DevOps - Gemeinsam produktiv werden

DevOps ist eine von Entwicklern und Administratoren initiierte Bewegung, die sich zum Ziel gesetzt hat, die Zusammenarbeit zu verbessern. Die Bereiche Software-Entwicklung und IT-Betrieb in großen Unternehmen haben meist eine sehr verschiedene Vorstellung davon, wie selbst-entwickelte Anwendungen produktiv genommen und betreut werden. Diese unterschiedliche Zielsetzung führt jedoch an mehreren Stellen zu geschäftsrelevanten Verzögerungen, Behinderungen oder gar Ausfällen. DevOps adressiert genau dieses Problemfeld und bietet Lösungswege.

Von Udo Pracht, Dezember 2011

Was halten denn in Ihrem Unternehmen die Entwickler und die Administratoren so voneinander? Klappt die Zusammenarbeit problemlos? Oder aber: Wie lange benötigt Ihre IT, um eine minimale Änderung – zum Beispiel aufgrund einer korrigierten Code-Zeile – produktiv zu nehmen? Wie viele Support-Mitarbeiter betreuen wie viele produktive Programme? Und wie gut können sich diese Anwendungs-Betreuer untereinander vertreten? Wie häufig gibt es neue Releases und wie groß ist jedes Mal der Aufwand?
Die obigen Fragen beleuchten alle die Schnittstelle zwischen Software-Entwicklung und IT-Betrieb. Genau dort treffen nämlich zwei ganz unterschiedliche Interessen aufeinander:

- Die Aufgabe der Entwicklung ist es, zu verändern. Sie soll im Auftrag des Business, beziehungsweise der Fachbereiche, neue oder verbesserte Funktionen erstellen und nutzbar machen. Ein wichtiger Aspekt dabei ist, wie schnell sich etwas realisieren lässt – Time-to-Market! Bei einem agilen Vorgehen soll dies zudem in einer möglichst hohen Frequenz passieren („Release often!“).
- Der IT-Betrieb (englisch: Operations) wird an Verfügbarkeit der Systeme, Stabilität und Effizienz gemessen. Änderungen sind so etwas wie der natürliche Feind, bergen sie doch die Gefahr, dass etwas schief geht und dadurch die genannten Ziele gefährdet werden. "Never change a running System!" lautet die Maxime.

Entwicklung versus Betrieb

Wahrscheinlich ist es genau dieser Zielkonflikt, der dazu führt, dass sich Software-Entwickler und Administratoren oft schwer miteinander tun. Für manchen Programmierer ist der Job getan, wenn eine erstellte Anwendung auf dem eigenen Rechner beziehungsweise in der Entwicklungs-Umgebung läuft und die Übergabe einer neuen Version sieht dann wie ein "über die Mauer werfen" aus. Operations hingegen erstellt möglichst hohe bürokratische Hürden, verschanzt sich dahinter und wehrt sich damit gegen "gefährdende" Veränderungen. Schnelle Reaktionszeiten und Flexibilität ersticken in Formalien.
Am Release-Prozess ist je nach Unternehmen noch eine dritte, unabhängige Gruppe beteiligt, zuständig für Test und Qualitätssicherung. Auch diese bringt wiederum eigene Ziele mit, denn ihre Arbeit wird oft lediglich daran gemessen, wie viele Fehler sie in der zu untersuchenden Software findet. In dieser Konstellation aus Entwicklung, Test/QS und Betrieb nimmt im Extremfall keiner mehr ein gemeinsames Ziel wahr sondern versucht jeder "seinen Kasten sauber zu halten" und nicht verantwortlich zu sein.

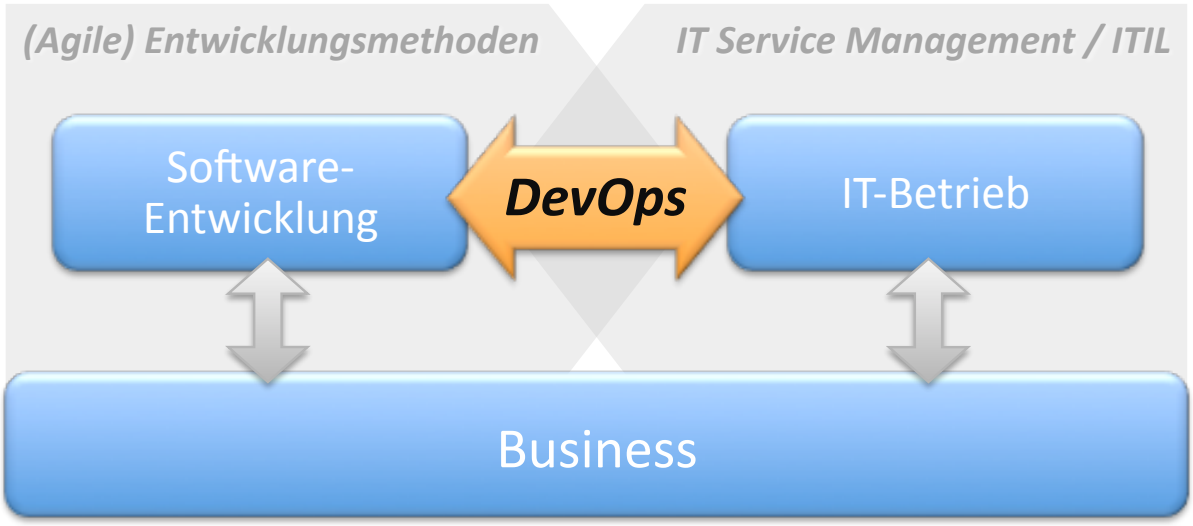


Abbildung 1: DevOps greift dort, wo sich Entwicklungs- und Betriebs-Methoden überschneiden

