende und ungeplante Arbeit und Überarbeitung draufgingen, wurden etwa 520 Milliarden Dollar verschwendet.[14]

Sofern uns der Einsatz von DevOps ermöglicht, durch besseres Management und verbesserte operative Leistungen diese Verschwendung zu halbieren und das Mitarbeiterpotenzial in etwas umzuwandeln, das den Wert verfünffacht (eine zurückhaltende Schätzung), könnten wir 2,6 Billionen Dollar Wert pro Jahr schaffen.

14 »IDC Forecasts Worldwide IT Spending to Grow 6%«, *Business Wire*

Die Moral von DevOps: Es gibt einen besseren Weg

In den vorigen Abschnitten haben wir die Probleme und negativen Konsequenzen des Status quo aufgrund des zentralen chronischen Konflikts beschrieben – von der Unmöglichkeit, Firmenziele zu erreichen, bis hin zum Schaden, den wir bei den Mitmenschen verursachen. Indem DevOps diese Probleme löst, ermöglicht es uns verblüffend gut, gleichzeitig die Produktivität in der Firma zu verbessern, die Ziele der verschiedenen funktionalen technischen Rollen zu erreichen (zum Beispiel Entwicklung, QA, IT Operations, Infosec) und die Arbeitsbedingungen zu verbessern.

Diese aufregende und seltene Kombination erklärt vielleicht, warum DevOps bei so vielen Menschen in so kurzer Zeit so viel Begeisterung und Enthusiasmus ausgelöst hat – unter anderem bei Technologie-Führern, Entwicklern und vielen anderen im Software-Ökosystem.

Die Abwärtsspirale mit DevOps aufbrechen

Idealerweise implementieren kleine Entwicklerteams unabhängig voneinander ihre Features, überprüfen die Korrektheit in produktivähnlichen Umgebungen und lassen ihren Code dann schnell, sicher und zuverlässig in die echte Produktivumgebung deployen. Code-Deployments sind reine Routine, und es lässt sich vorhersagen, was passieren wird. Statt mit den Deployments freitags um Mitternacht zu beginnen und das ganze Wochenende damit zu verbringen, sie über die Bühne zu bringen, werden sie an normalen Arbeitstagen zu den normalen Arbeitszeiten ausgeführt, wenn sowieso jeder im Büro ist. Die Kunden bekommen es gar nicht mit – es sei denn, sie entdecken neue Features und Fehlerkorrekturen, über die sie sich freuen. Und durch das Deployen des Codes während der Arbeitszeit kann IT Operations das erste Mal seit Jahrzehnten wie jeder andere auch zu normalen Arbeitszeiten tätig sein.

Durch das Erstellen schneller Feedback-Schleifen für jeden Prozessschritt kann jeder sofort die Ergebnisse seiner Aktivitäten sehen. Wann immer Änderungen in die Versionsverwaltung eingecheckt werden, laufen schnelle, automatisierte Tests in einer produktionsähnlichen Umgebung, sodass man stets sicher sein kann, dass Code und Umgebung wie gedacht funktionieren und sich alles immer in einem sicheren und auslieferbaren Zustand befindet.

Automatisierte Tests helfen den Entwicklern dabei, ihre Fehler schnell zu entdecken (normalerweise innerhalb von Minuten). Dies führt zu schnelleren Korrekturen und direkterem Lernen – was so nicht möglich ist, wenn man Fehler erst sechs Monate später während der Integrationstests entdeckt. Dann sind nämlich die gedanklichen Verbindungen zwischen Ursache und Wirkung schon lange vergessen. Statt technische Schulden anzuhäufen, werden Probleme behoben, sobald sie entdeckt werden, und notfalls wird die gesamte Firma dazu mobilisiert, da globale Ziele gegenüber den lokalen Zielen Vorrang haben.

Allgegenwärtige Produktiv-Telemetriedaten in unseren Code- und Produktivumgebungen stellen sicher, dass Probleme erkannt und schnell gelöst werden, und bestätigen, dass alles wie gewünscht funktioniert und die Kunden aus der von uns erstellten Software Wert schöpfen können.

In diesem Szenario fühlt sich jeder produktiv – die Architektur erlaubt es kleinen Teams, sicher und von der Arbeit anderer architektonisch entkoppelt zu arbeiten. Sie nutzen Self-Service-Plattformen, die die gesamte Erfahrung von Operations und Information Security großräumig zum Einsatz bringen. Statt jeden dauernd warten zu lassen und so viel zu spät erst Korrekturen vornehmen zu können, arbeiten die Teams unabhängig und an kleinen Einheiten, die regelmäßig neue Features für den Kunden bringen.

Selbst anspruchsvolle Produkt- und Feature-Releases werden durch Dark-Launch-Techniken zur Routine. Lange vor dem Launch-Datum bringen wir den gesamten benötigten Code für das Feature in die Produktivumgebung – unsichtbar für jeden außer den internen Mitarbeitern und kleinen Gruppen echter Anwender, sodass wir das Feature testen und weiterentwickeln können, bis es das gewünschte Geschäftsziel erfüllt.

Und statt Tage oder Wochen mit Brandbekämpfung zu verbringen, damit die neue Funktionalität auch das tut, was sie soll, legen wir nur einen Feature-Schalter um oder passen eine Konfigurationseinstellung an. Diese kleine Änderung macht das neue Feature für immer größere Kundengruppen sichtbar und lässt sich automatisch zurücknehmen, wenn etwas schiefgeht. Im Ergebnis sind unsere Releases kontrolliert, planbar, umkehrbar, und sie verursachen weniger Stress.

Nicht nur Feature-Releases laufen ruhiger ab – alle Arten von Problemen werden schnell gefunden und behoben, wenn sie kleiner, günstiger und einfacher korrigiert werden können. Mit jedem Fix lernt die ganze Firma, wir können verhindern, dass das Problem wieder auftaucht, und ähnliche Probleme lassen sich in Zukunft schneller erkennen und fixen.

Zudem lernt jeder dauerhaft dazu, und es wird eine hypothesengetriebene Kultur unterstützt, in der wissenschaftliche Methoden genutzt werden, um sicherzustellen, dass nichts als gegeben angenommen wird. Ohne Messung geschieht nichts, und wir behandeln Produktentwicklung und Prozessverbesserungen als Experimente.

Da wir keine Zeit verplempern wollen – weder unsere noch die von anderen –, verbringen wir nicht Jahre damit, Features zu bauen, die unsere Kunden nicht wollen, Code zu deployen, der nicht funktioniert, oder etwas zu reparieren, das gar nicht die Ursache unseres Problems ist.

Weil wir danach streben, Ziele zu erreichen, bauen wir langfristige Teams auf, die dafür verantwortlich sind, das auch umzusetzen. Statt Projektteams zu bilden, in denen die Entwickler nach jedem Release neu zugewiesen und verschoben werden

und dabei niemals Feedback zu ihrer Arbeit erhalten, lassen wir die Teams intakt, sodass sie weiter iterieren und sich verbessern können und dabei die gewonnene Erfahrung nutzen, um ihre Ziele besser zu erreichen. Das gilt für die Produktteams, die Probleme für externe Kunden lösen, aber genauso auch für die internen Plattformteams, die anderen Teams dabei helfen, produktiver, sicherer und zuverlässiger sein zu können.

Statt einer Kultur der Angst bilden wir eine Kultur des Vertrauens und der Zusammenarbeit, bei der Mitarbeiter dafür belohnt werden, dass sie Risiken eingehen. Sie können angstfrei über Probleme sprechen, statt sie in der Schublade verstecken zu müssen – schließlich müssen wir die Probleme sehen können, um sie zu lösen.

