ITIL4 und DevOps – Gegensätze, Komplemente oder ein lösbarer Konflikt?

Carsten Dorrhauer

Fachbereich III

Hochschule Ludwigshafen
Ernst-Boehe-Straße 4
D-67059 Ludwigshafen
E-Mail: carsten.dorrhauer@hwg-lu.de

Haio Röckle

Fachbereich III

Hochschule Ludwigshafen
Ernst-Boehe-Straße 4
D-67059 Ludwigshafen
E-Mail: haio.roeckle@hwg-lu.de

ABSTRACT
Die Entwicklung der Softwaretechnik einerseits und die Verbreitung von ITSM-Prozessen andererseits erfordern immer dringlicher die Verzahnung von DevOps und ITIL. Unterdessen ist aber deren Kompatibilität noch nicht systematisch untersucht. Erschwert wird die Gestaltung von Prozessen, die den Anforderungen von ITIL und DevOps Genüge tragen, durch die Verbreitung heterogener Landschaften sowohl technischer als auch organisatorischer Art. Aus technischer Sicht werden Cloud-Dienste mit On-Premise-Diensten kombiniert, aus organisatorischer Sicht wird dem Kunden ein IT-Service angeboten, der von mehreren Dienstleistern jeweils in Teilen angeboten wird.
Hier soll der Frage nachgegangen werden, ob und unter welchen Bedingungen beide Konzepte vereinbar sind. Besonderes Augenmerk liegt dabei auf ITIL4, das seit 2020 vorliegt und das an einigen Stellen explizit auf DevOps Bezug nimmt.

SCHLÜSSELWÖRTER
IT-Service-Management, ITIL, Software Engineering, DevOps

Einleitung

Seit langem herrscht weitgehender branchenweiter Konsens über die Notwendigkeit professioneller Prozesse für das IT-Service-Management. Die mit großem Abstand weiteste Verbreitung haben die Prozesse und Praktiken der IT Infrastructure Library (ITIL) gefunden, die ihren Ursprung vor rund 30 Jahren in britischen Regierungsorganisationen haben. Neben vielen anderen Aspekten regeln sie die Bereitstellung von Soft- und Hardwarereleases sowie von Patches zur Fehlerbehebung, deren Versionierung, Vorkehrungen zu ihrer Wiederherstellung im Störungs- und Katastrophenfall, Abläufe des Roll-Outs und ähnliches.
Seit einigen Jahren nun gewinnt das Konzept DevOps eine inzwischen ähnlich weite Verbreitung in der Softwareindustrie. Es zielt auf die Integration von Entwicklungs- und Betriebsprozessen. Letztere sind aber in vielen Organisationen die Domäne von ITIL.
Damit drängt sich die Frage auf, inwiefern diese Ansätze sich ausschließen, sich ergänzen oder sich kombinieren lassen. Im Folgenden sollen deshalb die aktuellen Standardwerke zu beiden Konzepten miteinander in Beziehung gesetzt werden. Im ersten Falle ist das die quasi standardgebende Axelos, die die Rechte an ITIL hält. Im zweiten Fall sind es die Publikationen der meistbeachteten DevOps-Protagonisten, allen voran Patrick Debois und Gene Kim.

Methodik

Zunächst sollen in aller Kürze die beiden Konzepte ITIL und DevOps mit ihrem jeweiligen Fokus vorgestellt werden. Eine Harmonisierung der Prozesse des Software Engineering und des IT-Service-Management steht aktuell vor der zusätzlichen Herausforderung, auf heterogene und virtualisierte Umgebungen Rücksicht nehmen zu müssen, auf die deshalb in Folge eingegangen wird. Anschließend wird der Frage nachgegangen, welche ITSM-Prozesse dieser Harmonisierung bedürfen. Schließlich wird die Vereinbarkeit der beiden Konzepte im Allgemeinen untersucht, um dann darzustellen, wie die aktuellen ITIL4-Publikationen im Besonderen diese Thematik explizit und implizit aufgreifen.

