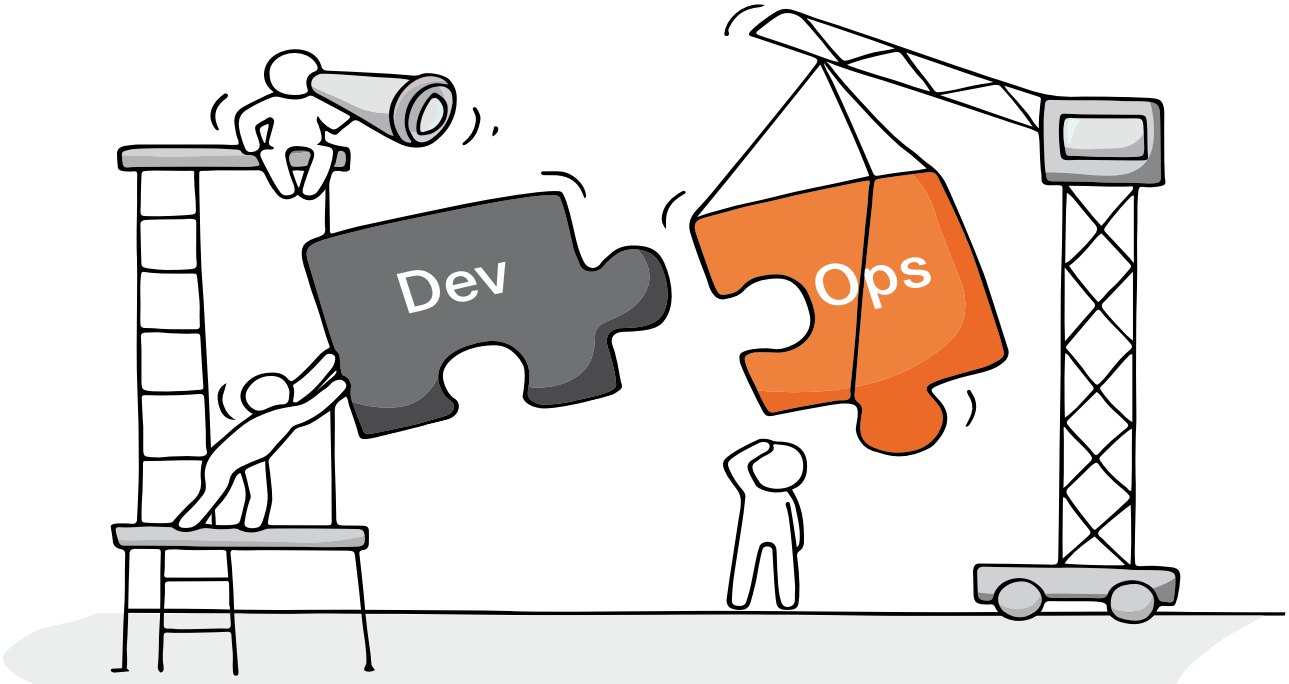

DevOps: Survival-Training für die Finanzbranche

Der Zwang zur Optimierung der Delivery Pipeline als Auswirkung der digitalen Transformation

von Erik Benedetto

Software wird heute zumeist agil entwickelt. Was noch vor wenigen Jahren als neue Vorgehensweise galt, ist heute ein De-facto-Standard in allen Branchen und für alle Unternehmensgrößen. Der Wunsch nach Geschwindigkeit, nach schnellen Ergebnissen in kurzen Sprints hat es ermöglicht, dass innerhalb eines kurzen Zeitraums agile Methoden die Softwareentwicklung substanziell verändert haben. Seit einiger Zeit geistert nun ein Begriff durch die Fachliteratur, der das Potenzial hat, die gesamte Prozesskette zu verändern – und zwar inhaltlich, strukturell und technisch: DevOps.