Und weil jeder für die Qualität seiner Arbeit verantwortlich ist, baut auch jeder dafür automatisierte Tests und nutzt Peer Reviews, um die Bestätigung zu erhalten, dass Probleme so früh angegangen werden, dass sie keinen Kunden erreichen. Diese Prozesse schwächen im Gegensatz zu irgendwelchen Genehmigungen durch nicht involvierte Instanzen Risiken ab, sodass wir schnell, zuverlässig und sicher Werte erzeugen und bereitstellen können – und dabei selbst skeptische Auditoren davon überzeugen, dass wir ein effektives System der internen Kontrolle besitzen.

Sollte doch etwas schiefgehen, führen wir Post-Mortem-Analysen ohne Schuldzuweisungen durch – nicht, um auf jemanden einzudreschen, sondern, um besser zu verstehen, was den Unfall verursacht hat und wie man es verhindert. Dieses Ritual verstärkt den Lerneffekt nochmals. Auch halten wir interne Technologie-Konferenzen ab, um unser Wissen zu verbreiten und sicherzustellen, dass jedermann dauerhaft lernt und lehrt.

Weil wir uns um Qualität sorgen, bringen wir sogar Defekte in unsere Produktivumgebung ein, um geplant herauszufinden, wie unser System in solchen Situationen reagiert. Wir führen Übungen durch, um größere Ausfälle durchzuspielen, schießen per Zufall in der Produktivumgebung Prozesse oder ganze Server ab und spielen mit Netzwerklatenzen und anderen gemeinen Dingen, um sicherzustellen, dass unsere Umgebung widerstandsfähiger wird und die ganze Firma lernt und sich verbessert.

In dieser Welt ist jeder für seine Arbeit verantwortlich – unabhängig von seiner Rolle in der Technologie-Organisation. Jeder ist sich bewusst, dass seine Arbeit wichtig ist und zu den Firmenzielen beiträgt; die entspannte Arbeitsumgebung und der Firmenerfolg im Markt sind das Resultat.

Der geschäftliche Wert von DevOps

Für den geschäftlichen Wert von DevOps haben wir deutliche Hinweise. In den Jahren 2013 bis 2016 haben wir als Teil des *State of DevOps Report* der Puppet Labs, zu dem die Mitautoren Nicole Forsgren, Jez Humble und Gene Kim beitrugen, Daten von über 25.000 Technologie-Experten gesammelt, um den Zustand

und das Verhalten von Firmen in allen Stadien einer DevOps-Umsetzung besser zu verstehen.[15]

Die erste Überraschung aus diesen Daten war, wie sehr leistungsstarke Firmen mit DevOps-Praktiken den nicht so leistungsstarken Firmen in den folgenden Bereichen voraus sind:[16]

- Durchsatzmetriken
  - Code- und Änderungs-Deployments (30-mal häufiger)
  - Vorbereitungszeit für Code- und Änderungs-Deployments (200-mal schneller)
- Zuverlässigkeitsmetriken
  - Produktiv-Deployments (60-fach höhere Erfolgsrate)
  - Durchschnittsdauer, um Services wiederherzustellen (168 Mal schneller)
- Firmenperformancemetriken
  - Erreichen von Produktivitäts-, Marktanteils- und Profitabilitätszielen (200-fach höhere Wahrscheinlichkeit)
  - Wachstum der Marktkapitalisierung (50 % höher in drei Jahren)

Mit anderen Worten: Leistungsfähige Firmen waren sowohl agiler als auch zuverlässiger. Die empirischen Daten zeigen, dass wir mit DevOps den zentralen chronischen Konflikt aufbrechen können. Der Code wurde 30 Mal häufiger deployt, und die Dauer von »Code eingecheckt« bis »läuft erfolgreich in der Produktivumgebung« war 200 Mal kürzer – die Vorlaufzeiten wurden bei den leistungsfähigen Firmen in Minuten oder Stunden gemessen, während es bei den schlechten Firmen Wochen, Monate oder sogar Quartale waren.

Zudem ist die Wahrscheinlichkeit bei den leistungsstarken Firmen doppelt so groß, dass sie ihre Profitabilitäts-, Marktanteils- und Produktivitätsziele übererfüllen. Und für die Unternehmen, die am Aktienmarkt gehandelt werden, haben wir herausgefunden, dass die leistungsstarken Firmen eine in drei Jahren um 50 % stärker gewachsene Marktkapitalisierung haben. Auch die Mitarbeiterzufriedenheit war größer, die Burn-out-Rate geringer, und die Mitarbeiter würden ihren Arbeitgeber 2,2-mal eher an Freunde empfehlen.[17] Leistungsstarke Firmen haben ebenfalls bessere Ergebnisse bei der Information Security. Durch das Einbinden von Sicherheitszielen auf allen Stufen des Entwicklungs- und Operations-Prozesses wurden 50 % weniger Zeit für das Lösen von Sicherheitsproblemen aufgewandt.

15 Der *State of DevOps Report* wurde seitdem jedes Jahr neu erstellt. Zusätzlich wurden die zentralen Ergebnisse der Reports 2013–2018 im Buch *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations* herausgegeben.

16 Kersten, »IT Revolution und PwC«, *2015 State of DevOps Report*

17 Gemessen durch den *Employee Net Promoter Score* (eNPS). Dies ist eine wichtige Erkenntnis, da Untersuchungen gezeigt haben, dass »bei Firmen mit stark engagierten Mitarbeitern die Umsätze zweieinhalbmal stärker wachsen als bei solchen mit schlechtem Engagementindex. Und [börsennotierte] Aktien von Firmen mit einer auf großem Vertrauen beruhenden Arbeitsumgebung haben die Branchenindizes von 1997 bis 2011 um den Faktor 3 übertroffen.« (Azzarello, Debruyne und Mottura, »The Chemistry of Enthusiasm«)

DevOps hilft dabei, die Entwicklerproduktivität skalierbar zu machen

Wenn wir mehr Entwickler beschäftigen, sinkt häufig die Produktivität des einzelnen Mitarbeiters deutlich – aufgrund des Overheads durch Kommunikation, Integration und Testen. Das wird in Frederick Brooks berühmtem Buch *Vom Mythos des Mann-Monats* beschrieben, in dem er erläutert, dass das Hinzufügen von mehr Entwicklern bei Projekten, die drohen, nicht rechtzeitig fertig zu werden, nicht nur die Produktivität der einzelnen Entwickler verringert, sondern auch die Gesamtproduktivität drückt.[18]

DevOps zeigt uns andererseits, dass bei der richtigen Architektur, den richtigen technischen Praktiken und den richtigen kulturellen Normen kleine Teams mit Entwicklern schnell, zuverlässig und unabhängig voneinander Änderungen für die Produktivumgebung entwickeln, integrieren, testen und deployen können.

Randy Shoup, früher Director of Engineering bei Google, hat beobachtet, dass große Firmen, die DevOps verwenden, zwar »Tausende Entwickler einsetzen, Architektur und Praktiken es aber ermöglichen, dass kleine Teams unglaublich produktiv sein können – als ob es sich um ein Start-up handeln würde.«[19]

Der *State of DevOps Report 2015* hat nicht nur die »Deploys pro Tag« ermittelt, sondern auch die »Deploys pro Tag pro Entwickler«.[20] Wir hatten die Hypothese aufgestellt, dass leistungsfähige Firmen die Anzahl ihrer Deployments skalieren lassen könnten, wenn die Teamgröße wachsen würde.