IT-Service-Management mit ITIL

ITIL ist eine Sammlung von sogenannten Best Practices, die auf einen Leitfaden für das IT-Service-Management zurückgehen, der ab 1989 für die britische Central Computer and Telecommunications Agency (CCTA) entwickelt wurde. (Vgl. Böttcher 2008, S. 1). Inzwischen legt jährlich eine sechsstellige Zahl von Menschen die Prüfungen ab, um ITIL-Zertifikate zu erwerben. (Vgl. Corless 2017). Diese Verbreitung ist für Praktiken und Prozesse in der IT-Industrie einzigartig und nur mit der Marktdurchdringung führender Softwareprodukte zu vergleichen.

Software Engineering mit DevOps

2009 organisierte Patrick Debois die ersten DevOps-Days in Gent. Grundlegende Idee waren und sind *gemeinsame* Ziele, Abläufe und Werkzeuge (Goals, Processes, Tools) für Entwicklungsprojekte *und* Systembetrieb. Damit soll die klassische und bis heute häufige Dichotomie von IT-Organisationen abgemildert und am besten schließlich überwunden werden, weil sie häufig zu Konflikten, unübersichtlichen Abläufen und unklaren

Zuständigkeiten führt und letztlich die Produktivität der Gesamtorganisation behindert.
DevOps ist im Grunde unabhängig vom Vorgehensmodell, ergibt aber aus Sicht des Projektmanagements besonders zusammen mit agilen Methoden Sinn. Aus technischer Sicht wird DevOps i.d.R. mit Methoden der Continuous Integration kombiniert. Die Kombination geeigneter Werkzeuge ermöglicht das mindestens tägliche Zusammenführen der Entwicklungsstände aller beteiligten Entwickler.

Heterogene und virtualisierte Umgebungen
ITIL adressiert u.a. Betriebsprozesse. DevOps adressiert u.a. Deployment-Prozesse. Beide Aufgaben werden immer komplexer angesichts heterogener Systemlandschaften und angesichts der Verbreitung virtualisierter Lösungen, insbesondere in der Cloud.
So muss z.B. ein Incident-Management-Prozess nach ITIL zur Behebung von Betriebsstörungen durch multiple Teams mit unterschiedlichen Kompetenzfeldern für Störungsursachen auf unterschiedlichen Plattformen komplex gestaltet werden.
Ein automatisiertes Deployment muss z.B. neben der eigentlich auszuliefernden Software auch Konfigurationsdateien berücksichtigen. (Infrastructure as code). Beide Konzepte müssen auch auf diese Herausforderung gemeinsam reagieren, um erfolgreich zu sein. Die Konfiguration der VM, auf denen die mit DevOps auszuliefernde Anwendung läuft, insbesondere in der Cloud, unterliegt genau wie die Software selbst dem Change und Release Management nach ITIL.

Betroffene ITSM-Prozesse
ITIL beschreibt eine Vielzahl von Practices, die die Vorgehensweise eines IT-Service-Providers abdecken sollen. Viele davon haben keine oder keine wesentlichen Berührungspunkte zum Software Engineering und damit zu DevOps. Bei ITIL stehen der Servicelebenszyklus und die kontinuierliche Serviceverbesserung im Mittelpunkt, bei DevOps der Softwarelebenszyklus und die kontinuierliche Verbesserung von Software. Aus Sicht von ITIL ist eine Software ein Configuration Item (CI). Da viele Services primär mittels Software-CIs erstellt werden, gibt es in diesem Umfeld Überschneidungen. Zu den Practices, die davon betroffen sind, gehören sicherlich vor allem *Change Control*, *Release Management* und *Deployment Management*. Sie befassen sich mit der Kontrolle von Änderungen an der Infrastruktur, mit dem Management von Releases, die i.d.R. mehrere solcher Änderungen umfassen sowie mit den technischen Aspekten der Auslieferung von neuer oder geänderter Software, Hardware und Dokumenten in eine Produktiv- oder auch Testumgebung.
Weitere Practices sind eher mittelbar betroffen, so das *Availability Management* sowie das *Capacity and Performance Management.* Für beide ist Software eine von mehreren CI bei der Erstellung eines Service, die seine Verfügbarkeit und Kapazität mitbestimmen. In einer DevOps-Organisation benötigen beide Practices Zielvorgaben und Informationen aus DevOps-Teams.