DevOps kann ein Bestandteil der Maßnahmen sein, um die digitale Transformation der traditionellen Finanzbranche zu vollziehen und zu meistern. Aber was genau verbirgt sich dahinter? Wo liegt das Potenzial? Was sind die Bestandteile von DevOps und wie lässt sich diese Methode in den Application Lifecycle integrieren?

In einer mehrteiligen Artikelreihe setzen wir uns mit allen Aspekten dieses Themas auseinander. Der erste Teil setzt sich mit folgenden Fragen auseinander:

- DevOps – Ursprung und Ziele?
- Wieso gewinnt DevOps im Kontext der digitalen Transformation eine neue Bedeutung?

Was ist DevOps?

DevOps ist ein Kunstwort, bestehend aus den Begriffen Development und Operations. Der seit 2008 verwendete Begriff setzt sich zusammen aus dem Wortbestandteil „Dev“, der die Softwareentwicklung (Development) repräsentiert, und „Ops“, der für den IT-Betrieb (Operations) steht. Die Kombination zu „DevOps“ symbolisiert intuitiv einen Schulterschluss zwischen Softwareentwicklung und IT-Betrieb. Und tatsächlich ist das der Grundgedanke von DevOps und der Auslöser der dazugehörigen Bewegung: ein Zusammenrücken der beiden in der traditionellen Wahrnehmung grundverschiedenen Bereiche Softwareentwicklung und IT-Betrieb. Einerseits ist die Wortschöpfung DevOps sehr griffig. Andererseits lässt sie große Interpretationsspielräume zu, was zu Missverständnissen führen kann. Aktuell sieht die DevOps-Bewegung ihre Hauptaufgabe darin, die vielen Interpretationen zu kanalisieren und eine klare Definition von DevOps zu formulieren. Um zu wissen, warum dieser Schulterschluss notwendig ist, ist ein Verständnis des zugrunde liegenden Interessenkonflikts zwischen Dev (Entwicklung) und Ops (IT-Betrieb) nötig.

Development: Ziele der Softwareentwicklung

Ziel der Entwicklung ist die Realisierung neuer Features durch schnelle und häufige Releases. Die Aufgabe von Softwareentwicklern besteht darin, die vom Auftraggeber gewünschten Funktionen und Features möglichst schnell umzusetzen. Durch die Entwicklung wird erst eine neue Funktion verfügbar, durch die sich ein potenzieller Mehrwert für die Endnutzer ergibt.

Je häufiger und schneller neue Features entwickelt und komplettiert werden, desto schneller kann dem Endnutzer also auch ein Mehrwert zur Verfügung gestellt (und monetarisiert) werden. Gleichzeitig bedeutet ein schneller Entwicklungsprozess, zügig auf Markt- und Kundenanforderungen reagieren zu können. Daher wird dies – ebenso wie die Entwickler, die diesen Anspruch erfüllen können – in vielerlei Hinsicht positiv wahrgenommen. Oft wird der Mehrwert einer Funktion allerdings schon bei der ersten Abnahme durch den Auftraggeber anerkannt und nicht erst, wenn diese dem Endnutzer letztendlich zur Verfügung steht. Der Grund hierfür ist einfach: Bis zum nächsten Release kann es schon mal ein paar weitere Tage dauern. Dadurch ist es allerdings für die Entwickler auch weitestgehend irrelevant, ob die neuen Funktionen/Features tatsächlich auf dem Produktionssystem verfügbar sind oder nicht.

Operations: Ziele des IT-Betriebs

Ziele des IT-Betriebs sind Stabilität und Sicherheit der Anwendungen und Infrastruktur durch wenige Releases bis hin zur Releasevermeidung.

Die Aufgabe des IT-Betriebs besteht darin, die von der Entwicklung gelieferte Funktion in Form von Software auf der Produktivumgebung für die Endnutzer verfügbar zu machen. Dazu zählen das Deployment im Rahmen neuer Softwarereleases und gleichzeitig die Sicherstellung des laufenden Betriebs gemäß den definierten Qualitätsanforderungen.