Symptome, die eine Schieflage anzeigen

Es gibt eine ganze Reihe von Anzeichen, dass es in der Zusammenarbeit von Software-Entwicklung und IT-Betreib nicht rund läuft und die letztendlich Nachteile für die Erfüllung der eigentlichen Geschäftsziele bedeuten:

- "Silo-isierung" bezeichnet eine Entwicklung, bei der stark fokussierte Experten in eigenständigen Organisationseinheiten zusammengefasst werden und nur noch für eine bestimmte Teilaufgabe verantwortlich sind, wobei ihnen jedoch der Gesamtüberblick verloren geht. Selbst innerhalb des Betriebes wird dieses Spezialistentum vorangetrieben und Anwendungsbetreuer sind nur noch für ein bestimmtes System zuständig – dafür aber unersetzlich. Wie weit die Silo-isierung fortgeschritten ist, lässt sich an einem üblichen Vorgang wie der Bereitstellung eines neuen Servers erkennen: Wie viele Teams oder Stellen müssen dafür eventuell scharf abgegrenzte Aufgaben (Beantragung, Genehmigung, Bestellung, Einbau, Verkabelung, Betriebssystem-Installation, Speicher-Anbindung, Software-Installation, Test und noch mehr) durchführen, wer behält dabei den Überblick und klappt das ganze aus Business-Sicht schnell genug?
- Seltene Release-Termine: Wenn im Unternehmen mit agilen Methoden entwickelt wird, es aber nur seltene und fest vorgegebene Release-Termine gibt, macht das viele agile Vorteile zunichte. Für Scrum und ähnliche Methoden ist es grundlegend, dass fertig gestellte Funktionen auch schnell produktiv genommen werden, denn nur so bekommt man wichtiges Benutzer-Feedback. Der Betrieb sieht Release-Termine jedoch als Ausnahme- oder gar Gefahren-Situation und versucht derartiges möglichst selten zu halten. Eine Reduzierung auf vier oder gar noch weniger Releases je Jahr ist in großen Unternehmen durchaus üblich. Je seltener jedoch Produktivnahmen durchgeführt werden, desto größer sind jeweils die Änderungen und damit die Fehlermöglichkeiten. Übung und Erfahrung bleiben gering und die Befürchtungen groß.
- Cloud-Computing und Virtualisierung bieten die Chance, Rechen- und Speicherkapazitäten dynamisch und bedarfsgenau zu nutzen und einzukaufen. Sind die für das Unternehmen geltenden Compliance-Anforderungen erfüllt beziehungsweise erfüllbar, müssen sich Entwicklung und Betrieb auf neue Technik, andere Schnittstellen und eine veränderte Administration einstellen und ihr Vorgehen unter anderem beim Release

anpassen. Nicht überall können diese Hürden erfolgreich genommen werden und es bleibt nur das diffuse Fazit: Cloud-Computing ist nichts für uns.

- Ein Kern-Element eines professionell geführten IT-Betriebes ist das Monitoring. Der Status der Systeme wird automatisch überwacht und ein Alarm ausgelöst, wenn definierte Fehler-Situationen eintreten. Wie oft kommt es dennoch vor, dass geschäftsrelevante Anwendungs-Probleme trotzdem erst von den Anwendern gemeldet werden? Häufig ist es doch so, dass zwar technische Kennzahlen und die grundsätzliche Verfügbarkeit von Anwendungen im Monitoring beachtet werden, nicht aber die Verfügbarkeit und Fehlerfreiheit einzelner Funktionen, die aber letztendlich erst den Nutzen des Systems für die Anwender darstellen.
- Wie ein Release-Prozess aussehen kann oder soll, wird von verschiedenen Konzepten beschrieben, sowohl in ITIL, in den Agilen Methoden als auch in den einschlägigen Projektmanagement-Lehren. Leider sind diese Beschreibungen und die zugehörigen Anforderungen nicht komplett deckungsgleich. Sofern eines der Vorgehen in einem Unternehmen eine stärkere Lobby hat und ohne Rücksicht auf ein Gesamtziel durchgesetzt wird oder aber erst gar kein klarer Weg gewählt wird, kann dies zu folgenden Symptomen führen: unstabile und pflege-intensive Anwendungen in der Produktion oder untragbar lange Zeiten, bis geschäftliche Anforderungen tatsächlich produktiv verfügbar sind.
- ITIL beschreibt recht detailliert und auch praxisnah, wie in der Produktion entdeckte Fehler effektiv und effizient bearbeitet werden können. Im Problem Management geht es darum, die Ursachen von Fehlerserien oder schwerwiegenden Fehlern zu ermitteln und zu beseitigen. Liegen die Ursachen im Umfeld von selbst entwickelten Anwendungen, muss der Betrieb zur Behebung mit den betreffenden Projekt-Teams zusammen arbeiten. Die Entwickler sind jedoch ohnehin schon voll ausgelastet und Fehlerbehebung ist meist nicht gerade deren Lieblingstätigkeit. Bevor also der Help-Desk oder der Second-Level-Support nicht ganz klar bewiesen hat, dass es sich um einen Programmfehler handelt, gibt es keine Hilfe. Durch das Hin- und Herschieben der Verantwortung und andere Verzögerungen in der Zusammenarbeit verlängert sich die Zeit bis zur Behebung der Fehlerursache, mit negativen Auswirkungen auf die eigentliche Geschäftstätigkeit.
- Wer bestimmte die Infrastruktur-Architektur, die Software-Entwicklung oder der IT-Betrieb? Beide beanspruchen dieses Vorrecht für sich, mit jeweils durchaus validen Begründungen (siehe Abbildung 2). Auch hier gilt: Wird dies einseitig, ohne Rücksicht und Beteiligung entschieden oder kein klarer Weg gewählt, entstehen aus gesamtgeschäftlicher Sicht Probleme: entweder organisch gewachsene, undurchschaubare und nur schwer wartbare Systemlandschaften oder aber unflexible, veraltete Umgebungen in die moderne, für das Geschäft benötigte Technik nur schwer Einzug findet.

