DevOps—Welcome to the Jungle

Georgia König · René Kugel

Eingegangen: 30. November 2018 / Angenommen: 5. Februar 2019 / Online publiziert: 4. März 2019
© The Author(s) 2019

Zusammenfassung Der vorliegende Artikel beschäftigt sich zentral mit dem Thema DevOps. Dafür werden die Ursprünge, Entstehung und Entwicklung der DevOps-Bewegung beleuchtet. Es wird diskutiert, warum der Begriff nicht konkret definiert werden kann und welche Auswirkungen dies mit sich bringt. Weiterhin werden ausgewählte Grundlagenprinzipien erläutert und verschiedene Strategien aufgezeigt mittels welchen DevOps in den ersten Schritten in IT-Unternehmen eingeführt werden kann. Zusätzlich wird das Konzept von NoOps eingeführt, welches als Weiterentwicklung von DevOps betrachtet werden kann. Auf der Basis dieser Konzepte wird argumentiert wie sich der Beruf des Developers, als auch der Operations-Fachkraft verändert und mit welchen neuen Herausforderungen diese konfrontiert werden.

Schlüsselwörter Softwareentwicklung · IT-Betrieb · DevOps · NoOps · Agile Methoden · digitale Transformation

DevOps—Welcome to the Jungle

Abstract This article talks about the topic of DevOps. The origins, emergence and further expansion over time of the DevOps movement will get highlighted. A discussion takes place, why the term cannot be specifically defined and which effects this entails. The underlying core principles are explained and a number of strategies are shown on how IT companies can implement initial steps for a DevOps adaptation. In addition, the concept of NoOps is presented which can be regarded as a further development of DevOps. On the basis of the two presented concepts it is

G. König (✉) · R. Kugel
Novatec Consulting GmbH, Dieselstraße 18, 70771 Leinfelden-Echterdingen, Deutschland
E-Mail: georgia.koenig@novatec-gmbh.de

R. Kugel
E-Mail: rene.kugel@novatec-gmbh.de

argued how the profession of developer, as well as operation experts might change and with which new challenges they will be confronted.

Keywords Software development · IT operations · DevOps · NoOps · Agile methods · Digital transformation

1 DevOps – Ursprung, Grundlagen und Definitionsproblematik

„But, NoOps means that application developers will never have to speak with an operations professional again.“ Mit diesem Satz löste Mike Gualtieri im Jahr 2011 in seinem Blogpost „I don’t want DevOps, I want NoOps“ (Gualtieri 2011) eine weitreichende Diskussion aus und brachte eine Bewegung ins Rollen, die in der IT-Branche bis heute kontrovers betrachtet wird. Acht Jahre später herrscht in vielen Unternehmen immer noch Verwirrung, was genau unter den Begriffen zu verstehen ist und wie damit assoziierte Konzepte effizient im Arbeitsalltag umgesetzt werden können. In diesem Artikel wollen wir betrachten, aus welcher Motivation heraus beide Begriffe entstanden sind, wie sie sich entwickelt haben und welche Auswirkungen diese mit sich bringen. Zudem wollen wir einen Einblick geben auf die Konflikte, denen sich der IT-Betrieb derzeit gegenüber gestellt sieht und einen Ausblick geben auf mögliche Umsetzungsstrategien der DevOps-Philosophie, sowie den daraus erwachsenden Konsequenzen für IT-Betriebsmitarbeiter.

Am Anfang war es ein Wasserfall. Dieses Modell der Softwareentwicklung findet auch heute noch Anwendung. Doch im Informationszeitalter und einer IT-Welt, in der Veränderung die einzige Konstante ist, ist Flexibilität ein essenzieller Faktor. Deshalb haben gerade in den letzten Jahren agile Software-Entwicklungsmethoden immer mehr Anklang und Bedeutung in der Wirtschaft gefunden. *Scrum* und *Kanban* sind die, im deutschsprachigen Raum, am weitesten verbreiteten Modelle, doch auch Konzepte und Methoden wie *Design Thinking*, *Lean* oder auch das schon ältere *eXtreme Programming* schreiben sich die Eigenschaft der Agilität auf die Fahne.

Der Einzug agiler Entwicklungsmethoden in zuvor „Wasserfall-artig“ gestalteten Arbeitsprozessen und Abteilungsstrukturen stellt Unternehmen vor deutliche Herausforderungen. Denn um agile Vorgehensmodelle im Unternehmen zu etablieren, genügt es eben nicht ein weiteres Tool einzuführen oder ein vordefiniertes Framework anzuwenden. Eine Änderung der kulturellen Denkweise in Kombination mit neuen Werten, Rollen, Strukturen und Tools ist hierfür grundlegend. Dabei sind die Vorteile dieser Entwicklungsmethoden nicht von der Hand zu weisen. Neben erheblich reduzierten Kosten, wird die Flexibilität, mit der ein Entwicklungsteam auf die Bedürfnisse des Business in einem dynamischen Markt reagieren kann, stark erhöht – dies führt letztendlich zu einer höheren Wahrscheinlichkeit des Projekterfolgs (Bradley 2015). Ein zentrales Ziel vieler agiler Methoden besteht darin, die Frequenz zu erhöhen, in der Features ausgerollt werden, die einen Mehrwert für das Unternehmen schaffen (sog. MVPs für *minimum viable products*).