Die operative Arbeit mit dem Benutzer am *Service Desk* beim *Incident Management* nach ITIL ist offensichtlich mit DevOps kompatibel. Sie verändert sich höchstens insofern, dass im Rahmen des 2nd- oder 3rd-Level-Supports Entwickler an der Bearbeitung von Störungsmeldungen beteiligt werden und damit Teil der Prozesskette werden - falls das nicht ohnehin längst der Fall sein sollte. Analog lässt sich für den Prozess *Problem Management* argumentieren, der nach Fehlerursachen sucht und weitere Störungen vermeiden soll.
Zu den wesentlichen Voraussetzungen einer erfolgreichen DevOps-Organisation gehört die Erfassung und Auswertung von Produktiv-Telemetriedaten, um „immer genug Telemetriedaten (zu) haben, ... dass sich ... Services im Produktivumfeld korrekt verhalten. Und wenn Probleme auftreten, ... schnell herausfinden (zu) können, was falsch läuft, und ausreichend Informationen haben...“ (Kim et al. 2017, S. 190) Die Herausforderung ist dabei weniger eine technische als eine organisatorische, namentlich die Überwindung von Daten-Silos und die Definition von Zuständigkeiten und Prozessen für die Auswertung der Logs und die Erzeugung von Metriken.
Exakt diese Voraussetzung von DevOps schafft ITIL mit dem V3-Prozess *Event Management*, der in der ITIL4-Practice *Monitoring and Event Management* aufgeht.

Kompatibilität zwischen DevOps und ITIL
Bereits in der Einleitung zu ihrem DevOps-Standardwerk zählen die wohl wichtigsten DevOps-Protagonisten als einen von ihnen kurz zu widerlegenden Mythos über DevOps den Satz auf: „DevOps ist nicht kompatibel zu ITIL.“ (Kim et al. 2017, S. XVI) Sie argumentieren noch auf Basis von ITIL V3, dass DevOps kompatibel zu ITIL-Prozessen gestaltet werden könne, aber einer weitgehenden Automatisierung der ITIL-Prozesse bedürfe (vgl. ebenda, S. XVII) Umgekehrt diene die mit DevOps einhergehende höhere Deploymentgeschwindigkeit und –frequenz der Aktualität der Konfigurationsdatenbank CMDB und der Softwarebibliothek DML nach ITIL. (vgl. ebenda)
Eine der wichtigsten und ersten Maßnahmen zur organisatorischen Implementierung von DevOps-Prinzipien ist die Definition von Zuständigkeiten, Rollen und Kommunikationsbeziehungen an der Schnittstelle zwischen Entwicklung und Betrieb. (vgl. Kim et al. 2017, S. 91ff.) Das kann im Einzelfall unterschiedlich aussehen. Denkbar sind

- der Einbezug von Operatoren in die Entwicklungsteams, z.B. als Teilnehmer des Daily Scrum
- Self-Service-Angebote
- sogenannte Ops-Liaisons, also eigene Rollen an der Schnittstelle zwischen Entwicklung und Betrieb (vgl. ebenda, S. 93)

Da ITIL zwar vielfältige Zuständigkeiten, Rollen und Kommunikationsbeziehungen vorschlägt, darüber hinausgehende zusätzliche Zuständigkeiten, Rollen und

Kommunikationsbeziehungen aber nur in seltenen Ausnahmen ausschließt, überrascht es nicht, dass dieser Schritt in Richtig DevOps in keiner Weise mit ITIL im Konflikt steht.