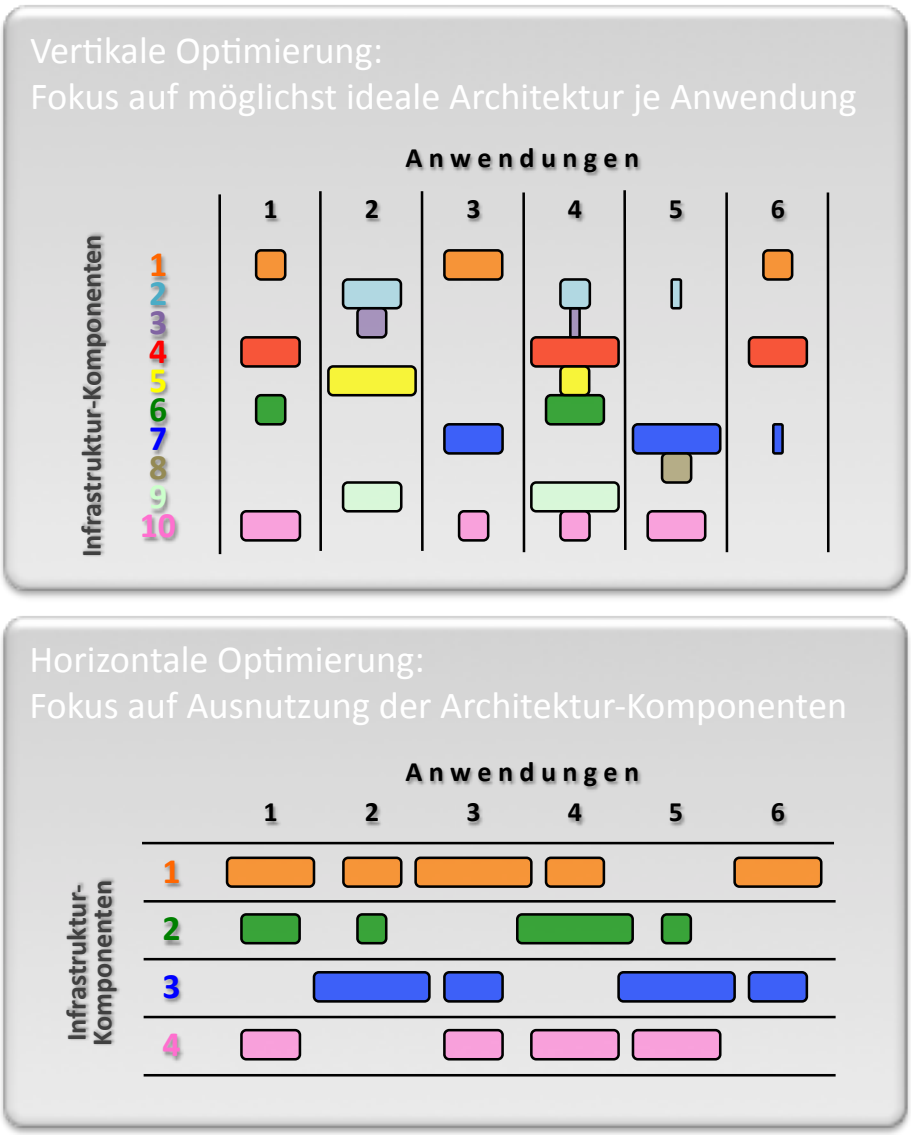
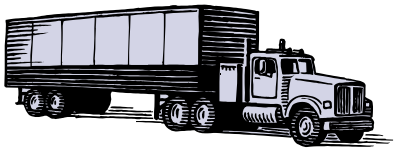


Abbildung 2: Die beste Architektur je Anwendung mit beliebig vielen Infrastruktur-Komponenten oder optimierte Ausnutzung von Ressourcen mit begrenzten Auswahlmöglichkeiten

- Ein weiterer Diskussionspunkt zwischen Entwicklung und Betrieb ist die Frage, welche Freiheiten Entwickler auf ihren Arbeitsgeräten oder auf den Entwicklungs-Systemen haben. Wenn eine Anwendung auf dem Entwickler-Rechner oder in der Entwicklungs-Umgebung fehlerlos läuft aber keiner mehr weiß, was dafür alles installiert und konfiguriert wurde, dann ist was schief gelaufen. Ein anderes extremes Symptom sind Entwickler, die ihre Zeit überwiegend damit verbringen müssen, Anträge zur Installation von Software zu stellen, die sie für ihre Arbeit benötigen oder die darauf warten, dass ihre beantragten Konfigurations-Änderungen auf dem Entwicklungs-Server durchgeführt werden.
- Als letztes seien noch zwei Aspekte genannt, die sich durch einfache – allerdings recht provokative - Fragen (siehe Abbildung 3) beleuchten lassen. Auch wenn die eine der beiden auf den ersten Blick makaber erscheinen mag, lenken beide doch den Blick auf etwas wesentliches, nämlich die Verwundbarkeit.



Truck Faktor

Wie viele Teammitglieder könnten von einem Truck überfahren werden, bevor das Projekt beziehungsweise der Betrieb ernsthafte Schwierigkeiten hat?

10ter-Stock-Test

Können Sie eine beliebige Komponente aus der Infrastruktur heraus nehmen, sie aus dem 10ten Stock werfen und die Infrastruktur innerhalb von maximal 10 Minuten wieder herstellen?