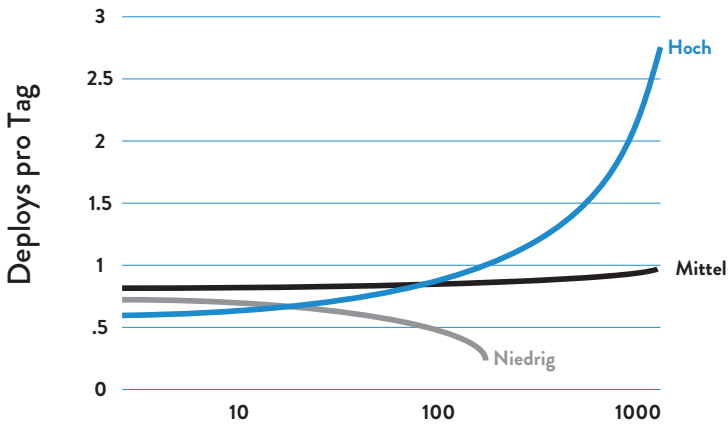


Abbildung 1: Deployments pro Tag vs. Anzahl Entwickler (Quelle: Puppet Labs, 2015 State of DevOps Report[21])

18 Brooks, *The Mythical Man-Month*

19 Kim et al., »Exploring the Uncharted Territory of Microservices

20 Kersten, »IT Revolution und PwC«, *2015 State of DevOps Report*

21 Es sind nur Firmen enthalten, die mindestens einmal pro Tag deployen.

Und das haben wir auch messen können. In Abbildung 1 sehen Sie, dass bei Low-Performern die Deploys pro Tag pro Entwickler bei wachsender Teamgröße weniger werden. Bei mittelguten Firmen bleiben sie gleich, bei leistungsstarken Firmen wächst diese Zahl linear. Anders gesagt: Firmen, die DevOps einführen, können durch das Einstellen von mehr Entwicklern die Anzahl der Deployments pro Tag erhöhen, wie das zum Beispiel bei Google, Amazon und Netflix der Fall ist.[22]

Die Universalität der Lösung

Eines der einflussreichsten Bücher der Lean-Manufacturing-Bewegung ist *Das Ziel: Ein Roman über Prozessoptimierung* von Dr. Eliyahu M. Goldratt aus dem Jahr 1984. Es beeinflusste eine ganze Generation von Fabrikmanagern auf der ganzen Welt. Dabei handelt es sich um die Geschichte eines Werkleiters, der seine Kosten und die ständig verspäteten Lieferungen in den Griff bekommen muss. Er hat dafür nur 90 Tage Zeit, ansonsten wird seine Fabrik geschlossen.

Ein paar Jahre später hat Dr. Goldratt über die Briefe geschrieben, die er als Reaktion auf *Das Ziel* erhalten hat. Oft stand darin so etwas wie: »Sie haben sich offensichtlich in unserer Fabrik versteckt, denn Sie haben mein Leben [als Fabrikmanager] sehr genau beschrieben ...«[23] Wichtiger ist aber: Diese Briefe haben gezeigt, dass die Menschen dazu in der Lage waren, die in dem Buch beschriebenen deutlichen Performanceverbesserungen in ihren eigenen Arbeitsumgebungen reproduzieren zu können.

Projekt Phoenix: Der Roman über IT und DevOps – Neue Erfolgsstrategien für Ihre Firma aus dem Jahr 2013, geschrieben von Gene Kim, Kevin Behr und George Spafford, orientiert sich an *Das Ziel*. Dabei handelt es sich um die Geschichte eines IT-Managers, der sich all den typischen Problemen gegenübersieht, die in IT-Organisationen verbreitet sind: ein das Budget sprengende und nicht im Zeitplan liegende Projekt, das unbedingt den Markt erreichen muss, damit die Firma überleben kann. Er hat mit katastrophalen Deployments zu tun, mit Problemen bei der Verfügbarkeit, bei Sicherheit und Compliance und so weiter. Letztendlich nutzen er und sein Team DevOps-Prinzipien und -Praktiken, um diese Herausforderungen zu meistern und der Firma dabei zu helfen, auf dem Markt wieder erfolgreich zu sein. Zudem zeigt das Buch, wie DevOps-Praktiken die Arbeitsumgebung für das Team verbessern, den Stress reduzieren und für eine höhere Zufriedenheit sorgen, weil die Praktiker in den Prozess eingebunden werden.

Analog zu *Das Ziel* gibt es deutliche Hinweise auf die Universalität des Problems und der Lösungen, die in *Projekt Phoenix* beschrieben sind. Dies sind ein paar Ausschnitte aus den Amazon-Bewertungen zum englischsprachigen Original: »Ich er-

22 Ein weiteres extremes Beispiel ist Amazon. 2011 wurden dort ungefähr 7.000 Deployments pro Tag durchgeführt. 2015 sind es 130.000 pro Tag. (Jon Jenkins, »Velocity Culture«; Exner, »Transforming Software Development«)

23 Goldratt, *Beyond the Goal*

kenne die Charaktere aus *Projekt Phoenix* wieder ... Vermutlich habe ich die meisten von ihnen im Laufe meiner Karriere schon getroffen«, »Wenn Sie jemals in irgendeinem Bereich von IT, DevOps oder Infosec gearbeitet haben, werden Sie dieses Buch nachvollziehen können« oder »Es gibt keine Figur in *Projekt Phoenix*, in der ich nicht mich oder jemanden aus meinem Umfeld wiedererkennen kann ... ganz zu schweigen von den Problemen, mit denen diese Figuren zu kämpfen haben.«[24]

In diesem Buch werden wir erläutern, wie Sie die in *Projekt Phoenix* beschriebenen Änderungen replizieren können, aber auch viele Fallstudien vorstellen, wie andere Firmen DevOps-Prinzipien und -Praktiken eingesetzt haben, um ebenfalls diese Ergebnisse zu erreichen.

Das DevOps-Handbuch: ein unentbehrlicher Leitfaden

Ziel des *DevOps-Handbuchs* ist, Ihnen die Theorien, Prinzipien und Praktiken vorzustellen, die Sie brauchen, um Ihre DevOps-Initiative erfolgreich starten und die gewünschten Ergebnisse erzielen zu können. Dieser Leitfaden basiert auf Jahrzehnten fundierter Management-Theorie, dem Untersuchen leistungsfähiger Technologie-Organisationen, unserer Beratung bei der Transformation solcher Organisationen und Forschungsarbeiten, die die Effektivität der beschriebenen DevOps-Praktiken überprüfen. Dazu kommen Gespräche mit relevanten Experten zum Thema und die Analyse von fast 100 Fallstudien, die auf dem *DevOps Enterprise Summit* vorgestellt wurden.

Dieses Buch behandelt in seinen sechs Teilen die DevOps-Theorien und -Prinzipien anhand der Drei Wege, einer besonderen Sicht auf die zugrunde liegende Theorie, die ursprünglich im *Projekt Phoenix* eingeführt wurde. Das *DevOps-Handbuch* ist für alle gedacht, die Anteil an der Technologie-Wertkette haben – sei es direkt oder indirekt. Typischerweise gehören dazu Produktmanagement, Entwicklung, QA, IT Operations und Information Security. Es soll aber auch Business- und Marketing-Führungskräften helfen, da von dort die meisten Technologie-Initiativen ausgehen.

Wir erwarten nicht, dass der Leser umfassendes Wissen über einen dieser Bereiche mitbringt – und ebenso wenig, dass er etwas über DevOps, Agile, ITIL, Lean oder Prozessverbesserung wissen muss. Jedes dieser Themen wird im Buch aufgegriffen und erläutert, wenn es ins Spiel kommt.