Der Betrieb trägt also die unmittelbare Verantwortung für die dauerhafte Verfügbarkeit der Anwendungen und deren Sicherheit. Der Erfolg wird daran gemessen, inwieweit die formalen Qualitätsanforderungen erreicht werden; meist in Form der definierten Service Levels und KPI, die in den Service Level Agreements dokumentiert sind. Da der Endnutzer in der Regel die volle Verfügbarkeit und Sicherheit der Anwendungen erwartet, ist es oberste Priorität des IT-Betriebs, diese im Rahmen der Service Levels sicherzustellen.

Aus diesen Gründen geht der IT-Betrieb entsprechend vorsichtig mit Veränderungen um. Abhängigkeiten zwischen verschiedenen Softwaredeployments werden ausgiebig getestet und Deployments in Releasepaketen gebündelt. Insgesamt werden die Anforderungen an die Dokumentation und Tests sehr hoch angesetzt, bevor es überhaupt zu einem Deployment kommt.

Servicebeeinträchtigungen sollen schnell beseitigt beziehungsweise von vornherein möglichst ausgeschlossen werden können.

Dies führt zu einer nachweislichen Entschleunigung der anfänglich schnellen Softwareentwicklung. Die Gründe hierfür sind nachvollziehbar, denn ist die Verfügbarkeit einer Anwendung erst einmal beeinträchtigt, fällt das direkt auf den IT-Betrieb zurück. Die Folge: eine stark negative Wahrnehmung durch die Auftraggeber, besonders wenn die Nutzer der Anwendung ein „Problem“ melden, noch bevor die verwendeten Monitoring-Systeme Alarm schlagen. Um die Wahrscheinlichkeit für unerwartete Ausfälle zu minimieren, setzt der IT-Betrieb deshalb oft alles daran, den Zustand einer stabil laufenden Anwendung vor Änderungen zu schützen: durch wenige, gebündelte und ausgiebig getestete und dokumentierte Releases.

Blame Game: der traditionelle Konflikt zwischen Entwicklung und Betrieb

Der Vergleich zeigt, dass beide Einheiten entgegengesetzte Anreize haben. Die Entwicklung ist an schnellen und häufigen Releases interessiert, der Betrieb hingegen würde Releases am liebsten vermeiden. Beide Seiten verfolgen damit jedoch das gleiche Ziel, nämlich ihren eigenen Wert für das Unternehmen unter Beweis zu stellen.

Genau das führt aber regelmäßig zu Konflikten, denn in der Regel treffen Dev und Ops unter Zeitdruck aufeinander, zum Beispiel beim Deployment eines neuen Release, oder wenn es ein Problem wie beispielsweise einen Systemausfall gibt. Dann beginnt oft das typische Blame Game, bei dem beide Lager sich gegenseitig die Schuld an der Situation geben.

Wer solche Situationen noch nicht erlebt hat, kann sich glücklich schätzen. Wer das Blame Game hingegen kennt, weiß: Schuldzuweisungen bringen nichts, die Zeit ist besser in die Lösung des Problems investiert (siehe Schaukasten).

Der IT-Betrieb als Flaschenhals

Seit die Softwareentwicklung verstärkt auf agile Methoden setzt, eskaliert die Situation immer häufiger, denn diese setzen auf kontinuierliche Interaktion zwischen Auftraggebern und Entwicklern sowie auf kurze Releasezyklen. In der Regel setzt sich die damit einhergehende Philosophie aber nicht in den Betrieb fort. Die Vorteile agiler Methoden werden ausgebremst. Softwarereleases werden zwar in kurzen Iterationen erstellt, der geschaffene Mehrwert wird aber erst viel später auf der Produktivumgebung sichtbar. Die immer kürzer werdenden Releasezyklen offenbaren den IT-Betrieb zunehmend als Flaschenhals auf dem Weg der Software zum Endnutzer. Außerdem erhöhen