Abbildung 3: Fiktive Fragen als Anregungen zur Reflektion

Eine kurze Historie

Die gerade genannten Symptome und die Erfahrungen in derartigen Umgebungen waren es, die einigen Systemadministratoren, Entwicklern und Projektleitern den Antrieb gaben, daran etwas verbessern zu wollen. Die Ursprünge der Bewegung reichen unter anderem zurück bis ins Jahr 2005 in das Consulting-Unternehmen ThoughtWorks. Dort hat man, mit viel Wissen und Erfahrung aus der agilen Entwicklung, ein Projekt mit einer neuartigen Zusammenarbeit zwischen Entwicklung und Betrieb durchgeführt. "Continuous Delivery" bezeichneten Dan North, Jez Humble, Chris Read, David Farley und Julian Simpson das Vorgehen, welches mittlerweile auch in einem gleichnamigen Buch beschrieben ist.
Insgesamt ist der Hintergrund der Bewegung von den Erkenntnissen des agilen Vorgehens geprägt. Das, was die Software-Entwicklung dadurch in der Zusammenarbeit mit den Kunden beziehungsweise den Fachbereichen erreicht und verbessert hat, will man nun in den Betriebsbereich bringen. Unter Schlagwörtern wie "Agile Infrastructure", "Agile Sysadmin", "Agile Service Management", "Dev2Ops" und anderen wurde und wird das Thema in Blogs, Büchern und auf Konferenzen behandelt und abgesteckt. Es fließen Ansätze und Ideen aus Lean Thinking und Kanban mit ein. Ende 2009 fand in Belgien die von Patrick Debois organisierte, erste von mittlerweile zehn Konferenzen unter dem Titel "DevOpsDays" statt. Seit dem hat sich "DevOps", eine Abkürzung aus Development und Operations, als vereinigender, fokussierender Begriff für das Thema gefestigt.
DevOps ist jedoch noch in der Findung. Es wurden Probleme erkannt und benannt und es gibt diverse Lösungsansätze. Eine das Thema komplett umfassende Veröffentlichung steht noch aus. Beachtenswert ist jedoch, dass es ein Ansatz ist, der von Betroffenen und aus der Praxis kommt. DevOps ist kein abstraktes, theorieüberladenes Framework um (wieder mal) alles abzudecken. Es ist auch keine neue Anforderung, die es aus Compliance-Gründen oder um irgendwo dabei sein zu dürfen zu erfüllen gilt.

Was bringt DevOps dem Business

Schaut man sich die Abbildung 1 oben an, kann man den Eindruck bekommen, dass DevOps ein reines IT-Thema ist. Schließlich soll ja "nur" die Zusammenarbeit zwischen zwei Bereichen innerhalb der IT verbessert werden. Betrachtet man jedoch, wie eine Anwendung von der Idee zur Nutzung kommt, dann ist erkennbar, dass DevOps einen wesentlichen Schritt auf genau diesem Weg adressiert: den Übergang von der Realisierung zur Bereitstellung (siehe Abbildung 4). Denn ein Mehrwert für das Unternehmen entsteht ja

nicht, wenn die Entwicklung die Anwendung zur Zufriedenheit der Auftraggeber erstellt hat sondern erst dann, wenn diese Umsetzung auch produktiv verfügbar ist.

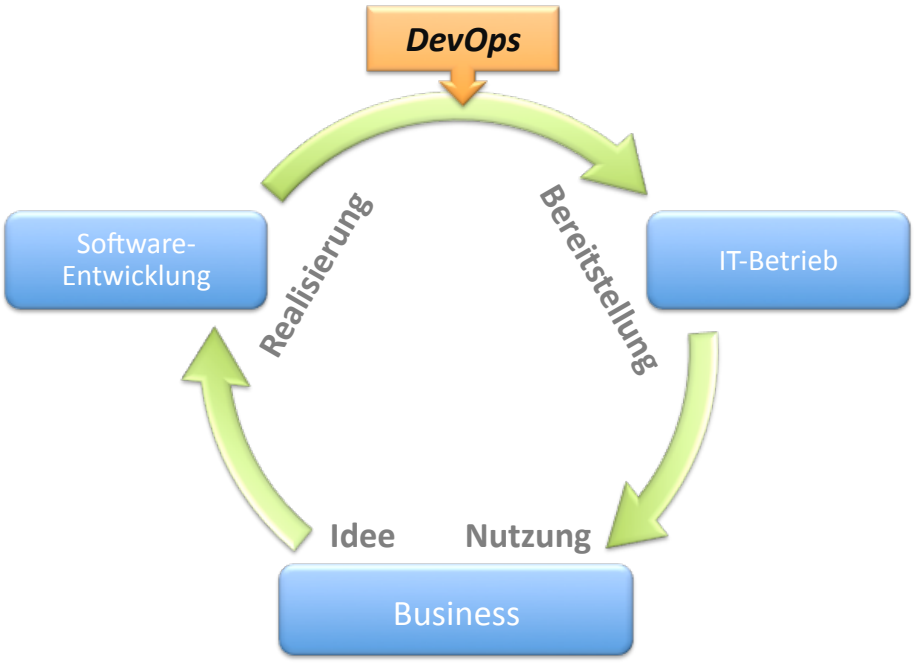


Abbildung 4: Software-Lifecycle und dessen Unterstützung durch DevOps

Dem Business verspricht DevOps folgendes:

- Time-to-Market reduzieren
- Aufwand und Kosten für Produktivnahme, Betrieb und Betreuung reduzieren
- Proaktiv vor Betriebs-Störungen schützen und diese schnell beheben können
- Virtualisierung und Cloud-Computing effektiv nutzen können

Aber, so der direkte Einwand, für all das gibt es doch schon längst genügend Konzepte und Methoden, die das lösen! Agiles Vorgehen hat die Entwicklungszeiten extrem reduziert, die Betriebs-Themen sind hinreichend durch IT Service Management Konzepte abgedeckt und die Nutzung von Cloud-Computing ist kein echtes und erst recht kein eigenständiges Problem. Die stärkste Kritik an DevOps lautet daher auch: Das ist doch von Scrum, ITIL, PMBOK oder irgend einem anderen Konzept schon längst adressiert, längst gelöst und muss nur entsprechend und konsequent umgesetzt werden.