Wir möchten in jedem dieser Bereiche die Grundlagen der entscheidenden Konzepte vermitteln, damit Sie wissen, worum es geht, und um die notwendigen Begriffe bekannt zu machen, die Sie für die Kommunikation mit den Kollegen in der gesamten IT-Wertkette brauchen und mit denen Sie gemeinsame Ziele beschreiben können.

24 JGFLL, Review von *The Phoenix Project;* Townsend, Review von *The Phoenix Project*; Van Den Elzen, Review von *The Phoenix Project*

Dieses Buch wird auch für Führungskräfte und andere Stakeholder von Interesse sein, da sich diese immer mehr auf die Technologie-Organisation verlassen müssen, um ihre Ziele erreichen zu können.

Zudem soll dieses Buch Leser ansprechen, deren Firma all die beschriebenen Probleme nicht hat (zum Beispiel lange Vorlaufzeiten beim Deployment oder schmerzhafte Auslieferungen). Auch Leser in dieser glücklichen Position werden davon profitieren, die DevOps-Prinzipien zu verstehen – speziell die zu gemeinsamen Zielen, Feedback und kontinuierlichem Lernen.

In **Teil I** stellen wir kurz die Geschichte von DevOps vor und führen in die zugrunde liegende Theorie und die wichtigsten Themen ein, die sich aus dem über Jahrzehnte erworbenen Wissensschatz speisen. Dann erklären wir die obersten Prinzipien der Drei Wege: Flow, Feedback sowie fortlaufendes Lernen und Experimentieren.

In **Teil II** beschreiben wir, wie und wo man beginnt, und stellen Konzepte wie die Wertkette, organisatorische Designprinzipien und -muster, Veränderungsmuster und Fallstudien vor.

Teil III zeigt, wie wir den Flow durch den Aufbau der Grundlagen unserer Deployment-Pipeline beschleunigen: das Ermöglichen schneller und effektiver automatisierter Tests, Continuous Integration, Continuous Delivery und eine Architektur, die risikoarme Releases unterstützt.

In **Teil IV** wird besprochen, wie man das Feedback beschleunigt und verstärkt, indem man eine effektive Produktiv-Telemetrie aufsetzt, um Probleme zu erkennen und zu lösen, sie besser vorherzusehen und Ziele leichter zu erreichen. Feedback soll ermöglichen, dass Dev und Ops Änderungen zuverlässig deployen können. Dazu sollen A/B-Tests in die tägliche Arbeit integriert und Review- und Koordinierungsprozesse aufgesetzt werden, um die Qualität unserer Arbeit zu erhöhen.

Teil V beschreibt, wie wir das kontinuierliche Lernen und Experimentieren beschleunigen, indem wir eine Just Culture schaffen, lokale Erkenntnisse global bekannt machen und ausreichend Zeit einplanen, um das firmenweite Lernen und Verbessern zu ermöglichen.

In **Teil VI** zeigen wir schließlich, wie wir Sicherheit und Compliance sauber in unser Tagesgeschäft integrieren, indem wir Sicherheitsmaßnahmen in die gemeinsamen Quellcode-Repositories und Services einbinden, sie in unsere Deployment-Pipeline einfließen lassen, die Telemetrie verbessern, um das Erkennen von Problemen und die Reaktion darauf leichter zu machen, die Deployment-Pipeline schützen und Änderungsmanagement-Ziele anstreben.

Durch das Kodifizieren dieser Praktiken hoffen wir, die Übernahme von DevOps-Praktiken schneller gestalten zu können, den Erfolg von DevOps-Initiativen zu verbessern und die notwendige Aktivierungsenergie für DevOps-Transformationen zu verringern.

TEIL I

Die Drei Wege

In Teil I des *DevOps-Handbuchs* werden wir uns anschauen, wie die verschiedenen wichtigen Management- und Technologie-Bewegungen die Grundlage für die DevOps-Bewegung geschaffen haben. Wir beschreiben Wertketten, wir zeigen, wie DevOps das Ergebnis der Anwendung von Lean-Prinzipien auf die Technologie-Wertkette ist, und wir erklären die Drei Wege: Flow, Feedback sowie kontinuierliches Lernen und Experimentieren.

Der Hauptfokus in diesem Kapitel liegt auf:

- den Prinzipien des Flow, der das Ausliefern von Arbeit von der Entwicklung über Operations hin zu unseren Kunden beschleunigt,
- den Prinzipien des Feedbacks, durch das wir zuverlässigere Arbeitssysteme schaffen können, und
- den Prinzipien des kontinuierlichen Lernens und Experimentierens, mit denen eine Kultur des Vertrauens und ein wissenschaftlicher Ansatz unterstützt werden, durch die Risiken bei Verbesserungsmaßnahmen Teil unserer täglichen Arbeit werden.

Eine kurze Historie

DevOps und seine dazugehörigen technischen, architektonischen und kulturellen Praktiken sind das Ergebnis des Zusammenfassens diverser Philosophie- und Managementrichtungen. Während viele Firmen diese Prinzipien unabhängig voneinander entwickelten, ist es sehr hilfreich, zu verstehen, dass DevOps aus verschiedensten Bewegungen entstand – ein Phänomen, das Mitautor John Willis als »Konvergenz von DevOps« beschrieb. Hat man das verstanden, führt dies zu einem deutlich besseren Verständnis, und viele vorher unlogische Zusammenhänge klären sich auf. Lektionen, die in Jahrzehnten in der Herstellung, bei hoch zuverlässigen Firmen und Managementmodellen sowie in anderen Bereichen gelernt wurden, haben uns schließlich zu den DevOps-Praktiken gebracht, die wir heutzutage kennen.

DevOps ist das Ergebnis der Anwendung der besten Prinzipien aus den Bereichen der Herstellung und des Managements auf die IT-Wertkette. DevOps basiert auf Lean, der Theory of Constraints, dem Toyota Production System, Resilience Engineering, firmenweitem Lernen, einer Sicherheitskultur, menschlichen Faktoren und vielen anderen Dingen. Weitere Inspiration erhält DevOps aus Managementkulturen, die stark auf Vertrauen basieren, Servant Leadership sowie Organizational Change Management.

Die Ergebnisse sind erstklassige Qualität, Zuverlässigkeit, Stabilität und Sicherheit bei gleichzeitig geringeren Kosten und weniger Aufwand sowie ein beschleunigter Flow und verlässlicher Durchfluss in der technologischen Wertkette, zu der Produktmanagement, Entwicklung, QA, IT Operations und Infosec gehören.

Die Grundlagen von DevOps liegen durchaus in Lean, der Theory of Constraints und der Toyota-Kata-Bewegung, aber viele sehen DevOps auch als logische Fortsetzung der Agile-Software-Bewegung, die 2001 begann.

Die Lean-Bewegung

Techniken wie Value Stream Mapping, Kanban-Boards und Total Productive Maintenance wurden in den 1980er-Jahren für das Toyota Production System kodifiziert. 1997 begann das Lean Enterprise Institute damit, Anwendungen von Lean auf andere Wertketten zu untersuchen, wie zum Beispiel in der Serviceindustrie oder im Gesundheitswesen.

Zwei der wichtigsten Grundsätze von Lean bestehen in dem Glauben, dass es sich bei den Durchlaufzeiten bei der Herstellung, die zum Umwandeln von Rohmaterialien in fertige Güter gebraucht werden, um die besten Vorhersagewerte für die Qualität sowie die Kunden- und Mitarbeiterzufriedenheit handelt und dass eine der besten Voraussetzungen für kurze Durchlaufzeiten kleine Arbeitseinheiten sind.