Blame Game – zwei Beispiele

Die Entwicklung gibt ein neues Release zum Deployment an den Betrieb weiter, dem es jedoch nicht gelingt, die Software auf der Produktivumgebung lauffähig zu machen. Als der Betrieb die Entwickler kontaktiert und die auftretenden Fehler beschreibt, blocken diese ab: Die Software laufe auf der Entwicklungsumgebung fehlerfrei, der Fehler müsse beim Betrieb liegen. Beide Seiten schieben sich den Schwarzen Peter zu. Nach Krisensitzungen und Unstimmigkeiten zwischen den Abteilungen ergibt eine Untersuchung, dass sich Entwicklungs- und Produktivumgebung in einem wichtigen Detail unterscheiden. Das war keiner der beiden Seiten vorher bewusst.

Im Produktivsystem taucht ein Performanceproblem auf. Unter großem Druck arbeiten die Entwickler mehrere Nächte durch und liefern schließlich einen Patch. Der Betrieb fürchtet jedoch, dass der Patch die Stabilität des Systems gefährden könnte, da er Änderungen an einer kritischen Komponente umfasst. Deshalb wird zunächst eine genaue Qualitätskontrolle auf einer Testumgebung verlangt, um die Lösung in realistischen Testszenarien zu überprüfen. Doch die benötigte Last lässt sich in der Testumgebung nicht adäquat darstellen. Einen Monat später ist der Patch noch immer nicht eingespielt. Die Entwickler sind enttäuscht, da „es ja offenbar doch nicht so eilig war“.

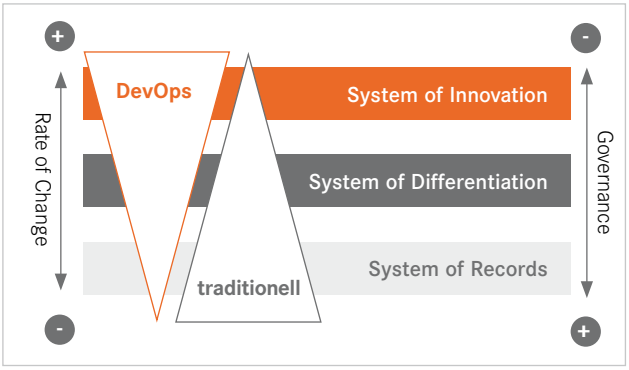

Abbildung 1: Fokus der Releasestrategien von Banken im Schichtenmodell der IT-Architektur, Quelle: Gartner

häufig stattfindende Releases das Potenzial für das direkte Aufeinandertreffen zwischen Softwareentwicklung und Betrieb.

Doch nun ist ein neuer Aspekt hinzugekommen, der den Handlungsbedarf besonders bei den traditionellen Banken verschärft und schnelles Umdenken erfordert: die digitale Transformation.

Das traditionelle Geschäftsmodell der Banken und ihre IT-Architektur

Betrachtet man das bisherige Geschäftsmodell und die IT-Architektur der traditionellen Marktteilnehmer in der Finanzbranche, stellt man fest, dass der Fokus bisher vor allem auf den Produkten lag. Bankprodukte und deren Ausgestaltung definierten den Wettbewerb unter den Marktteilnehmern. Zunehmender Kostendruck erforderte eine Standardisierung der Produkte, hohe Transaktionsvolumina waren erforderlich, um die Business Cases zu rechtfertigen. Dies hatte Auswirkungen auf die zugrunde liegende, zumeist monolithische IT-Architektur, um die Verfügbarkeit der Anwendungen im Hintergrund sicherstellen zu können. Hostanwendungen waren das Maß der Dinge und die Basis der sogenannten „Systems of Records“. Nur wenige Releasetermine pro Jahr waren erforderlich, um die IT-Architektur an die sich ändernden Rahmenparameter anzupassen. Noch heute sind zwei bis drei Releases pro Jahr der Standard in der Finanzbranche.