Doch gerade diese erhöhte Frequenz mit der Software unter Verwendung von agilen Methoden ausgeliefert werden kann, verstärkt einen Konflikt, der in der Branche bereits seit langem besteht, jedoch bis dato zum größten Teil unter Kontrolle ge-

halten werden konnte. Der Konflikt zwischen Entwicklung (engl. „Development“ – kurz „Dev“) und Betrieb (engl. „Operations“ – kurz „Ops“). Der Konflikt, der inzwischen auch die *Wall of Confusion* genannt wird und ein zentraler Motivationsfaktor zur Entstehung der DevOps-Bewegung war. Um die Wall of Confusion zu verstehen, müssen wir die Motivationen und Ziele der beteiligten Parteien genauer betrachten. Auf der gedachten linken Seite der Wand befindet sich das Development, dessen grundlegendes Ziel es ist, neue Funktionalität zu implementieren. Nun erlaubt die Anwendung von agilen Methoden, dass diese Funktionalitäten in Form eines MVP immer schneller gebaut und geliefert werden können.

In einer perfekten Welt sind diese MVPs unter der Berücksichtigung von „Betriebs-relevanten“ Faktoren wie Sicherheit, Stabilität und Kompatibilität entwickelt worden, um ein Deployment in eine Produktivumgebung zu erleichtern. In der realen Welt lässt sich allerdings oft beobachten, dass genau diese Faktoren bei der Entwicklung vernachlässigt werden, da – erstens – diese Qualitätsmerkmale nicht im Verantwortungsbereich des, oft unter Zeitdruck stehenden, Development-Teams liegen und – zweitens – das Development zumeist anhand der Anzahl der gelieferten MVPs bewertet wird, solange eine grundlegende Funktionalität vorhanden ist. Kurz gesagt: Das Development will Features liefern, und zwar schnell und viele – *Das Development will Veränderung!*

Auf der gedachten rechten Seite der Wand steht das Operations-Team, dessen übergeordnetes Ziel darin besteht, die Software als Ganzes zu betreiben. Dabei soll das Produkt möglichst stabil und sicher laufen. Denn Sicherheits- oder Verfügbarkeitszwischenfälle kosten Geld und sollen vermieden werden. Jede Änderung, die in die Produktion ausgerollt werden soll, stellt somit eine potenzielle Gefahr und gleichzeitig einen entsprechenden Arbeitsaufwand dar. Kurz gesagt: Der Betrieb will die eigenen Systeme vor Ausfällen schützen – *Der Betrieb will Stabilität!*

Die Wall of Confusion repräsentiert also die Schnittstelle, an der die auszurollenden Software-Artefakte von der Entwicklung an den Betrieb übergeben werden. (An dieser Stelle hört man oft auch die Formulierung „... über den Zaun werfen“). Der beschriebene Konflikt kann definitiv als kritisch betrachtet werden, sowohl für das Betriebsklima zwischen den Teams als auch für die Qualität des Gesamtprodukts.

Die DevOps-Bewegung beschreibt Ansätze um die „Wand“ zwischen Development und Operations einzureißen und den gesamten Software-Lebenszyklus zu optimieren. Namensgeber für das Kofferwort sind die „DevOpsDays“, eine Konferenz die erstmals 2009 im belgischen Gent stattfand. Was damals noch als eine kleine Versammlung mit knapp 60 Teilnehmern begann, hat sich bis heute zu einer international relevanten Konferenz entwickelt, auf der die neusten Erkenntnisse von Experten diskutiert werden. Doch selbst jetzt ist es noch schwierig in Worte zu fassen, was genau unter dem Begriff „DevOps“ zu verstehen ist. Während IT-Operation-Frameworks oder agile Entwicklungsmethoden klare Entitäten, Rollen und Maßnahmen definieren, sucht man dies bei DevOps vergeblich. DevOps kann und will keine one-size-fits-all Umsetzung anbieten oder vorschreiben. Die Frage danach, wie ein jeweiliger Software-Lebenszyklus konkret aufgebaut und durchgeführt werden muss, um „DevOps zu sein“ wird bewusst nicht explizit beantwortet: So schreibt Jez Humble welcher als ein Vordenker der DevOps Bewegung gilt: *„DevOps, a movement of people who care about developing and operating reliable, secure, high performance*

systems at scale, has always – intentionally – lacked a definition or manifesto“. (Kim et al. 2017).