Lean-Prinzipien konzentrieren sich darauf, wie mithilfe von systemischem Denken ein Wert für den Kunden geschaffen werden kann, indem Zielkonsequenz geschaffen, wissenschaftliches Denken unterstützt, Flow und Pull (statt Push) erzeugt, schon an der Quelle die Qualität sichergestellt, mit Bescheidenheit geführt und jeder einzelne Mitarbeiter respektiert wird.

Das Agile Manifesto

Das Agile Manifesto wurde 2001 von 17 führenden Köpfen der Softwareentwicklung geschaffen, die sich damals mit »Lightweight Methods« befassten. Sie wollten einen Satz von Werten und Prinzipien, die die Vorteile dieser eher adaptiven Methoden erfassten, gegen die vorherrschenden schwergewichtigen Softwareentwicklungsprozesse, wie die Wasserfall-Entwicklung, und Methoden, wie den Rational Unified Process, stellen.

Ein wichtiges Prinzip war, »regelmäßig alle paar Wochen oder Monate funktionierende Software auszuliefern – wobei der kürzere Zeitraum vorzuziehen ist«.[1] Damit wurde der Wunsch nach kleineren Batchgrößen und inkrementellen Releases anstelle von großen Big-Bang-Releases hervorgehoben. Andere Prinzipien unterstrichen den Bedarf nach kleinen, selbst motivierten Teams, die in einem Managementmodell arbeiten, das stark auf Vertrauen setzt.

Agile wird mit einer massiv wachsenden Produktivität und Reaktionsfähigkeit in vielen Entwicklungseinheiten in Verbindung gebracht. Und interessanterweise traten viele der entscheidenden Momente in der Geschichte von DevOps gerade in der Agile-Community oder auf Agile-Konferenzen auf.

Agile Infrastructure und Velocity-Bewegung

Patrick Debois und Andrew Shafer hielten 2008 auf der Agile-Konferenz in Toronto eine »Birds of a Feather«-Session über das Anwenden von Agile-Prinzipien auf Infrastruktur statt auf Anwendungscode ab. (In diesen frühen Tagen wurde das als *Agile System Administration* bezeichnet.) Auch wenn sie die einzigen Teilnehmer waren, gewannen sie schnell Gleichgesinnte hinzu, unter anderem den Mitautor John Willis.

Continuous Learning

Ungefähr zur gleichen Zeit begannen ein paar Akademiker damit, Systemadministratoren daraufhin zu beobachten, wie sie Entwicklungsprinzipien auf ihre Arbeit anwendeten und wie das ihre Performance beeinflusste. Zu den führenden Experten gehörte damals eine Gruppe von IBM Research, deren Ethnografen von Dr. Eben Haber, Dr. Eser Kandogan und Dr. Paul Magio geleitet wurden. Die Gruppe wurde in den Jahren 2007 bis 2009 für quantitative Verhaltensstudien um die Koautorin Dr. Nicole Forsgren erweitert. Nicole leitete dann in den Jahren 2014 bis 2019 die Forschung zu den *State of DevOps Reports* – der Branchenstandardforschung zu Praktiken und Fähigkeiten, die Software-Delivery und -Performance voranbringen. Diese Reports wurden von Puppet und DORA veröffentlicht.

Im Jahr darauf hielten John Allspaw und Paul Hammond auf der Velocity-Konferenz den bahnbrechenden Vortrag »10 Deploys per Day: Dev and Ops Cooperation at Flickr«. Dort beschrieben sie, wie Dev und Ops gemeinsame Ziele aufgestellt hatten und wie mithilfe von Continuous-Integration-Praktiken das Deployment ein Teil des Tagesgeschäfts eines jeden wurde. Augenzeugen berichten, dass jedem, der diesem Vortrag gelauscht hat, klar war, dass hier etwas Grundlegendes und historisch Wichtiges geschah.

1 Beck et al., »Twelve Principles of Agile Software«

Patrick Debois war von der Idee von Allspaw und Hammond so begeistert, dass er im gleichen Jahr in Gent die ersten DevOpsDays initiierte. Damit war der Begriff »DevOps« geboren.

Die Continuous-Delivery-Bewegung

Jez Humble und David Farley nutzten die Entwicklungspraktiken von Continuous Build, Test und Integration und entwickelten daraus das Konzept des Continuous Delivery, das die Rolle einer »Deployment-Pipeline« definierte, um sicherzustellen, dass sich Code und Infrastruktur immer in einem auslieferbaren Zustand befinden und dass der gesamte eingecheckte Code sicher in die Produktivumgebung transportiert werden kann. Diese Idee wurde erstmals auf der Agile-Konferenz im Jahr 2006 präsentiert und unabhängig davon 2009 von Tim Fitz in dem Blogartikel »Continuous Deployment« auf seiner Website entwickelt.[2]

Toyota Kata

2009 schrieb Mike Rother *Die Kata des Weltmarktführers: Toyotas Erfolgsmethoden*, in dem er seine 20-jährige Erfahrung mit dem Toyota Production System zusammenfasst und es kodifiziert. Er war einer der Uniabsolventen, die mit Führungskräften von GM zum Besuch der Toyota-Fabriken mitflog, und er half beim Entwickeln des Lean Toolkit. Allerdings war er verwirrt, weil keine der Firmen, die diese Praktiken übernahmen, die Leistung erbringen konnte, die bei Toyota beobachtet wurde.

Mike schloss daraus, dass die Lean-Community die wichtigste Praktik von allen übersehen hatte, die er *Verbesserungs-Kata* nannte.[3] Er erklärte, dass jede Firma ihre Arbeitsroutinen nutzt und dass für das Verbesserungs-Kata Strukturen für die tägliche und ständige Verbesserungsarbeit geschaffen werden müssen, da gerade die tägliche Praxis die Ergebnisse verbessert. Der kontinuierliche Kreislauf aus dem Definieren von gewünschten zukünftigen Zuständen, dem Setzen von getakteten Zielen und dem kontinuierlichen Verbessern der täglichen Arbeit hat Toyota so weit gebracht.

Im Rest von Teil I schauen wir uns Wertketten an, sehen, wie Lean-Prinzipien auf die Technologie-Wertkette angewendet werden können, und beschreiben die Drei Wege aus Flow, Feedback sowie kontinuierlichem Lernen und Experimentieren.

---

2 DevOps nutzt auch die von Dr. Mark Burgess, Luke Kanies sowie Adam Jacob entwickelten Praktiken von *Infrastructure as Code* und baut diese aus. Dabei wird die Arbeit von Operations automatisiert und wie Anwendungscode behandelt, sodass moderne Entwicklungspraktiken auf den gesamten Entwicklungsstrom angewendet werden können. Damit wird ein schneller Deployment-Flow ermöglicht, zu dem auch *Continuous Integration* (von Grady Booch entworfen und eine der entscheidenden zwölf Praktiken des Extreme Programming), *Continuous Delivery* (entwickelt von Jez Humble und David Farley) und *Continuous Deployment* (entstanden bei Etsy, Wealthfront und durch Eric Ries' Arbeit bei IMVU) gehören.

3 Rother, *Toyota Kata*, Part III

KAPITEL 1

Agile, Continuous Delivery und die Drei Wege

In diesem Kapitel wird eine Einführung in die zugrunde liegende Theorie des Lean Manufacturing gegeben, und mit den Drei Wegen werden auch die Prinzipien gezeigt, aus denen alle DevOps-Vorgehensweisen abgeleitet werden können.