Was aber, wenn trotz Agiler Methoden und diverser ITIL-Prozesse die weiter oben geschilderten Symptome immer noch auftreten? Vielleicht haben diese ihre Ursache genau in der gleichzeitigen Anwendung der verschiedenen Konzepte, weil sie eben nicht ganz kompatibel sind oder eben doch nicht alle Aspekte berücksichtigen. Die Symptome treten nämlich auch in Umgebungen auf, die sich agil, ITIL-konform oder beides nennen.

Lösungsansätze ganz konkret

DevOps will die Ursachen der oben genannten Symptome angehen und dadurch den versprochenen Businessnutzen generieren. Die konkreten Maßnahmen lassen sich unterteilen in **kulturelle** und **technische** – die jedoch wiederum ineinandergreifen oder aufeinander aufbauen. Der Schwerpunkt ist, die Haltung und den Umgang der Beteiligten miteinander zu ändern – die Technik ist eher ein Vehikel dafür.

DevOps-Ansätze zu Einstellungen, Kultur & Prozessen

Wie im bekannten Harvard-Verhandlungskonzept geht es auch hier darum, dass alle die eigenen Interessen und die der "Gegenseite" kennenlernen und ihre Berechtigung akzeptieren. Die Entwickler lernen, **warum** manche restriktiven Anforderungen den Administratoren wichtig sind und Produktions-Fehler schnell behoben werden müssen. Administratoren erkennen hingegen, **warum** die Entwickler Releases schnell produktiv nehmen möchten, bestimmte Zugriffs- oder Konfigurations-Rechte brauchen oder eine bestimmte Architektur präferieren. Das bedeutet nicht, dass man Forderungen widerstandslos erfüllt sondern im ersten Schritt nur: gegenseitiges Verständnis. Dies soll die Grundlage bilden für eine Veränderung der Zusammenarbeit: Weg vom "Fingerzeigen", hin zum "An-einem-Strang-ziehen". DevOps beschreibt eine Kultur beziehungsweise die Veränderung zu dieser.

Gegenseitige Ziele kennen und achten, Haltung ändern

Stephen Nelson-Smith hat einen aus der agilen Entwicklung bekannten Begriff genommen, ihn ein wenig abgewandelt und mit Elementen aus David Allens Natural Planning kombiniert. Mit "Collective Ownership" möchte er erreichen, dass zusammen arbeitende Entwickler und Administratoren jeweils das ganze Thema kennen und sich dafür verantwortlich sehen (siehe Abbildung 5).

Stephen Nelson-Smith empfiehlt fünf Schritte für die Zusammenarbeit:

- **Zweck und Grundsätze festlegen** – In einem Workshop die Grundlagen, Prinzipien und Erfolgsindikatoren für die gemeinsame Arbeit finden sowie die Frage nach dem Warum klären, bis hinunter zu den letztendlich darunter liegenden geschäftlichen Anforderungen.
- **Überragendes Ergebnis ausdenken** – Antworten finden zu den Fragen: Wie würde ein außergewöhnlich großer Erfolg der Zusammenarbeit aussehen? Was wäre anders und wie?
- **Brainstorming des Lösungsweges** – Brainstorming zum Thema: Wie kommen wir zu dem gerade skizzierten Ergebnis unter Berücksichtigung der festgelegten Grundsätze?
- **Organisieren** – Antworten aus dem Brainstorming thematisch sortieren (clustern), zum Beispiel nach: Technologie, Ideen, mögliche Probleme, zu klärendes, benötigte Ressourcen, hinzuzuziehende Personen und so weiter.
- **Nächste Schritte bestimmen** – Eine Liste mit Aufgaben erstellen und für die als nächstes anzugehenden, Zuständigen und Termin festlegen. Diese Schritte sollen dann – wann immer möglich – jeweils ein Entwickler und ein Administrator gemeinsam durchführen, ähnlich dem Pair Programming.

Abbildung 5: Collective Ownership für DevOps

Ein zentrales Anliegen – aus Business-Sicht und passend zur agilen Software-Entwicklung – sind häufige Releases. Wie bereits oben dargelegt, wird auf ein derartiges Ansinnen von Operations meist mit Vorbehalten und Befürchtungen ablehnend reagiert. Dies ist ein zentraler Punkt zur Frage des gegenseitigen Verständnisses. Der Entwicklung rät DevOps, die Sorgen ernst zu nehmen und zusammen mit dem Betrieb geeignete Mitigation-Maßnahmen sowie Vorteile einer höheren Release-Frequenz auch für den Betrieb zu erarbeiten (siehe Abbildung 6). Als Argumentationshilfe weißt DevOps darauf hin, dass bei häufigeren kleineren Releases...

- das Risiko und der Aufwand eines einzelnen Releases sinkt.
- (üblicherweise) die Differenz zum Fallback-Szenario kleiner wird.
- der Betrieb mehr Übung im Umgang mit Releases bekommt.
- Config-Switches eine gut praktikable Lösung für das Problem bei zeitlich versetzten Releases zwischen verbundenen Systemen darstellen.