DevOps ist eben keine weitere Rolle, Team, Abteilung, Toolchain, Prüfsiegel oder Regulierung, die eingestellt, eingekauft oder erfüllt werden kann. (Humble 2012) Ähnlich wie beispielsweise die Umweltschutz- oder Gleichberechtigungsbewegung identifiziert DevOps in seinen Grundzügen lediglich einen Problemzustand und ein übergreifendes Ziel, welches als Ideal gesehen wird. Um diesem Ideal kontinuierlich näher zu kommen, mögen lediglich menschliche, technische und organisatorische Lösungsansätze propagiert werden, welche sich in einer gegebenen Situation bewährt haben. Dies ist besonders für eine wissenschaftliche Herangehensweise zu unkonkret: Es lässt sich eine Vielzahl an wissenschaftlichen Publikationen finden, welche sich explizit mit dem Finden einer angemessenen DevOps-Definition beschäftigen. Gleichgültig dessen, welche Arbeitsdefinition von DevOps verwendet wird, muss diese allen Beteiligten deutlich machen, wo die Grenzen abteilungstechnisch gesehen werden, d. h. ob eine jeweilige DevOps-Praktik tatsächlich nur die beiden namentlichen Abteilungen oder weitere einschließt. In der Literatur finden sich zudem etliche Zusatz-Bezeichnungen wie *ArchOps*, *BizDevOps*, *SecDevOps*, *NoOps*, *AgileOps* etc. Manche davon behaupten eine Erweiterung von DevOps zu sein, andere bezeichnen eine Spezialisierung während wieder andere sich als Alternative zu DevOps positionieren. Um die Verwirrung perfekt zu machen, widersprechen oder überlappen sich diese Ansätze teilweise gegenseitig. All diese Ableitung können unter StarDevOps kategorisiert werden. Dabei steht „Star“ für eine Wildcard (*). Generell kann jeder weitere Akteur, der an der IT-Wertschöpfungskette beteiligt ist, hier eingesetzt werden wie DEV, OPS, QA, IT Sicherheit, Produktmanagement, Personal-, Recht-, Finanzabteilung etc. Ein Begriff wie QaDevOps würde demnach die Kollaboration zwischen Dev- und Ops-Teams und vor allem der Qualitätssicherungsabteilung (engl. Quality Assurance) bezeichnen. DevOps und die entsprechenden Variationen sollten somit eher als ein ganzheitlicher Ansatz verstanden werden, der diverse Aspekte über die gesamte IT-Wertschöpfungskette fokussiert. Diese Aspekte sind sowohl technischer-, organisatorischer- aber besonders auch menschlicher Natur. Glücklicherweise konnte durch den Diskurs der letzten Jahre der Begriff immer mehr konkretisiert werden. Heute gelten die „drei Wege“ für DevOps als richtungsweisend. Diese wurden u. a. im Jahr 2016 im „DevOps Handbook“ publiziert und geben Kerngedanken von DevOps wieder (Kim et al. 2017).

Die drei Wege beschreiben Prinzipien, die als Eckpfeiler von DevOps betrachtet werden können. Sämtliche Maßnahmen zur Einführung und Verwendung von DevOps sollten unter Berücksichtigung dieser Eckpfeiler stattfinden. Die drei Wege sind dabei nicht als Phasen oder Stufen zu sehen. Gemäß der kontinuierlichen Natur von DevOps darf nicht von einem „Fertig“ ausgegangen werden. Oftmals dienen sie als verschiedene Sichten auf eine Situation oder einen Prozess, um Probleme auf verschiedenen Ebenen (1: Von der Entwicklung zum Kunden, 2: vom Kunden zurück in die Entwicklung, 3: kontinuierliche Verbesserung aller beinhalteten Prozesse) zu sehen und ganzheitliche Lösungen zu finden.

- Der erste Weg – Flow: Hierbei handelt es sich um eine Sammlung an Praktiken und Prinzipien die darauf abzielen den Fluss von links (Development) nach rechts

(Operations) zu verbessern. Mögliche Maßnahmen als Beispiel: Integration von Operations-Fachpersonal in die Development-Teams, Erstellen von „Definition-of-Done“ für die Development-Teams unter Einbeziehen des Operation-Teams.

- Der zweite Weg – Feedback: Der zweite Weg beschreibt Praktiken und Prinzipien, die sich darauf fokussieren aus dem Betrieb gewonnene Erkenntnisse in Form von Feedback an die Entwicklung zurückzugeben um somit den ersten Weg weiter zu verstärken. Mögliche Maßnahmen als Beispiel: A/B Testing, Verstärken der Review-Prozesse.
- Der dritte Weg – Continual Learning and Experimentation: Während die ersten zwei Wege vor allem mit technischen und organisatorischen Mittel unterstützt werden können, zielt der dritte Weg darauf ab kulturelle Änderungen zu etablieren. Im Fokus steht dabei das Schaffen einer kontinuierlichen Lernkultur die Fehler toleriert und zum Experimentieren motiviert. Dadurch können die Feedback-Zyklen aus den ersten zwei Wegen verkürzt werden, wodurch wiederum ein stabileres und sichereres Produkt geschaffen werden kann.