Unser Fokus liegt hier vor allem auf der Theorie und den Prinzipien. Wir beschreiben die Erfahrungen aus vielen Jahrzehnten in der Herstellung, in hoch zuverlässigen Firmen, in Managementmodellen, die auf starkem Vertrauen basieren, und andere Dinge, aus denen DevOps-Praktiken abgeleitet wurden. Die resultierenden konkreten Prinzipien und Muster sowie ihre praktische Anwendung auf die Technologie-Wertkette werden in den restlichen Kapiteln dieses Buchs vorgestellt.

Die Wertkette in der Herstellung

Eines der fundamentalen Konzepte in Lean ist die *Wertkette* oder der *Wertfluss*. Wir werden sie zuerst im Herstellungskontext definieren und basierend darauf entwickeln, wie sie auf DevOps und die Technologie-Wertkette angewendet wird.

Karen Martin und Mike Osterling definieren die Wertkette in ihrem Buch *Value Stream Mapping: How to Visualize Work and Align Leadership for Organizational Transformation* als die »Folge von Aktivitäten, die eine Firma vornimmt, um eine Kundenanforderung zu erfüllen« oder die »Folge von Aktivitäten, die erforderlich sind, um eine Ware oder einen Service für einen Kunden zu entwerfen, zu produzieren und auszuliefern, wozu auch der doppelte Fluss von Informationen und Material gehört«.[1]

In herstellenden Betrieben ist die Wertkette häufig einfach zu erkennen und zu beobachten: Sie beginnt, wenn eine Kundenbestellung eintrifft und die Rohmaterialien in die Fabrikhalle gebracht werden. Um schnelle und planbare Durchlaufzeiten in einer Wertkette zu ermöglichen, konzentriert man sich im Allgemeinen unablässig darauf, einen reibungslosen und gleichmäßigen Arbeitsfluss zu erzeugen, indem

1 Martin und Osterling, *Value Stream Mapping*, Kap. 1

man verschiedene Techniken nutzt, zum Beispiel kleine Batchgrößen, einen geringen *Work in Process* (WIP), das Verhindern von doppelter Arbeit, indem keine fehlerhaften Elemente an folgende Arbeitsstationen weitergereicht werden, und das konstante Optimieren des Systems auf die globalen Ziele hin.

Die Technologie-Wertkette

Viele Prinzipien und Muster, die den schnellen Arbeitsfluss bei der Herstellung echter Güter ermöglichen, lassen sich auch auf die technologische Arbeit anwenden (und auch auf alle anderen Wissensarbeiten). In DevOps definieren wir unsere Technologie-Wertkette im Allgemeinen als den Prozess, der erforderlich ist, um eine Geschäftshypothese in einen durch Technologie unterstützten Service oder ein Feature umzuwandeln, der den Kunden einen Wert bietet.

Die Eingangsdaten für unseren Prozess sind das Formulieren eines Geschäftsziels, eines Konzepts, einer Idee oder Hypothese. Begonnen wird der Prozess, wenn wir die Arbeit in der Entwicklung annehmen und sie zu unserem bestätigten Backlog hinzufügen.

Von dort werden Entwicklungsteams, die einem typischen Agile- oder iterativen Prozess folgen, diese Idee sehr wahrscheinlich in User Stories und eine Form von Feature-Spezifikation umwandeln, die dann in Code für die zu bauende Anwendung oder den Service implementiert wird. Der Code wird nun ins Repository der Versionsverwaltung eingecheckt, in das jede Änderung integriert und mit dem Rest des Softwaresystems getestet wird.

Da Wert nur dann entsteht, wenn die Services auch produktiv laufen, müssen wir sicherstellen, dass wir nicht nur in schnellem Flow liefern, sondern unsere Deployments auch durchgeführt werden können, ohne Chaos und Unterbrechungen zu verursachen – also ohne Serviceausfälle, Beeinträchtigungen sowie Sicherheits- oder Compliance-Probleme.

Fokussieren auf die Deployment-Durchlaufzeit

In diesem Buch werden wir unsere Aufmerksamkeit auf die Deployment-Durchlaufzeit richten – und damit nur auf einen Teil der oben beschriebenen Wertkette. Die Wertkette beginnt, wenn ein Techniker[2] (Entwicklung, QA, IT Operations oder Infosec) eine Änderung in die Versionsverwaltung eincheckt. Sie endet, wenn diese Änderung erfolgreich in der Produktivumgebung läuft, dem Kunden Wert bietet und nützliches Feedback und Analysedaten liefert.

Die erste Arbeitsphase – Design und Entwicklung – ähnelt stark dem Lean Product Development. Sie ist sehr variabel und unklar, häufig ist viel Kreativität gefragt,

2 Wir schreiben hier von »Technikern«, weil wir uns damit später auf alle beziehen, die in unserer Wertkette arbeiten – nicht nur auf Entwickler.

und es wird Arbeit geleistet, die nie wieder getan werden muss. Das führt zu sehr unterschiedlichen Prozesszeiten. Den Gegensatz dazu bildet die zweite Phase, zu der Testen, Deployment und Operations gehören – diese ähnelt dem Lean Manufacturing. Hier sind Kreativität und Erfahrung gefragt, und man bemüht sich, möglichst planend und mechanistisch vorzugehen, um Arbeitsergebnisse mit minimaler Variabilität zu erhalten (zum Beispiel kurze und zuverlässige Durchlaufzeiten sowie möglichst keine Fehler).

Statt große Arbeitsblöcke nacheinander erst durch die Design-/Entwicklungsphase und dann die Test-/Operations-Phase laufen zu lassen (wie bei einem Wasserfallprozess mit großen Einheiten oder bei langlebigen Feature-Branches), ist unser Ziel hier, Test, Deployment und Operations parallel zu Design und Entwicklung durchzuführen, um einen schnellen Flow und hohe Qualität zu ermöglichen. Diese Methode ist erfolgreich, wenn wir in kleinen Batches arbeiten und in jedem Teil unserer Wertkette Qualität sicherstellen.[3]

Der Unterschied zwischen Durchlaufzeit und Verarbeitungszeit

In der Lean-Community ist die Durchlaufzeit (Lean Time) eine von zwei Kennzahlen, die im Allgemeinen für das Messen der Leistungsfähigkeit einer Wertkette genutzt wird. Die andere ist die Verarbeitungszeit (Processing Time, manchmal auch als Touch Time oder Task Time bezeichnet).[4]

Während die Durchlaufzeit beginnt, sobald die Anforderung gestellt wird, und endet, wenn diese Anforderung erfüllt ist, beginnt die Verarbeitungszeit erst mit dem Beginn der Arbeit für die Anforderung – es wird also die Zeit ignoriert, in der die Arbeit in der Queue herumdümpelt und darauf wartet, angefasst zu werden (siehe Abbildung 1-1).