Releasestrategie	Fokus	Auswirkung
> Kurze Releasezyklen (2–4 Wochen) > Flexible Konfiguration von neuen Produkten und Services	> Garantiert Agilität und kurze Produkteinführungszeit	> Unterstützt neue Geschäftsmodelle und Produkte > Fördert Experimente und Innovation
> Mittlere Releasezyklen (4–8 Wochen) > Verbindet System of Innovation mit System of Records	> Interne Plattform-Entwicklung	> Wiederverwendbare Querschnittsfunktionen > Standardisierte, hochqualitative Information
> Traditionelle Releasezyklen (>2 Monate) > Stabile Funktion für das Kerngeschäft	> Garantiert Stabilität und Kosteneffizienz	> Unterstützt Geschäftskontinuität, Konsolidierung und Compliance

Abbildung 2: Releasestrategien im Vergleich

Die digitale Transformation in der Bankenbranche verschärft den Konflikt

Durch die digitale Transformation steigt in der Bankenbranche der Druck, den Konflikt zwischen Entwicklung und IT-Betrieb aufzulösen. Die Digitalisierung der Gesellschaft und der Wirtschaft sowie die Verfügbarkeit neuer Technologien haben die Anforderungen an die Banken hinsichtlich der Anpassungsgeschwindigkeit massiv erhöht.

- Neue, branchenfremde Wettbewerber, wie zum Beispiel Technologieunternehmen, verdrängen die Teilnehmer etablierter Märkte und zerstören bisherige Marktstrukturen.
- Das geänderte Nutzerverhalten der Kunden als „Digital Natives“ erfordert neue Prozesse.
- Neue Technologien ermöglichen eine modulare, schnell skalierbare IT-Architektur aus der Box.
- Die Datenerhebung, -auswertung und -nutzung ermöglicht neue Geschäftsmodelle.
- Daten und das „Lernen aus den Daten“ werden zum Wettbewerbsvorteil und Differenzierungsmerkmal.
- Zunehmende regulatorische Anforderungen (AnaCredit, BCBS 239, MaRisk) erfordern eine schnelle Umsetzung, um Strafzahlungen zu vermeiden.

Abbildung 3: Digitale Transformation – Status quo

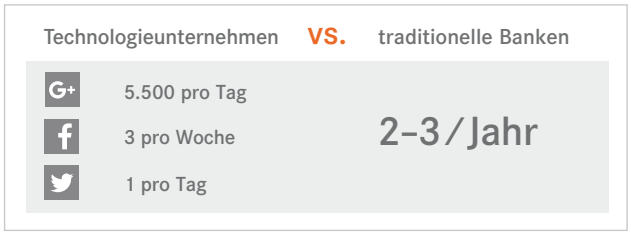

Abbildung 4: Deployment-Frequenzen der Marktteilnehmer im Vergleich

Die neuen Konkurrenten der Finanzbranche

Die digitale Transformation hat gleichzeitig auch den traditionellen, produktgetriebenen Bankenmarkt für neue, branchenfremde Marktteilnehmer geöffnet, welche die Existenz der Banken bedrohen.

- Die neuen Marktteilnehmer kommen aus dem technischen Umfeld, in dem sie die neuen Technologien definieren, entwickeln und betreiben.
- Diese Marktteilnehmer verfügen über große Datenmengen zum Nutzerverhalten und kennen sich mit den Kundenbedürfnissen durch detaillierte Datenauswertungen aus.
- Aufgrund ihrer technologischen Positionierung sind sie auf eine schnelle Time-to-Market „getrimmt“.
- Sie haben schnelle, kurze Releasezyklen, die durch eine modulare, integrierte IT-Architektur unterstützt werden.
- Die Marktteilnehmer verfügen über hohe liquide Mittel aus ihrem Kerngeschäft und drängen – nach dem Try-and-Error-Prinzip – mit neuen Geschäftsmodellen schnell in die traditionellen Branchen (zum Beispiel Apple, Amazon, Google).