Aus diesen Wegen und den von uns gemachten Erfahrungen lassen sich die folgenden übergeordneten Ziele von DevOps wie folgt zusammenfassen:

- Überwinden des chronischen IT-Konflikts – Einreißen der Wall of Confusion
- Häufigere Software-Releases erleichtern und optimieren
- Kontinuierliche Verbesserung der Arbeitsbedingungen aller Akteure des gesamten Software-Lebenszyklus
- Erhöhen der Stabilität und der Sicherheit der Software
- Den Weg von der Idee bis zum Kunden „Streamline-förmig“ zu gestalten

In diesem Artikel wurde bewusst DevOps lediglich als Kollaboration zwischen einem wie auch immer geformten Entwicklungs- und IT-Betriebs-Team dargestellt, ohne damit sagen zu wollen, dass die Fülle weiterer beteiligter Abteilungen, Teams oder Themenfelder bei einer DevOps Transformation ausgeschlossen werden sollen.

2 Strategien zur Verzahnung von Betrieb und Entwicklung

Im Rahmen dieses Artikels sollen verschiedene Strategien nach (Kim et al. 2017) beleuchtet werden, welche von Unternehmen für die eigene Wertschöpfungskette abgewägt werden können. Diese wurden ausgesucht, da sie keine Technologie oder Entwicklungsmethodik voraussetzen, um die Verzahnung zwischen beiden namensgebenden Abteilungen zu unterstützen.

2.1 Bereitstellung von Shared Services

Dieses Konzept beinhaltet die Erstellung und Verwaltung von *Self-Service-Funktionen* durch den IT-Betrieb. Dabei handelt es sich um zentralisierte Plattformen und Tooling-Services, die jedes Entwicklungsteam nutzen kann, um produktiver zu werden. Die Funktion dieser bereitgestellten Services kann beispielsweise das Aufsetzen von produktionsähnlichen Environments, Deployment Pipelines, automatisierten

Testing-Werkzeugen oder Produktionstelemetrie-Dashboards sein. Dabei sind diese Plattformen und Dienste idealerweise automatisiert und auf Abruf verfügbar, d. h. ohne, dass ein Entwickler ein Ticket öffnen und ein IT-Betriebsmitarbeiter es manuell abarbeiten muss. Damit würde sichergestellt werden, dass der Betrieb an dieser Stelle nicht Gefahr läuft, als Flaschenhals zum Kunden zu agieren. Organisationen, die ihren Teams nur die Nutzung genehmigter Tools erlauben, können als ersten Schritt damit beginnen, diese Anforderung für einige ausgewählte Teams probeweise aufzuheben, um zu experimentieren und herausfinden zu können, welche Fertigkeiten diese Teams produktiver machen. Interne Shared Services-Teams sollten kontinuierlich nach intern genutzten Tools und Toolchains suchen, welche in der Organisation weit verbreitet sind. Für solche kann dann entschieden werden, ob es sinnvoll ist, diese zentral zu unterstützen und für alle Teams zugänglich zu machen.

Im Allgemeinen ist es viel wahrscheinlicher, dass es gelingt, etwas, das bereits funktioniert, mitzunehmen und seine Nutzung zu erweitern, als diese Fähigkeiten von Grund auf neu aufzubauen. In fast allen Fällen sollte nicht verlangt werden, dass interne Teams diese Plattformen und Dienste nutzen. Das Plattformteam und Produkte müssen ihre internen Kunden überzeugen und zufrieden stellen, eventuell sogar mit externen Anbietern konkurrieren. Durch die Schaffung dieses effektiven Marktplatzes der Möglichkeiten wird dazu beigetragen, dass die erarbeiteten Plattformen und Dienste die einfachste und attraktivste Wahl ist.

2.2 Integration von IT-Betriebsfachkräften in Service-Teams: Das Embedded Ops-Modell

Ein weiterer Ansatz um schneller, besseren Output zu generieren besteht darin Produkt-Teams autarker aufzustellen. Um sie in die Lage zu versetzen, sich selbst zu versorgen und Abhängigkeiten zu einem zentralisierten IT-Betrieb zu verringern werden in diesem Ansatz einzelne IT-Betriebsfachkräfte explizit in das Team integriert. Dies wird im Folgenden als *Embedded Ops-Modell* bezeichnet. Aufgrund der Einbettung werden Prioritäten fast ausschließlich von den Zielen der zugeordneten Produktteams bestimmt, im Gegensatz zum zentralisierten IT-Betrieb, der sich nach innen auf die Lösung eigener Probleme konzentrieren muss. Dadurch werden die IT-Betriebsfachkräfte enger mit ihren internen und externen Kunden verbunden. Bei erstmalig initiierten, großen Entwicklungsprojekten können zu diesem Zweck von Beginn an IT-Betriebsfachkräfte in die jeweiligen Teams eingebunden werden. Zu deren Aufgaben kann es gehören, bei der Entscheidung zu helfen, was gebaut werden soll und wie es gebaut werden soll. Zudem können sie mittels funktionsübergreifendem Wissen die Produktarchitektur beeinflussen, Entscheidungen über interne und externe Technologienutzung unterstützen, neue Funktionalitäten auf internen Plattformen schaffen und vielleicht sogar neue operative Kapazitäten generieren. Nach einem Produkt-Release in die Produktion können eingebettete Mitarbeiter bei auftretenden Produktionsverantwortungen des Development-Teams helfen. Sie nehmen dazu an allen Ritualen teil, wie z. B. Planungsmeetings, Daily-Stand-Ups und Feature Demonstrationen. Wenn der Bedarf an Operations-Wissen und Kompetenzen sinkt, können sie zu anderen Projekten übergehen, wobei sie dem allgemeinen Muster folgen, dass sich die Zusammensetzung innerhalb der Produktteams während eines