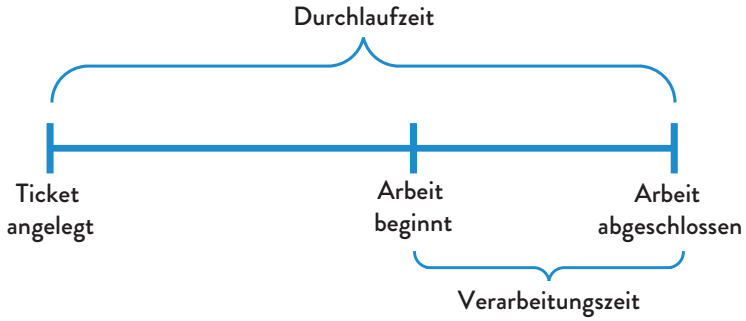


Abbildung 1-1: Durchlaufzeit vs. Verarbeitungszeit bei einem Deployment

3 Bei Techniken wie dem Test-Driven Development findet das Testen sogar schon statt, bevor die erste Codezeile entsteht.

4 In diesem Buch nutzen wir den Begriff *Verarbeitungszeit* – aus dem gleichen Grund wie Karen Martin und Mike Osterling: »Um eine Verwechslung zu vermeiden, verzichten wir auf den Begriff »Cycle Time«, da es für diesen eine Reihe von Definitionen gibt, die synonym zu Verarbeitungszeit und Ausgabegeschwindigkeit oder -frequenz sind (um nur ein paar zu nennen).« (Ebd., Kap. 3)

Da die Durchlaufzeit die für den Kunden relevante Zeit ist, konzentrieren wir uns im Allgemeinen bei unseren Versuchen zur Prozessverbesserung darauf und nicht auf die Verarbeitungszeit. Aber das Verhältnis von Verarbeitungszeit zu Durchlaufzeit ist eine wichtige Kennzahl für die Effizienz – um einen schnellen Flow und kurze Durchlaufzeiten zu erhalten, müssen wir uns fast immer auch darum bemühen, die Zeit zu verringern, die unsere Arbeit in einer Queue wartet.

Das häufige Szenario: Deployment-Durchläufe dauern Monate

Viele Teams und Organisationen befinden sich in Situationen, in denen die Deployment-Durchlaufzeiten im Bereich von Monaten liegen. Das gilt besonders für große, komplexe Organisationen, die mit eng gekoppelten monolithischen Systemen arbeiten, häufig nur eine dürftige Integrationstestumgebung besitzen, lange Durchlaufzeiten für Test- und Produktivumgebungen haben, stark auf manuelles Testen angewiesen sind und viele Bestätigungsprozesse durchlaufen müssen. Wenn so eine Situation vorliegt, kann die Wertkette wie in Abbildung 1-2 aussehen.

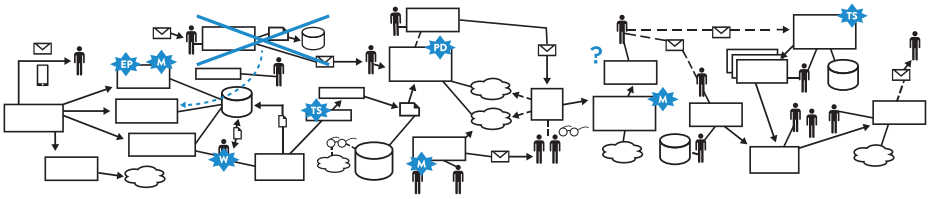


Abbildung 1-2: Eine Technologie-Wertkette mit einer Deployment-Durchlaufzeit von drei Monaten (Quelle: Damon Edwards, »DevOps Kaizen«, 2015)

Haben wir lange Deployment-Durchlaufzeiten, sind an fast jeder Stelle dieser Wertkette immer wieder Heldentaten notwendig. Wir erkennen vielleicht erst am Ende des Projekts beim Zusammenbringen aller Änderungen des Entwicklerteams, dass nichts funktioniert und sich der Code nicht mehr sauber bauen lässt oder irgendeinen unserer Tests nicht besteht. Das Beheben jedes einzelnen Problems kann Tage oder Wochen dauern, weil herausgefunden werden muss, wer einen Fehler eingebaut hat und wie man diesen lösen kann – und trotzdem sind die Ergebnisse für den Kunden mager.

Unser DevOps-Ideal: Deployment-Durchlaufzeiten von Minuten

Im DevOps-Ideal erhalten Entwickler schnell und dauerhaft Feedback zu ihrer Arbeit. Dadurch können sie schnell und unabhängig voneinander ihren Code implementieren, integrieren und validieren, und es ist ihnen möglich, den Code in die Produktivumgebung zu deployen (entweder selbst oder durch jemand anderen).

Wir erreichen das, indem wir kontinuierlich kleine Codeänderungen in unsere Versionsverwaltung einchecken, automatische und explorative Tests dagegen laufen und die Änderungen dann produktiv gehen lassen. So können wir sehr sicher sein, dass alles wie geplant auch in der Produktivumgebung läuft und Probleme schnell erkannt und behoben werden können.

Am einfachsten ist das zu erreichen durch eine modulare Architektur mit sauberer Kapselung und nur lose verbundenen Komponenten. Kleine Teams können so mit hoher Autonomie arbeiten, und Fehler bleiben klein und eingegrenzt und verursachen keine globalen Störungen.

In diesem Szenario lässt sich die Deployment-Durchlaufzeit in Minuten oder im schlimmsten Fall in Stunden messen. Unsere Wertkette sieht dann aus wie in Abbildung 1-3.

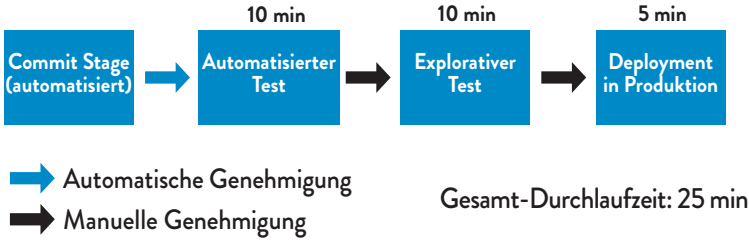


Abbildung 1-3: Eine Technologie-Wertkette mit einer Durchlaufzeit im Minutenbereich

%C/A als Kennzahl für die Nacharbeit

Neben der Durchlaufzeit und der Verarbeitungszeit ist die dritte Kennzahl in der Technologie-Wertkette der Prozentwert »Complete and Accurate« (%C/A). Diese Zahl zeigt die Qualität des Ergebnisses jedes Schritts in unserer Wertkette an.

Karen Martin und Mike Osterling schreiben: »%C/A kann man ermitteln, indem man die Kunden entlang der Wertkette fragt, wie viel Prozent der empfangenen Arbeit für sie ›wie gewünscht‹ nutzbar sind – sie also ihre eigene Arbeit erledigen können, ohne die erhaltenen Daten korrigieren oder ergänzen zu müssen oder nach Informationen zu fragen, die klarer sein könnten und sollten.«[5]

Continuous Learning: Flow-Metriken zum Messen des Erzeugens von Business Value

Wenn man den End-to-End-Wert einer Wertkette misst, ist es wichtig, sich von Proxy-Metriken fernzuhalten (zum Beispiel die Anzahl der Codezeilen zu ermitteln oder sich nur auf die Frequenz der Deployments zu verlassen). Diese Metriken können zwar lokale Optimierungen aufdecken, aber sie haben keine direkte Verbindung zu den Geschäftsergebnissen wie zum Beispiel dem Umsatz.