Das Wettrennen um die Kunden hat begonnen

Kurz gesagt: Der Handlungsdruck in der Bankenbranche, den Konflikt zwischen Softwareentwicklung und IT-Betrieb aufzulösen, war nie größer, denn er kostet Zeit,

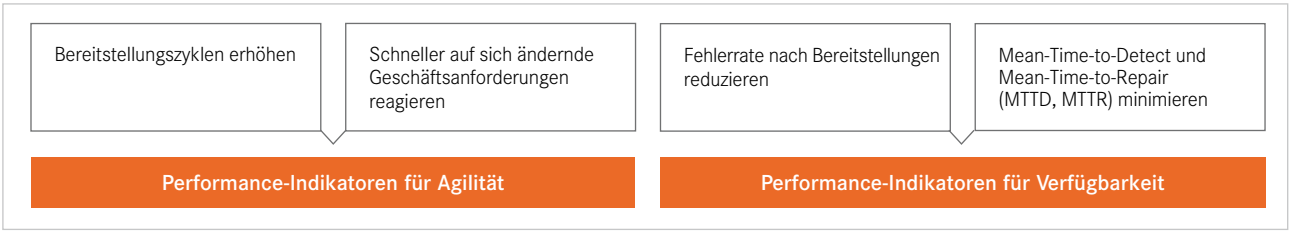

Abbildung 5: DevOps – Ziele und Metriken

- den die Branche für die digitale Transformation vom produktbasierten zum datengetriebenen Geschäftsmodell braucht,
- um die monolithische, transaktionsbasierte IT-Architektur in eine flexible, modulare umzubauen,
- um die Time-to-Market durch kurze Releasezyklen drastisch zu erhöhen, um neue Marktteilnehmer abzuwehren.

DevOps: Ziele und Metriken

DevOps kann mit seinen Mechanismen helfen die Delivery Pipeline zu optimieren. Dabei können drei Bereiche unterschieden werden:

- Zusammenarbeit,
- Automatisierung und
- Prozesse.

Durch die Betrachtung der gesamten Wertschöpfungskette beziehungsweise Delivery Pipeline unter dem Aspekt der Zusammenarbeit werden Mechanismen implementiert, die den Konflikt zwischen Entwicklung und Betrieb auflösen. Gleichzeitig kann dadurch schneller auf neue Businessanforderungen (Features, Funktionalitäten) reagiert werden. Als Ergebnis entsteht ein DevOps-Team, das organisiert, ausgerichtet und automatisiert auf die Lieferung und Inbetriebnahme von Applikationen ist.

Die Themen „Automatisierung“ und „Prozesse“ hingegen zielen auf eine hohe Releasefrequenz und -geschwindigkeit (Agilität), und zwar bei gleichzeitiger Verbesserung der Stabilität und Reproduzierbarkeit von Fehlern (Verfügbarkeit).

Eine Erhöhung der Effizienz und Kapazität in der Delivery Pipeline vermeidet gleichzeitig Rework, zum Beispiel durch Automatisierung von Vorgängen und Standardisierung der Dokumentation. Stabilität der Anwendungslandschaft wird durch Transparenz bei den Infrastrukturabhängigkeiten über den gesamten Wertschöpfungsprozess erreicht.

Feedbackschleifen und eine kontinuierliche Messung der Prozessqualität mittels KPI ermöglichen eine kontinuierliche Transparenz, Effizienzmessung und Verbesserung der Delivery Pipeline.

Der zweite Teil der Artikelreihe untersucht die Bestandteile und Mechanismen von DevOps und klärt dabei die Frage, wie sich DevOps in der Finanzbranche implementieren lässt.

Ansprechpartner

Erik Benedetto
Senior IT Consultant,
Center of Competence Application Management
> +49 (0) 173 7056 323
> erik.benedetto@msg-gillardon.de