Lebenszyklus ändert. Das beschriebene Paradigma hat einen weiteren Vorteil: Die Kopplung bietet eine sehr effiziente Methode, um funktionsübergreifendes Wissen und Expertise mit einem Service-Team zu teilen. Zudem kann es auch den großen Vorteil haben, dass das „Betriebs-Wissen“ in automatisierten Code umgewandelt wird, der wesentlich zuverlässiger und umfassender wiederverwendet werden kann.

2.3 Verbindungen zwischen Betrieb und Entwicklung aufbauen: Das Ops-Liaison-Modell

Eine Alternative zu der vorher besprochenen Idee ist das sog. *Ops-Liaison-Modell*. Aus vielfältigen Gründen, wie z.B. Standortverteilung, Kosten oder Belegschaftsknappheit, können oftmals nicht in jedem Produktteam IT-Betriebsfachkräfte eingesetzt werden. Viele der gleichen Vorteile lassen sich dennoch nutzen, indem für jedes Produktteam eine Liaison festgelegt wird. Die mit dieser Rolle verbundenen Lerninhalte könnten je nach Produkt und Team die folgenden umfassen:

- Die jeweils neue Produktfunktionalität sowie der Grund für die Entwicklung.
- Funktionsweise in Bezug auf Bedienbarkeit, Skalierbarkeit und Beobachtbarkeit.
- Wie und welche Metriken überwacht und gesammelt werden, um den Fortschritt, Erfolg oder Misserfolg der Funktionalität sicherzustellen.
- Alle Abweichungen von früheren Architekturen und Mustern, sowie die Begründungen dafür.
- Der zusätzliche Bedarf an Infrastruktur und Auswirkungen der Nutzung auf die vorhandene Infrastrukturkapazität.

Darüber hinaus begleitet diese Verbindung, wie im Embedded Ops-Modell, das Team, bringt dessen Bedürfnisse in die Operations-Roadmap ein und führt alle erforderlichen Aufgaben aus. Im Idealfall sollen mittel des Ops-Liaison-Modells Ressourcen-, Priorisierungs- und Zeit-Konflikte frühzeitig identifiziert und verringert werden.

2.4 Integration des Betriebs in „Entwickler-Rituale“

Als Folgeschritt der Einbettung von IT-Betriebsmitarbeitern oder als Verbindungspersonen in Projektteams können diese ebenso in die dort praktizierten „Entwicklungsteam-Rituale“ eingebunden werden. Zwar liegt der Fokus des nächsten Abschnitts auf Standart Ritualen nach der agilen Methodik, namentlich Daily Stand-ups und Retrospektiven, soll jedoch ferner auch Planungsmeetings und Feature Demonstrationen einschließen. Es bleibt zu betonen, dass agile Praktiken für diesen Schritt keineswegs Voraussetzung sind. An IT-Betriebsmitarbeiter wird appelliert, die praktizierten Rituale des Projektteams zu finden, sich in diese zu integrieren und ihnen einen Mehrwert zu verleihen.

Wenn die IT-Betriebs-Mitarbeiter an den Retrospektiven der Projektteams teilnehmen, können sie auch von neuen Erkenntnissen profitieren. Darüber hinaus sollte der Betrieb, wenn es in diesem Intervall zu einem Einsatz oder einer Freigabe kommt, die Ergebnisse und die daraus resultierenden Erkenntnisse präsentieren und Feedback an das Produktteam weitergeben. Das Feedback aus dem operativen Be-

reich hilft Produktteams, die Auswirkungen von Entscheidungen, die sie treffen, besser zu verstehen. Die zusätzlichen Arbeiten, die bei der Retrospektive des Projektteams identifiziert werden, fallen in die breite Kategorie der Verbesserungsarbeit, wie z. B. Fehlerbehebung, Refactoring und Automatisierung der manuellen Arbeit. Produktmanager und Projektmanager können die Verbesserungen zugunsten von Kundenmerkmalen verschieben oder entpriorisieren. Hierbei wird gemäß der Ansicht gehandelt, dass die Verbesserung täglicher Arbeit wichtiger ist als die tägliche Arbeit selbst, und dass alle Teams über eigene Kapazitäten verfügen müssen. Maßnahmen wie 20 % aller Zyklen für Verbesserungen zu reservieren (bzw. einen Tag pro Woche oder eine Woche pro Monat) werden dazu deutlich eingefordert. Ohne diese Maßnahmen sieht (Kim et al. 2017) die Gefahr, dass die Produktivität des Teams unter dem Druck der eigenen technischen und prozessualen Schulden (engl. *technical debt*) zum Stillstand kommen wird.