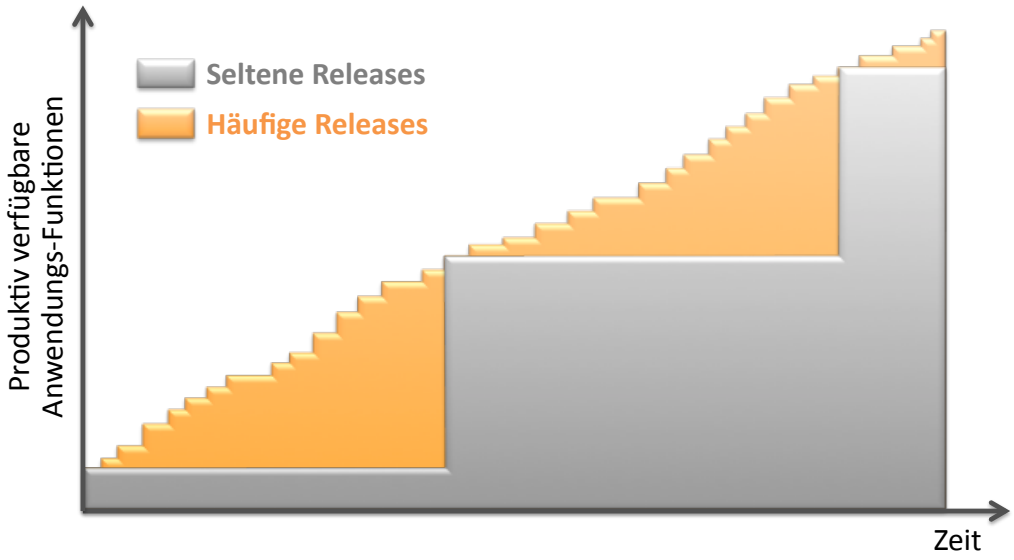


Abbildung 6: Release-Frequenz und der Umfang der geschäftlich nutzbaren Anwendung

Ein anderes Beispiel für – auf den ersten Blick – unterschiedliche Zielsetzungen ist das Monitoring. Wie schon oben unter den Symptomen erwähnt, gibt es meist ganz verschiedene Vorstellungen darüber was und wie gemessen wird, über sinnvolle Schwellwerte, die Meldungswege und die richtigen Alarm-Reaktionen. Mit einer gemeinsamen Sicht – auch zusammen mit dem Business – und einer betriebs-kompatiblen Implementierung lassen sich durch das Monitoring von wirklich business-relevanten Informationen Vorteile für alle erzielen (siehe dazu auch weiter unten unter "Geschäftsrelevantes Monitoring").
Letztendlich ist aber die Frage, welches die unterschiedlichen Ziele sind, situations- und unternehmensabhängig. DevOps will den Blick auf dieses Thema lenken, denn das Gefühl und die Überzeugung, dass alle "an einem Strang ziehen" motiviert deutlich mehr als die jeweiligen Forderungen in immer restriktivere Regelungen zu pressen oder durch immer findigere Lücken zu umgehen.

Zusammenarbeit und Kommunikation

Wie lässt sich die Kommunikation verbessern? Das ist schon oft beantwortet und DevOps erfindet nicht das Rad neu. In diversen Artikeln und Blog-Einträgen wird vor allem auf folgende Aspekte und Ansätze hingewiesen:

- Regelmäßige persönliche Treffen, anfangs gegebenenfalls häufiger und dann seltener werdend
- das bereits oben genannte "Collective Ownership"
- Gemeinsamer Einsatz von Kanban
- Aktiver, strukturierter Wissens-Austausch
- gemeinsames Lösen von Problemen und Klären von Fragen (zum Beispiel zur Infrastruktur-Architektur)

- Retrospektiven durchführen

Die DevOps-Unterstützer passen erprobte Methoden zur Kommunikations-Verbesserung aus anderen Bereichen an, setzten sie ein und berichten darüber. Daraus formt sich momentan langsam ein guter Grundstock an verwendbaren, konkreten Ansätzen.

Betrachtung und gegebenenfalls Änderung von Prozessen

In Unternehmen in denen die "Silo-isierung" durch Prozesse vorgeschrieben ist, kann DevOps auf Dauer sicher nur Erfolg haben, wenn diese Prozesse den Ansatz erlauben und aktiv fördern. Das gemeinsame Ziel ist also die Anpassung dieser Prozesse.
Ist es beispielsweise im Change Management des Betriebes vorgeschrieben, dass jedes Release vom einmal wöchentlich tagenden Change Advisory Board (CAB) abgenickt werden muss, kann es schwerlich tägliche Produktivnahmen geben. Hier gilt es kreativ zu überlegen, wie eine derartig hohe Frequenz mit der berechtigten Anforderungen nach Qualität und Stabilität vereinbar ist. Es könnte zum Beispiel ein technisch unterstütztes Verfahren (siehe weiter unten "Infrastruktur wie Code behandeln") die Qualität von Releases und die Stabilität der Produktionsumgebung sicherstellen und dann dieses Verfahren vom CAB einmalig bestätigt werden.
Welche Betriebs-Prozesse von der Bekenntnis zum DevOps-Gedanken berührt werden, ist sicherlich von Unternehmen zu Unternehmen unterschiedlich. Betrachten sollte man zumindest die kompletten Service Support Prozesse nach ITIL v2 beziehungsweise Service Operation und Teile aus Service Transition nach ITIL v3.

DevOps-Ansätze zu Technik, Werkzeugen & Automatisierung

Das Umfeld, in dem DevOps zum Tragen kommen soll ist geprägt von Technik und bevölkert von Computer-Experten. Natürlich ist es ein naheliegender Schritt zu schauen, wie die beschriebene Änderung in der Zusammenarbeit auch durch technische Lösungen unterstützt werden kann. Das Haupt-Anliegen der Bewegung ist ein kulturelles, es drängen sich jedoch häufig Tools in den Vordergrund. Neben der Technik-Affinität der Protagonisten sind ein weiterer Grund dafür auch Hersteller, die hier einen neuen Markt erkennen und das Thema als Zugpferd für ihre Software-Angebote nutzen oder gar missbrauchen.
Wichtig für die Werkzeuge und deren Einsatz ist: Sie sollen gemeinsam, von den Administratoren und den Entwicklern, genutzt und bedient werden. Denn diese Tools sind kein Selbstzweck sondern sollen die DevOps-Idee und deren Umsetzung unterstützen.