Im Gegenteil fordert ITIL genau den Übergang von Kooperation zu Kollaboration zwischen den Teams in einer IT-Organisation und meint damit die Zusammenarbeit an gemeinsam vereinbarten Zielen. (vgl. Axelos 2020, S. 22) Dies deckt sich mit der Forderung nach gemeinsamen Zielsetzungen von Entwicklung und Systembetrieb bei DevOps. (vgl. Kim et al. 2017, S. 73)

Es bleibt die Frage zu klären, ob DevOps womöglich mit operativen ITIL-Prozessen, allen voran Change Management (ab ITIL4: Change Control), in Konflikt steht. Change Control hält die Balance zwischen Flexibilität und Zuverlässigkeit. Dabei gilt aber die Prämisse: "This assessment… should not introduce an unnecessary delay." (Axelos 2019, S. 119)

Insbesondere das mit DevOps einhergehende automatisierte Deployment könnte schließlich von restriktiven Service-Management-Prozessen ausgebremst werden. Automatisiertes Deployment vieler kleinerer Änderungen stellt offensichtlich andere Anforderungen an die Operations-Prozesse als ein großer Releasewechsel. Vereinfacht oder erschwert also automatisiertes Deployment die SM-Prozesse? Vereinfachen oder erschweren SM-Prozesse automatisiertes Deployment? Die Antwort findet sich in der Change-Kategorisierung nach ITIL. Nicht alle Änderungen müssen vom zuständigen Gremium, dem Change Advisory Board (CAB), einzeln autorisiert werden. *Standard Changes* können vorab genehmigt werden; freilich müssen sie dennoch dokumentiert werden. (Vgl. Axelos 2019, S. 119) Das typische Beispiel dafür ist die Installation von Standardsoftware auf Clients, die bereits auf anderen Clients installiert ist. Die Worst-Case-Betrachtung liefert das Bild eines einzigen ausgefallenen Arbeitsplatzes – ein Risiko, das in Kauf genommen werden kann. Diese Kategorie kann natürlich auch auf serverseitige Changes angewandt werden. Entscheidend ist die Risikobewertung, die ihrerseits abhängt vom „Vertrauen in Kontrollelemente wie automatisierte Tests und das proaktive Produktivitäts-Monitoring." (Kim et al. 2017, S. 325) Wichtig für ein reibungsloses, weitgehend automatisiertes Deployment ist, dass das CAB vorab nachvollziehbare Kriterien für Standard-Changes definiert, auf die die Entwickler sich verlassen können. Verschmelzen Entwicklungs- und Betriebsorganisation, so wird das CAB, das über diese Vorab-Genehmigung entscheidet, u.a. mit Entwicklern besetzt sein. Und automatisiertes Deployment erfordert ohnehin penible Versionierung und Qualitätskontrolle. Wenn man so will, legen damit die Entwicklungsprojekte aus eigener Motivation eine Disziplin diesbezüglich an den Tag, die das Service Management schon lange einfordert. Kim et al kommentieren einen Fall aus der Praxis wie folgt: „Liegt darin nicht eine Ironie? Dev beschwert sich gern darüber, dass Ops sich scheut, den Code zu deployen. Aber in diesem Fall haben die Entwickler die Macht, ihren eigenen Code zu deployen, und trauen sich genauso wenig." (Vgl. Kim et al 2017, S 221) DevOps-Teams, die Verantwortung für ihre Deployments tragen, scheinen genau zu der Vorsicht zu tendieren, die seinerzeit im Operations erst zur Etablierung von formalen Change-Prozessen geführt hat.

Die Kategorisierung als *Standard Change* kann viele, aber nicht alle Deployments vereinfachen. Im Lebenszyklus jeder Software gibt es Updates, die mit einem gewissen Risiko behaftet sind und die konsequenterweise vom CAB zu genehmigen sind, ITIL bezeichnet sie als *Normal Changes*. Unterstellt man ein grundsätzlich vertrauensvolles Verhältnis zwischen Entwicklung und Betrieb im allgemeinen – ohne das DevOps sowieso zum Scheitern verurteilt wäre – und innerhalb des CAB im besonderen, so steht und fällt die Qualität und Geschwindigkeit seiner Entscheidungen mit der Verfügbarkeit ausreichender Informationen zu den Änderungsanforderungen (RfC). Kim et al sind dazu optimistisch: „Wir können ziemlich sicher das Erstellen vollständiger und genauer RfCs automatisieren und das Ticket mit den Details zur Änderung versehen." (Kim et al 2017, S. 328). Genau dies sieht ITIL ausdrücklich vor: „...organizations ... often automate most steps of the change control process." (Axelos 2019, S. 119)