2.5 Relevante Arbeit des IT-Betriebs für jede Anwendung sichtbar machen

Eine in Entwicklungsteams bereits weit verbreitete Praktik um die eigene Arbeit sichtbar zu machen sind Projekt- Boards oder Kanban-Boards. Diese sind ein ideales Werkzeug, um Transparenz zu schaffen, und Transparenz ist eine Schlüsselkomponente, um die Arbeit des IT-Betriebs richtig zu erkennen und in alle relevanten Wertströme zu integrieren. Leider zeigen diese nur selten die relevanten IT-Betriebsarbeitsvorgänge, die durchgeführt werden müssen, damit eine Anwendung erfolgreich in der Produktion laufen kann. Lapidar ausgedrückt wird dieser Teil der Arbeitskette meist vernachlässigt, da dies später „von der anderen Abteilung“, d. h. „hinter der Mauer“ durchgeführt wird. Unter anderem deshalb ist das Entwicklungsteam sich oft nicht der benötigten Betriebsarbeit bewusst, bis es zu einer dringenden Krise kommt, Deadlines gefährdet werden oder ein System-Ausfall auftritt. Ein Sichtbarmachen auf einem gemeinsamen Kanban-Board würde es ermöglichen, alle notwendigen Arbeiten, die erforderlich sind, um Code in die Produktion zu überführen, klarer zu sehen, sowie alle IT-Betriebsarbeiten zu verfolgen, die zur Unterstützung des Produkts erforderlich sind. Darüber hinaus würde dem Team ein Einblick dahingehend ermöglicht werden, wo die Arbeit blockiert ist und Bereiche hervorheben, in denen Verbesserungen benötigt werden.

3 NoOps

Einer der wohl meist diskutierten Ansätze von StarDevOps (siehe vorherige Definition) ist NoOps (für No Operations – dt.: kein Betrieb). Dieser Ansatz basiert auf DevOps und wird von Befürwortern als die nächste Evolutionsstufe (Starmer 2018) betrachtet. Das Ziel dabei ist – wie der Name offensichtlich verrät – den Betrieb von dem Software-Lebenszyklus vollkommen zu exkludieren. Betrachtet man den Ansatz von der anderen Seite, besteht der logische Umkehrschluss darin, dass das Development-Team für sämtliche Betriebstätigkeiten verantwortlich ist. Somit unterliegt der NoOps-Ansatz dem oft verwendeten Mantra „*You build it, you run it!*“. Doch wie ist ein solches Konzept umsetzbar, fehlt Software-Entwicklern doch meist

die Expertise um ihre Produkte in einer ausreichend hohen Qualität zu betreiben? Und was passiert eigentlich mit dem Operations- Fachpersonal?

Für eine erfolgreiche NoOps-Umsetzung ist die Anwendung der beschriebenen DevOps-Prinzipie eine wichtige Basis. Denn nur wenn das Entwicklungsteam die technischen Mittel und die organisatorischen Prozesse besitzt, um ein Feature reibungsfrei in eine Produktionsumgebung auszurollen, ist es auch in der Lage den Betrieb zu überwachen und bei Problemen entsprechende Gegenmaßnahmen zu ergreifen. Trotz dieser zusätzlichen Aufgaben und Verantwortungen, sollen die Entwicklungsteams sich weiterhin in erster Linie auf die Entwicklung neuer Features und Produkte fokussieren. Dies klingt zunächst nach einer Herkulesaufgabe – ist aber mit den richtigen Mitteln tatsächlich realisierbar, wie Technologievorreiter Netflix bereits im Jahr 2012 beweisen konnte (Cockroft 2012). Dreh- und Angelpunkt für eine solche Umsetzung ist Automatisierung! Durch den massiven Erfolg und die schnelle Weiterentwicklung von Cloud-Technologien der letzten Jahre und besonders durch, das Aufkommen von Plattform as a Service (z.B. AWS, MS Azure, Cloud Foundry, Kubernetes etc.) und Paradigmen wie Infrastructure as Code ist es technisch möglich mit einem einzelnen Knopfdruck Software zu bauen, auf verschiedenen Umgebungen zu testen und letztendlich direkt in Produktion auszurollen. Logging-, Monitoring-, und Alertingsysteme überwachen dabei die Umgebung. Entsprechend konfiguriert und idealerweise mit KI-Technologien kombiniert, informieren diese das Entwicklerteam über die Vorgänge und den Status der Umgebung. Viele der genannten PaaS-Plattformen bieten Selbstheilungsmechanismen, die Applikationen neu ausrollen, bzw. den letzten funktionierenden Stand wiederherstellen, sollte es zu Zwischenfällen kommen. Sollten diese Mechanismen für einen spezifischen Problemfall nicht anwendbar sein, können Entwickler immer noch manuell eingreifen und mit wenigen Handgriffen und in Sekundenschnelle Änderungen rückgängig machen.