Infrastruktur wie Code behandeln

Ein zentrales Konzept der Bewegung wird mit "Treat Infrastructure as Code" umschrieben. Das theoretische Maximal-Ziel ist es dabei, in der Lage zu sein, eine Infrastruktur nur aus entsprechendem Blech und Kabeln sowie einem Repository (wieder) her zu stellen. Das erfordert, dass das komplette Deployment und die Konfiguration der Systeme (von den Switches angefangen bis hin zum Netzwerk-Speicher) automatisiert ablaufen kann. Alle Schritte in diesem Vorgang müssen dazu in lauffähigen Skripten beschrieben und von entsprechenden Programmen ausgeführt werden.
Ein Schritt auf dem Weg zu diesem Maximal-Ziel ist "Continuous Build and Deployment", also eine idealerweise automatisierte Installation neuer beziehungsweise geänderter Software. Anders ausgedrückt: Alle Veränderungen während eines Releases laufen

skriptgesteuert ab. Ein solches Verfahren ist die Grundlage, um eine hohe Release-Frequenz verwirklichen zu können.
Automatisierte Installation von Systemen, gepaart mit der Unterstützung von Virtualisierungs-Techniken bilden zudem die Voraussetzung, um die Möglichkeiten und Chancen von Cloud-Computing voll nutzen zu können. Ein großer Teil des Engagements für DevOps kommt aus Unternehmen und Projekten, die Cloud-Technologie nutzende Anwendungen erstellen und betreiben.
Für die hier beschriebenen Aufgaben nutzbare Programme werden auch als Konfigurations- oder Infrastruktur-Management-Tools bezeichnet. Diverse Open-Source-Projekte und kommerzielle Anbieter verfolgen und unterstützen den Ansatz, wobei "Puppet" und "Opscode Chef" die momentan verbreitetsten sind.

Geschäftsrelevantes Monitoring

Eine komplexe technische Infrastruktur, ein Rechenzentrum, ein Netzwerk, schon ein einzelner Server bietet viele mögliche Messpunkte. Availability Management beschreibt die Maßnahmen, um die Verfügbarkeit der Systeme zu überwachen und auf Probleme rechtzeitig zu reagieren. Der Fokus liegt jedoch meist auf der grundsätzlichen Erreichbarkeit und Funktionsfähigkeit. In ITIL v3 wurde das Event Management neu aufgenommen, um spezifische Störungen die innerhalb von Anwendungen auftreten, zu behandeln. Es sollen die in den Systemen abgebildeten Geschäftsprozesse überwacht werden, kritische Abweichungen und Störungen sollen automatisch (und nicht erst, wenn ein Anwender Anzeichen dafür bemerkt) ins Incident Handling einfließen.
DevOps weist nun genau auf diesen Aspekt hin und rät, auch dieses Thema gemeinsam anzugehen. Ein Monitoring, was aus den Anwendungen keine oder irrelevante Daten erhält, nützt dem Business nichts. Entwickler, Administratoren und der Fachbereich sollten schon vor, beziehungsweise bei der Entwicklung von Anwendungen gemeinsam festlegen, welche Werte und Programm-Funktionen aus geschäftlicher Sicht überwachenswert sind und diese dann ins produktive Monitoring einbinden.
Aktuell erhältliche Monitoring-Systeme bieten mindestens eine entsprechende Schnittstelle, die selbst entwickelte Anwendungen nutzen können. Für das Open-Source Tool für Infrastruktur-Monitoring Nagios gibt es zudem eine Integrationsmöglichkeit für Cucumber. Mit diesem aus der Behaviour Driven Development Bewegung entstandenen Tool ist es zum Beispiel möglich, sogar über verschiedene Anwendungen hinweg Abfragen und die erwarteten Ergebnisse in einer domainspezifischen Sprache zu formulieren. Diese kann man in ein Plugin im Nagios-Format transformieren.

Weitere Werkzeuge und Tools

Neben Infrastruktur-Management und Monitoring gibt es eine Reihe weiterer zum Thema DevOps gehörender Bereiche, in denen Tools Optimierung versprechen:

- Defect Tracking
- Automated Issue Resolution
- Requirements Management
- Source Code Control
- Help Desk / Ticket Tools
- App Performance Monitoring

Dies sind alles keine Anwendungen, die aus der DevOps-Bewegung heraus initiiert wurden, die aber deren Ziele unterstützen können und die zudem zunehmend von der Idee

befruchtet und verändert werden. Es ist jedoch umstritten, ob sich der Technik-Einsatz lohnt oder ob er nicht sogar teilweise kontraproduktiv wirkt. Die Gefahr ist, wie so oft bei neuen Ansätzen, dass der eigentliche Gedanke in den Hintergrund gedrängt wird und die Hersteller die Lösung allein durch ihre "neue DevOps-Toolsuite XY" versprechen.

Welche Relevanz und Verbreitung hat das Thema momentan

HP hat zusammen mit Replay Solutions im April 2011 eine frei verfügbare Umfrage zum Stand von DevOps veröffentlicht. Aus unterschiedlichen Unternehmens-Größen haben über 1.100 englischsprachige IT-Mitarbeiter teilgenommen. Wichtige Erkenntnisse aus den Antworten sind:

- fast 50% der Befragten integrieren bereits die DevOps-Idee
- die Behebung von in der Produktion auftretenden Fehlern läuft deutlich schneller bei denen, die DevOps einsetzen
- der Prozentsatz der Fehler, die erst in der Produktion entdeckt werden, ist deutlich geringer bei denen, die DevOps einsetzen

Die Studie sagt nicht, ob es einen kausalen Zusammenhang oder eine andere Korrelation innerhalb der letzten beiden Punkte gibt, beides spricht aber letztendlich für DevOps. Die Ergebnisse der Umfrage, auch als Rohdaten, sind frei verfügbar und unter dem am Ende dieses Artikel angegebenen Link zu finden.