Stehen also Peer-Reviews aus agilen Modellen im Widerspruch zum formalen Change Control durch ein dafür vorgesehenes Gremium wie das CAB? Nur, wenn sie es in allen Fällen ersetzen sollten und aus überschneidungsfreien Personengruppen bestünden, die gegensätzliche Interessen verfolgen würden. Dies ist aber nicht notwendigerweise der Fall.

Teams, die nach DevOps arbeiten, brauchen schnelle Deployments. ITIL kann diese ermöglichen, wenn die Beteiligten es wollen. Ein Change Manager, der einen RfC grundsätzlich ablehnt, weil eine Information fehlt, anstatt mit dem Entwicklungsteam Rückfrage zu halten, wird ein DevOps-Team ausbremsen. So kann der falsche Eindruck entstehen, ITSM-Prozesse seien ein Hindernis für DevOps.

Hat ein geplantes Deployment den Change-Control-Prozess durchlaufen, so können die Installationen stattfinden. ITIL fordert einen gesicherten Speicherort, der Kopien aller verwendeten Softwaresysteme in den verschiedenen Versionen enthält, die Definitive Media Library (DML). Bei Bedarf kann aus der DML eine bestimmte Software jederzeit zu einem gewünschten Stand wiederhergestellt werden, unabhängig von den beteiligten Personen. Realisiert wird die DML für Individualsoftware meist als privater Cloudspeicher, für Standardsoftware gegebenenfalls als physischer Ablageort optischer oder anderer Datenträger. Kann nun ein automatisiertes und häufiges Deployment mit der DML in Übereinstimmung gebracht werden? Offensichtlich ist das eher eine organisatorische als eine technische Herausforderung. Erstens muss in der DevOps-Toolchain ein schreibender Zugriff auf die DML integriert sein. Auf diese Weise lässt sich im Störungsfall, z.B. nach dem Austausch einer Festplatte oder eines Servers, zweifelsfrei exakt die Softwareversion wiederherstellen, die in der Produktivumgebung zuletzt betrie-

ben wurde – auch an Feiertagen, mitten in der Nacht oder lange nachdem die verantwortlichen Entwickler das Unternehmen verlassen haben. Alternativ ist auch ein Pull-Mechanismus denkbar, bei dem die DevOps-Tools nicht schreiben, sondern den Schreibvorgang anstoßen. Zweitens müssen die ITIL-Prozesse für Change Control und Deployment Management natürlich genau diesen Schreibzugriff gestatten. Auch hierbei steht und fällt der Erfolg mit dem Zusammenwirken der Zuständigen für ITSM und DevOps, der personellen Überschneidung der Teams sowie angemessen zugeschnittenen Zielvereinbarungen und Verantwortlichkeiten.

Lösungen bei ITIL4

ITIL versteht sich als Konzept für IT-Dienstleister (IT Service Provider, ITSP). Indem es die Perspektive von IT-Produkten auf IT-Dienstleistungen lenkt, fasst es die Entwicklung der ihnen zugrunde liegenden Produkte mit deren Inbetriebnahme, Betrieb und dem Benutzersupport zusammen. Zum Design des Service gehört mehr als das Design einer Anwendungssoftware. Es umfasst u.a. die Vorbereitung der SLA und der Lieferantenbeziehungen sowie die Planung der Kapazität, der Sicherheit und der Verfügbarkeit nicht nur der Hardware, sondern z.B. auch des Personals. (vgl. Axelos 2020, S. 70) Zur Inbetriebnahme des Service gehört u.a., die Rollen und Prozesse des Service Desk aufzusetzen, wobei dem Early-Life Support (ELS) besondere Bedeutung zukommt. (vgl. ebenda, S. 73f.) Die Trennung von Dev und Ops ist bereits dem Kerngedanken von ITSM fremd.