In der Tat ersetzen die neuen Technologien viele der Aufgaben die bisher von Operations-Teams übernommen wurden. Es ergibt sich natürlich die Frage, ob diese Stellen dadurch ersatzlos wegrationalisiert werden können. Aus unserer Sicht ist das nicht der Fall. Allerdings werden sich durch die Umsetzung von NoOps die Aufgaben des IT-Betriebs stark verändern. Man muss bedenken, dass die beschriebenen Automatisierungsprozesse eine sehr hohe Komplexität aufweisen, dabei oft sehr spezifisch auf ein Projekt angepasst werden müssen und sich mit der Zeit weiterentwickeln. Für genau solche Aufgaben wird weiterhin die Operations-Expertise benötigt. Das heißt natürlich im Umkehrschluss, dass sich auch das Operations-Fachpersonal mit diesen Technologien auseinandersetzen und neue Kompetenzen entwickeln muss. Abschließend lässt sich sagen, dass die DevOps-Adaption (und damit auch die Adaptionrate für NoOps) weiterhin steigen wird und dabei den IT-Betrieb nicht unberührt lässt. DevOps und die damit assoziierten Werte, Praktiken und Technologien finden anhaltende Resonanz in verschiedensten IT-Unternehmen und -Wertschöpfungsketten. Die Wahrscheinlichkeit, dass eine IT-Fachkraft mit einer wie auch immer gearteten DevOps-Adaption zu tun haben wird, steigt mit jeder adaptierenden Firma. In der Stackoverflow Developer Survey von 2018 bezeichneten 10,4 % der rund 92.000 Befragten ihre Rolle als „DevOps Spezialist“ (Stackoverflow 2018) und bis 2020 werden 20 % der Gartner-Kunden DevOps zur Unterstützung tra-

ditioneller IT-Initiativen nutzen, gegenüber weniger als 5 % im Jahr 2016. (Gartner 2016).

3.1 Konsequenzen für den IT-Betrieb

In diesem Abschnitt werden mögliche Konsequenzen der vorgestellten DevOps-Strategien und zunehmender DevOps-Adaption im Allgemeinen auf den IT-Betrieb betrachtet. Der IT-Betrieb wurde zu oft zu spät in den Lebenszyklus einer Anwendung einbezogen, sowie nicht in Planung und Verhandlungen von Beginn an berücksichtigt. Daher fühlt es sich im Verlauf eines gegebenen Projektes an, als würde dieser auf die Bremse treten oder Initiativen aufhalten. Ein besseres und früheres Engagement mit dem IT-Betrieb hätte diese Situation vermeiden und zu einem besseren Übergang führen können. Die Notwendigkeit immer weiter verkürzter Zykluszeiten bedeutet, dass die Beziehung zwischen beiden Seiten kooperativer werden muss, um die geforderten Geschwindigkeits- und Qualitätsstandards zu erfüllen. Trends wie Cloud und Infrastruktur as Code rücken Bedenken des IT-Betriebs weiter nach links innerhalb des Delivery Lifecycles. Auf der anderen Seite müssen sich Entwickler mehr für ihren vollen Anwendungs-Stack und die Produktionsumgebung interessieren, da ihre Anwendungs-Stacks immer komplexer werden. All diese Trends verändern die Kultur der IT-Operations-Teams und die Art und Weise, wie sie mit der Entwicklung interagieren. (Kim et al. 2015).

Eine Frage, die sich IT-Betriebsfachkräfte stellen könnten, ist die der Vereinbarkeit von DevOps mit dem bewährten Referenzmodell zum IT-Service-Management ITIL®. Manche betrachten DevOps als die Gegenreaktion auf ITIL, welches mehrere Generationen von Operations-Praktikern weitgehend beeinflusst hat. Dies soll nicht heißen, dass DevOps-Praktiken nicht mit ITIL-Prozess kompatibel gemacht werden können. Jedoch ist zu beachten, dass für die mit DevOps verbundenen, kürzeren Vorlaufzeiten und höheren Bereitstellungsfrequenzen viele Bereiche der ITIL-Prozesse vollständig automatisiert werden müssen. Da DevOps eine schnelle Erkennung und Wiederherstellung bei Serviceausfällen anstrebt, bleiben die ITIL-Disziplinen *Service Design*, *Incident* und *Problem Management* weiterhin relevant. Somit sieht die verwendete Literatur nach wie vor große Bedeutung in den IT-Betriebsdisziplinen *Service Design* und *Service Operation* (beschrieben in der ITIL 2011). Dies gilt besonders für *Incident Management* und *Problem Management*, als Unterprozesse von *Service Operation*, die weiterhin bedeutungsvolle Aufgabenfelder darstellen. Diesen Aufgabenfeldern sollte innerhalb des Software-Lebenszyklus eine größere Bedeutung zugemessen werden, damit ein Umdenken erfolgen kann: Weg von reaktivem, panischem Feuerlöschen im Falle eines auftretenden Fehlers oder eines Ausfalls hin zu einer proaktiven Problemidentifizierung sowie umfassender Problem Reviews. Für beides finden sich Unterprozesse innerhalb des *Problem Managements* (namentlich *Major Problem Review* und *Proactive Problem Identification*) die mit den neu erwachsenden technologischen und organisatorischen Möglichkeiten umgesetzt werden können. (Kim et al. 2017).