Der Einsatz von Flow-Metriken bietet die Möglichkeit, sich den End-to-End-Wert Ihrer Software-Delivery anzuschauen und Softwareprodukte und

5 Ebd.

Wertketten genauso sichtbar zu machen wie Produkte in einer Fertigungslinie. In seinem Buch *Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework* beschreibt Dr. Mik Kersten folgende Flow-Metriken: Flow-Geschwindigkeit, Flow-Effizienz, Flow-Zeit, Flow-Last und Flow-Verteilung:[6]

- **Flow-Geschwindigkeit**: Anzahl der Flow-Elemente (zum Beispiel Arbeitspakete), die in einer gegebenen Zeit abgeschlossen werden.
- **Flow-Effizienz**: Das Verhältnis der Flow-Elemente, an denen aktiv gearbeitet wird, zur gesamten vergangenen Zeit. Damit erkennen Sie Ineffizienzen, wie zum Beispiel lange Wartezeiten, und helfen Teams dabei, herauszufinden, ob sich der Arbeitsfluss in einem Wartezustand befindet oder nicht.
- **Flow-Zeit**: Eine Einheit mit Geschäftswert, die von einem Stakeholder durch die Wertkette eines Produkts befördert wird (zum Beispiel Features, Defekte, Risiken oder Schulden). Das hilft Teams dabei, herauszufinden, ob die Time to Value kürzer wird.
- **Flow-Last**: Anzahl der aktiven oder wartenden Flow-Elemente in einer Wertkette. Das entspricht dem Messen der *Work in Progress* (WIP) basierend auf Flow-Elementen. Eine hohe Flow-Last führt zu Ineffizienzen und zu einer verringerten Flow-Geschwindigkeit oder einer steigenden Flow-Zeit. Teams können damit ermitteln, ob die Anforderungen größer als die Kapazität sind.
- **Flow-Verteilung**: Der Anteil jeder Art von Flow-Element in einer Wertkette. Jede Wertkette kann diese Abhängigkeiten verfolgen und an ihre Bedürfnisse anpassen, um den gelieferten Business-Wert zu maximieren.

Die Drei Wege: Die Prinzipien als Basis von DevOps

In *Projekt Phoenix* werden die Drei Wege als zugrunde liegende Prinzipien vorgestellt, die die Verhaltensweisen und Muster bei DevOps erklären (siehe Abbildung 1-4).

Der Erste Weg ermöglicht einen schnelleren Flow der Arbeit von links nach rechts – von der Entwicklung über Operations zum Kunden. Um den Fluss maximieren zu können, müssen wir die Arbeit sichtbar machen, unsere Batchgrößen und Arbeitsintervalle reduzieren, qualitativ hochwertig sein, indem wir verhindern, dass Fehler an die nächsten Arbeitsstationen weitergereicht werden, und kontinuierlich auf die globalen Ziele hin optimieren.

6 Kersten, *Project to Product*

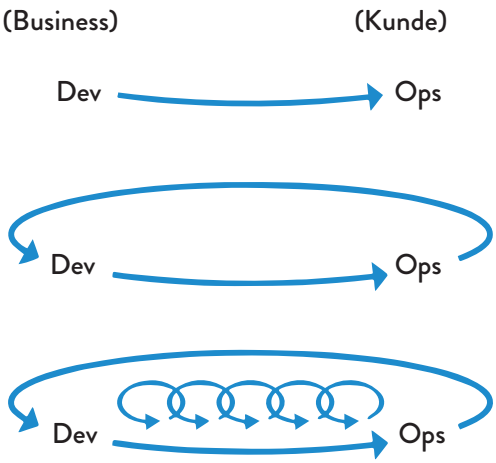


Abbildung 1-4: Die Drei Wege (Quelle: Gene Kim: »The Three Ways: The Principles Underpinning DevOps«, ITRevolution.com (Blog), 22. August 2021, http://itrevolution.com/the-three-ways-principles-underpinning-devops/)

Indem wir den Flow durch die Technologie-Wertkette beschleunigen, verringern wir die Durchlaufzeit, die für das Erfüllen von internen oder Kundenanforderungen notwendig ist – insbesondere die Zeit zum Deployen von Code in die Produktivumgebung. Dadurch verbessern wir die Qualität der Arbeit, aber auch unseren Durchsatz, und verpassen unserer Fähigkeit, die Konkurrenz durch Innovation und Experimente hinter uns zu lassen, einen deutlichen Schub.

Die Praktiken, die sich daraus ergeben, sind Continuous Build, Integration, Test und Deployment, das Erstellen von Umgebungen auf Anforderung hin, das Beschränken der Work in Process (WIP) und das Aufbauen von Systemen und Organisationen, die sich sicher ändern lassen können.

Der Zweite Weg ermöglicht den schnellen und kontinuierlichen Feedback-Fluss von rechts nach links bei allen Schritten unserer Wertkette. Das Feedback dafür müssen wir einfordern, damit Probleme kein zweites Mal auftreten bzw. damit sie sich schneller erkennen und beheben lassen. Dadurch sorgen wir schon an der Quelle für Qualität. Wir schaffen oder hinterlassen dort Wissen, wo es gebraucht wird, sodass wir immer sicherere Arbeitssysteme aufbauen, in denen Probleme gefunden und behoben werden, lange bevor ein katastrophaler Fehler eintreten kann.

Indem wir Probleme erkennen, wenn sie auftreten, und uns so lange um sie kümmern, bis Gegenmaßnahmen aktiv sind, beschleunigen und verstärken wir fortlaufend unsere Feedback-Schleifen – ein wesentlicher Bestandteil so gut wie aller modernen Methoden zur Prozessverbesserung. Damit geben wir unserer Organisation die Gelegenheit, optimal zu lernen und besser zu werden.

Der Dritte Weg ermöglicht das Erstellen einer generativen, vertrauensvollen Kultur, die einen dynamischen, disziplinierten und wissenschaftlichen Ansatz zum Ex-

perimentieren und Eingehen von Risiken unterstützt und das firmenweite Lernen ermöglicht – sowohl durch Erfolge als auch durch Fehlschläge. Zudem sorgen wir durch das kontinuierliche Beschleunigen unserer Feedback-Schleifen für immer sicherere Arbeitssysteme und können eher Risiken und Experimente eingehen, um schneller als unsere Konkurrenz zu lernen und im Markt erfolgreich zu sein.

Als Teil des Dritten Wegs entwerfen wir unser Arbeitssystem so, dass wir die Effekte durch gewonnenes Wissen vervielfachen und lokale Entdeckungen in globale Verbesserungen umwandeln. Egal wo jemand Arbeit ausführt – er tut das mit der gesammelten und kollektiven Erfahrung aller in der Firma und deren Geschichte.

Continuous Learning: Durch Forschung untermauert: Die Drei Wege

Die Drei Wege sind nicht einfach nur eine gute Idee: Forschungen haben gezeigt, dass die Übernahme dieser Strategien zu besseren Ergebnissen sowohl für Organisationen wie auch für Personen führen.

In der sechsjährigen Studie, die von unserer Koautorin Dr. Nicole Forsgren in den *State of DevOps Reports* von 2014 bis 2019 vorgenommen (erst mit Puppet, dann mit DORA) und im Buch *Accelerate: The Science of Lean and DevOps* veröffentlicht wurde, zeigen die Daten, dass man bessere Ergebnisse erhält, wenn man Fähigkeiten und Praktiken wie Continuous Integration, Testen, Deployment und das Arbeiten in kleinen Einheiten (der Erste Weg), schnelles Feedback und Monitoring (der Zweite Weg) und eine generative Kultur (der Dritte Weg) einsetzt.[7]

Die Drei Wege unterstützen Teams dabei, zur Leistungselite zu werden, indem sie Software schneller und zuverlässiger ausliefern und dabei helfen, zu Umsatz, Marktanteil und Kundenzufriedenheit der Organisation beizutragen. Die Leistungselite wird die Performanceziele ihrer Organisation mit einer doppelt so hohen Wahrscheinlichkeit erreichen oder sogar übererfüllen. Die Drei Wege verbessern auch das Wohlbefinden derjenigen, die die Arbeit erledigen. Die Forschung aus den *State of DevOps Reports* zeigt, dass das Übernehmen dieser Praktiken zu weniger Burn-outs und Problemen beim Deployment führt.[8]

7 Forsgren, Humble, and Kim, *Accelerate* 2018

8 Leibman und Clanton, »DevOps: Approaching Cruising Altitude«