Die Protagonisten von ITIL (vgl. Axelos 2020, S. 9) und DevOps (vgl. Kim et al 2017, S. 83) sind sich einig, dass statisches technisches Expertentum, das traditionell weit in der IT verbreitet ist, den Anforderungen moderner IT-Organisationen nicht mehr genügt. Organisationen benötigen Mitarbeiter mit breiten und flexiblen Kompetenzprofilen. Was DevOps-Teams benötigen, sind Teammitglieder, die beide Felder, Dev und Ops, abdecken können. Dies deckt sich mit dem „pi-shaped individual“ nach ITIL. (vgl. Axelos 2020, S. 11) Dort wird auch beschrieben, wie Organisationen die Rahmenbedingungen setzen können, um die Entwicklung solcher Kompetenzprofile zu fördern. (vgl. ebenda). Ähnlich, aber ausführlicher und mit starkem Fokus auf das Lernen aus Fehlern, wird bei DevOps argumentiert. (Vgl. Kim et al 2017, S. 265ff.)

Für einen ITSP, der nach DevOps arbeitet, kann die Bildung marktorientierter Teams sinnvoll sein, die „funktionsübergreifend und unabhängig“ (Kim et al. 2017, S. 78) gestaltet werden. Sie ermöglichen flexible Kollaboration zwischen Entwicklung und Systembetrieb mit einem begrenzten Fokus. Die Herausforderung, die sich daraus ergibt, liegt in der Binnenorganisation dieser multifunktionalen Einheiten. ITIL4 schlägt als Lösung Swarming vor. Dabei bearbeiten Experten verschiedener Felder, hier also Dev und Ops, solange gemeinsam eine Aufgabe, bis sich herausstellt, wer am geeignetsten ist, sie weiterzuverfolgen. (Vgl. Axelos 2020, S. 94)

Wohl am offenkundigsten tritt der Einfluss von DevOps auf die ITIL4-Practices beim V3-Prozess *Release and Deployment Management* zutage. DevOps braucht schnelle und weitgehend automatisierte Lösungen dafür. Detallierter werden in ITIL4 Werkzeuge und Methoden beschrieben, mit deren Hilfe dieses Ziel erreicht werden kann. (Axelos 2020, S.34ff) Dass in ITIL4 die Practices *Release Management* und *Deployment Management* separat aufgeführt und beschrieben werden, ist sicherlich nicht nur dem Umstand geschuldet, dass ITIL4 endlich zwischen Service Management Practices und Technical Management Practices unterscheidet und beide entsprechend einsortiert. Vielmehr erleichtert diese Unterscheidung die Verwendung von ITIL in DevOps-Umgebungen, wenn zwischen einem Release einerseits sowie einem oder mehreren Deployments andererseits sauber unterschieden wird. Beide können nunmehr zeitlich und inhaltlich unabhängig voneinander betrachtet, erstellt, validiert und genehmigt werden. Ein Deployment muss nicht unmittelbar den Benutzern verfügbar gemacht werden, erst Recht nicht der Gesamtheit der Benutzer. Ein Vorgehen der Art Test - Pilotierung - allgemeine Verfügbarkeit ohne erneutes Deployment war in ITIL3 schwer abbildbar, ggf. hätten Dummy-Releases erstellt werden müssen, um dem Prozess Genüge zu tun. Um dies abbilden zu können, berücksichtigt die ITIL4-Practice *Release Management* nun insbesondere Blue/Green-Releases in zwei gespiegelten Produktivumgebungen (vgl. Axelos 2019, S. 135) sowie Feature Flags zur kontrollierten Freischaltung einzelner Features für einzelne Nutzer oder Nutzergruppen. (vgl. ebenda, S. 136) Völlig unproblematisch ist aus Sicht von DevOps die Zuständigkeit für Release-Entscheidungen bei ITIL: “The tools ... may be the responsibility of a dedicated person, but decisions about the release can be made by the development team.“ (Ebenda)