Als Abschlussappell möchten wir das Folgende mitgeben: Zum einen schadet es nicht, sich bei Interesse an DevOps mit dessen Erfolgsgeschichten und auch den damit verbundenen finanziellen Vorteilen vertraut zu machen. Mittels diesen

lässt sich besser, vor allem gegenüber Entscheidungsträgern, für riskant erscheinende DevOps-Testläufe argumentieren. Dazu empfiehlt sich zum Beispiel der *State of DevOps Report 2016* der u. a. auf den Return of Investment einer DevOps Transformation eingeht (Puppet 2016). Zum anderen bieten die vorgestellten drei Wege einen guten Startpunkt, um in Form von drei simplen Fragestellungen im eigenen Arbeitsalltag Prozesse sichtbar zu machen, Probleme zu identifizieren und angemessene Lösungen zu finden. Auch bei der Werkzeugauswahl können die drei Wege genutzt werden, um zu differenzieren auf welcher Ebene das Problem (am signifikantesten) auftritt: Entsprechende Fragestellungen wären beispielsweise 1. „Wie soll ein Flow-Prozess geschaffen oder verbessert werden?“, 2. „Wie soll ein Feedback-Prozess geschaffen oder verbessert werden?“ und 3. „Wie sollen Prozesse für Lernen und/oder Experimentation geschaffen oder verbessert werden?“. Gefundene Lösungen, egal ob übernommen und erprobt oder sogar selbst entwickelt, können dann dem stetig wachsenden DevOps-Werkzeugkasten hinzugefügt werden, um den einem Betroffenen viel zu vertraut gewordenen Ursprungskonflikt weiter zu mildern. Die Fachkräfte innerhalb des IT-Betriebs, deren Fähigkeiten und Charaktere, müssen sich von einer tiefen Spezialisierung und einem isolierten Denken zu einem generalistischen Ansatz entwickeln. Nicht nur ihre lokalen Verantwortungen müssen weiter optimiert werden, sondern über den gesamten Software-Lebenszyklus hinweg zusammenarbeitet und Verbesserungen gefunden werden. Denn weder eine erstklassige Softwareentwicklung noch IT-Betrieb allein zählen hierbei – beides muss zusammenkommen. Die Menschen, die in den nächsten fünf Jahren in ihrer Softwareentwicklungs- oder IT-Betriebskarriere erfolgreich sein werden, sind jene „Silobrecher“, die das große Ganze optimieren. (Kim et al. 2015).

Open Access Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz (<http://creativecommons.org/licenses/by/4.0/deed.de>) veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden.

Literatur

- Bradley C (2015) Large scale agile and Scrum vs. Waterfall. <https://www.scrum.org/resources/blog/largescale-agile-and-scrum-vs-waterfall-agile-6x-more-successful-14-cost-and-10x>. Zugegriffen: 30. Nov. 2018
- Cockroft A (2012) Ops, Devops and PaaS (noops) at Netflix. <https://perfcap.blogspot.com/2012/03/opsdevops-and-noops-at-netflix.html>. Zugegriffen: 30. Nov. 2018
- Gartner (2016) Seven steps to kick-start your devops initiative. <https://www.gartner.com/binaries/content/assets/events/keywords/data-center/dc11a/seven-steps-to-kick-startyour-devops-initiative-infographic.pdf>. Zugegriffen: 30. Nov. 2018
- Gualtieri M (2011) I don't want Devops, I want noops. https://go.forrester.com/blogs/11-02-07-i_dont_want_devops_i_want_noops/. Zugegriffen: 30. Nov. 2018
- Humble J (2012) There's no such thing as a „Devops team“. <https://continuousdelivery.com/2012/10/theresno-such-thing-as-a-devops-team/>. Zugegriffen: 30. Nov. 2018
- Kim G, Wootton B, Thair S, Skelton M, Taylor K (2015) Agile operations and the three ways. <https://www.ca.com/content/dam/ca/us/files/ebook/agileoperations-and-the-three-ways.pdf>. Zugegriffen: 30. Nov. 2018
- Kim G, Humble J, Debois P, Willis J (2017) The Devops handbook. How to create world-class agility, reliability, & security in technology organizations. IT Revolution, Portland
- Puppet (2016) State of Devops report. <https://puppet.com/resources/whitepaper/2016-state-of-devops-report>. Zugegriffen: 30. Nov. 2018

- Stackoverflow (2018) Developer survey results. <https://insights.stackoverflow.com/survey/2018>. Zugegriffen: 30. Nov. 2018
- Starmer R (2018) Noops – is this the future of Devops? <https://kumul.us/noops-future-devops/>. Zugegriffen: 30. Nov. 2018