Ein häufig zitiertes, prominentes Beispiel für eine hohe DevOps-Durchdringung ist die Foto-Plattform Flickr. Auf deren Seiten findet man eine stetig aktualisierte Information in folgender Form: "Flickr was last deployed 31 hours ago, including 5 changes by 3 people. In the last week there were 86 deploys of 669 changes by 19 people." Diese hohe Release-Frequenz geht einher mit einer Spitzenposition im Social Media Bereich, was die Verfügbarkeit beziehungsweise Uptime angeht.

Durchsucht man die einschlägigen sozialen Netzwerke, stellt man fest, dass es auch im deutschsprachigen Raum schon einige große Unternehmen gibt, deren Mitarbeiter sich engagiert mit dem Thema beschäftigen: Deutsche Post, Axel Springer AG, Nokia ImmobilienScout24, 1&1 Internet AG, DuMont, und die Deutsche Flugsicherung sind Beispiele dafür. Dr. Johannes Mainusch, Vice President Operations bei der XING AG, schreibt auf Anfrage des Autors: "Besonders wichtig wird das [DevOps] für Organisationen, die den Übergang vom Startup-Zeitalter zu einer stabilen und professionellen Organisation geschafft haben und sich nun der Herausforderung immer stärkerer Spezialisierung der Fachbereiche stellen müssen. Dabei geht häufig der gemeinsame Blick aufs Ganze verloren." Die XING AG setzt auf DevOps und schafft intern ein wachsendes Bewusstsein dafür. Dr. Mainusch: "Ich habe seit langer Zeit endlich wieder Experten euphorisch gesehen."

Kritik an DevOps

Dass der Bereich, den DevOps adressiert, relevant ist und die Praxis oft noch viel Raum für Verbesserungen bietet, darüber gibt es keinen Dissens. Einige Verfechter agiler Methoden sehen das Thema jedoch vollständig von der eigenen Lehre abgedeckt und wenn der Betrieb konsequent ihrem Ansatz folge, dann gäbe es auch kein Problem. Auf der anderen Seite wird ein IT Service Manager antworten, dass all dies durch eine klare Umsetzung der ITIL-Prozesse geregelt werden kann.

Für einige Vertreter auf beiden Seiten ist das Thema also irrelevant, längst adressiert oder "alter Wein in neuen Schläuchen". Wenn die eingangs in diesem Artikel geschilderten Symptome in einer Organisation nicht auftreten, gibt es natürlich auch keinen Grund, sich um eine Verbesserung zu bemühen. Falls die Symptome doch auftauchen, ist aber wahrscheinlich genau diese unterschiedliche Überzeugung "Ich weiß, wie ihr zu arbeiten habt!" der Grund dafür und DevOps biete dann durchaus eine neue Perspektive.
Ein anderer Kritikpunkt, entsteht aus der Wahrnehmung, dass DevOps ein rein technisches Thema ist, das Hersteller als Vehikel nutzen, um neue Software-Werkzeuge und Funktionen zu verkaufen. Dies mag tatsächlich eine drohende Gefahr für das Thema sein aber das Engagement, mit dem in der DevOps-Community gerade die kulturellen Aspekte angegangen und vorangetrieben werden, lässt hoffen, dass die Bewegung das Ziel nicht verliert. Eine "Silver Bullet", also die endgültige Lösung des Themenkomplexes, wird aber auch DevOps nicht sein.

Zukunft und die Chance jetzt das Thema aufzugreifen

Der Artikel hier beleuchtet stark Symptome und die erkannten Probleme und nennt beispielhaft einige erste Lösungsansätze. Diese Gewichtung ist symbolisch für den Stand des Themas insgesamt. In der kommenden Zeit werden sich aber immer mehr Interessierte darüber austauschen, weitere Erfahrungen und Lösungen zusammengetragen und dabei dem Begriff eine klarere Form geben. Wahrscheinlich wird es dabei auch Diskussion über den genauen Weg geben und es wird Versuche der Vereinnahmung durch Unternehmen oder Gruppen geben.
DevOps ist kein Job-Titel, keine (zusätzliche) Rolle und erst recht kein weiteres Funktions-Silo sondern eine Einstellung und Umgangsform. Diese klare Darstellung hat sich beispielsweise erst im letzten Jahr durch Diskussionen und Austausch geformt. Wenn der Begriff also auch in deutschen Stellenanzeigen zunehmend auftaucht, sollte dies bedeuten, dass ein Bewerber auf diese Position den DevOps-Gedanken kennt und unterstützt und über den Tellerrand eines Entwicklers oder Administrators schaut.
Auch wenn das Gesamt-Ziel eine kulturelle Veränderung ist, empfiehlt es sich für viele Unternehmen, die DevOps aufgreifen möchten, die Nutzung der oben beschriebenen Tools als Einstieg zu nehmen. Über diese Werkzeuge und Maßnahmen zur Kommunikations-Verbesserung kann dann eine Änderung im Verhalten angestoßen werden und sich damit letztendlich eine geeignetere Form der Zusammenarbeit zwischen Entwicklung und Betrieb etablieren.
Und auch wenn hier im Artikel hauptsächlich von DevOps im Zusammenspiel mit agilen Entwicklungs-Methoden die Rede ist, lässt sich das Vorgehen auch mit der klassischen Wasserfall-Methode verbinden. Einige Teilaspekte, wie etwa die hohe Release-Frequenz, verlieren dabei jedoch an Bedeutung.
DevOps bietet jetzt die Chance die Vorteile und den Schwung der agilen Methoden in den IT-Betrieb zu bringen und sich dadurch eine Vorreiter-Position bei der Bereitstellung von Anwendungen und IT-Services zu verschaffen.