Deployment Management ist in ITIL4 zu einer eigenen Practice im neu geschaffenen Abschnitt *Technical Management Practices* befördert worden. Es erlaubt naheliegenderweise inkrementelle Deployments. (Vgl. Axelos 2019, S. 161) Es nimmt explizit Bezug auf Continuous Delivery, also das fortlaufende Integrieren, Testen und Ausliefern von einzelnen Komponenten, das im Grunde nur bei Verwendung agiler Methoden in der Softwareentwicklung sinnvoll ist. Für seinen Erfolg bedarf es der Entscheidung für den Ansatz *Direct Integration*, der ein Produktivdeployment bei Fertigstellung erlaubt, aber umfangreiche Testautomatisierung voraussetzt (vgl. Axelos 2020, S. 31) Nur der Vollständigkeit halber sei erwähnt, dass ITIL schon immer die Beteiligung der Entwickler am Deployment in Produktivumgebungen unterstützt hat. ITIL4 erwähnt nun auch ausdrücklich die Möglichkeit, dass es sich dabei um Externe oder um Serviceintegratoren handelt. (vgl. Axelos 2019, S 161)

Die ITIL4 Foundation führt sieben universelle Leitprinzipien ein, die eine ITIL-basierte IT-Organisation bestimmen:

- Konzentration auf Wertschöpfung,
- Start mit dem Ist-Zustand,
- iterativer Fortschritt mit Feedback,
- transparente Zusammenarbeit,
- Ganzheitlichkeit,
- Einfachheit sowie
- Optimierung und Automatisierung. (vgl. Axelos 2019, S 39)

Ein Jahr später wird ausführlich beschrieben, wie diese Prinzipien mit Continuous Delivery in Bezug gebracht werden können:

- Wertschöpfende Entwicklung früher zum Kunden bringen.
- Inkremente werden zum bereits bestehenden Ist-Zustand hinzugefügt.
- Kurze Feedback-Schleifen sind Kern agilen Arbeitens.
- Transparenz der Arbeitsergebnisse für alle Beteiligten, also Entwicklung, Betrieb und Kunden.
- Die Ziele des IT-Dienstleisters und des Kunden werden holistisch betrachtet.
- Auf unproduktive Aktivitäten wird nach Möglichkeit verzichtet.
- Auslieferung von Code und Feedback sollen automatisiert werden. (vgl. Axelos 2020, S.45)

Die offensichtliche Nähe der neuen Leitprinzipien zu Agilität und Continuous Delivery lässt vermuten, dass entgegen der Reihenfolge der Veröffentlichungen diese originär Leitbild bei der Entstehung von ITIL4 waren. Gleichwohl wird betont, dass diese Lösung nicht in jedem Fall anwendbar ist. Sie passt insbesondere dann nicht, wenn gegenwärtige und zukünftige Anforderungen besonders unklar sind. (vgl. Axelos 2020, S. 46)
Die Schnittstelle zwischen Entwicklungsprojekten einerseits sowie Systembetrieb und Benutzersupport andererseits birgt seit jeher großes Konflikt- und Fehlerpotential. DevOps ist angetreten, dieses abzumildern und schlägt dazu u.a. vor, die strenge organisatorische Trennung beider Zuständigkeiten aufzulösen. Schon in ITIL V3 finden sich Ansätze dazu insbesondere im Prozess *Application Management*: "Umfassendes Application Management hebt das Nebeneinander von Software-Entwicklung und Service Management auf, indem es die Software-Entwicklungsphasen und die ... Service Management-Phasen in einen einzigen Lebenszyklus integriert." (Ebel, S. 324)
Wie bereits ausgeführt, kann die ITIL4-Practice *Monitoring and Event Management* eine der wesentlichen Voraussetzungen für erfolgreiche DevOps-Organisationen bereitstellen: eine Telemetrie-Infrastruktur. Zunächst sind die zu monitorenden Services und Infrastrukturkomponenten zu identifizieren, dann sowohl deren native als auch die werkzeugbasierten Monitoringfunktionen einzurichten, Metriken und Schwellenwerte festzulegen, vorab die Reaktion auf bestimmte Events zu bestimmen und die zugehörigen Prozesse einzusetzen. (vgl. Axelos 2019, S. 118f.) Ausdrücklich erwähnt wird der Einbezug von Betriebs- *und* Applikationsverantwortlichen: „For the definition of monitoring strategies and specific thresholds and assessment criteria, it can help to bring in a broad range of perspectives, including infrastructure, applications,..." (Ebenda, S. 129) Technisch wird diese holistische Sicht auf Erfassung und Auswertung von Monitoring-Informationen ermöglicht, indem anfallende Daten zentral erfasst werden. Auf diesen Daten wird dann algorithmisch nach Mustern und Auffälligkeiten gesucht. (vgl. Axelos 2020, S. 40)

Fazit

Es wurde deutlich, dass alleine die Existenz funktionierender ITSM-Prozesse nach ITIL der Arbeitsweise nach DevOps nicht im Wege steht. Wenn dieser Eindruck entsteht, dürfte er auf eine Überformalisierung zurückzuführen sein, in der insbesondere die Prozesse für Änderungskontrolle und Konfigurationsmanagement zu bürokratisch mit Leben gefüllt werden. Beide bieten die Flexibilität, DevOps-kompatibel gestaltet zu werden. Ein Konfigurationsmanagement, das gewissenhaft die DML aktuell hält, ist umgekehrt sogar Voraussetzung für agiles Deployment, ebenso ein funktionierendes Event Management, das Monitoringdaten verarbeitet.
Sowohl ITIL als auch DevOps fordern gemeinsame Ziele für Entwicklung und Systembetrieb sowie die dazu passenden Qualifikationen der Beteiligten.
Diese Synopse der beiden Konzepte liefert als Erkenntnis, dass ITIL-Prozesse ohne DevOps-Prinzipien möglich sind, dass umgekehrt aber DevOps ohne ITIL oder vergleichbare ITSM-Prozesse kaum erfolgreich in größeren Organisationen sein wird. Wesentliche Bedingung für die Kompatibilität von ITIL und DevOps ist ein flexibler und unbürokratischer Umgang mit ITSM-Praktiken, insbesondere mit Change Control. Wünschenswert bleibt für die Zukunft eine empirische Untermauerung dieses Ergebnisses anhand von Beispielen aus der Praxis.

LITERATUR

Alt, Rainer; Auth, Gunnar; Kögler, Christoph. 2017. Innovationsorientiertes IT-Management mit DevOps, Wiesbaden: Springer

Andenmatten, Martin. 2014: Wie Cloud Services das Betriebsmodell in Unternehmen verändert". In: HMD Praxis der Wirtschaftsinformatik 51.6 (2014), S. 782–794

Axelos (Hrsg.) 2019. ITIL Foundation. ITIL4 Edition, Norwich: The Stationery Office, 2019

Axelos (Hrsg.) 2020. ITIL4: Create, Deliver and Support, Norwich: The Stationery Office, 2020

Betz, Charles. 2017. ITIL and DevOps: Differences & Frameworks Working Together – BMC Blogs. 2017. https://www.bmc.com/blogs/itil-and-devops-lets-not-paper-over-the-differences

Böttcher, Roland. 2008. IT-Servicemanagement mit ITIL V3, Hannover 2008

Corless, Barry. 2017. ITIL® Retains Rank Among Top 5 Most Popular and Top 20 Highest-Paying IT Certi-

fications, 2017. https://www.globalknowledge.com/us-en/resources/resource-library/articles/itil-retains-rank-among-top-5-most-popular-and-top-20-highest-paying-it-certifications/
Ebel, Nadin. 2008. Itil V3 Basis-Zertifizierung, München 2008
Kim, Gene; Debois, Patrick; Willis, John; Humble, Jez. 2017. Das DevOps Handbuch: Teams, Tools, und Infrastrukturen erfolgreich umgestalten. 1. Auflage. Heidelberg: dpunkt.verlag

KONTAKT

Prof. Dr. Carsten Dorrhauer
Hochschule Ludwigshafen
Ernst-Boehe-Straße 4
D-67059 Ludwigshafen
carsten.dorrhauer@hwg-lu.de
+49 621 5203 330