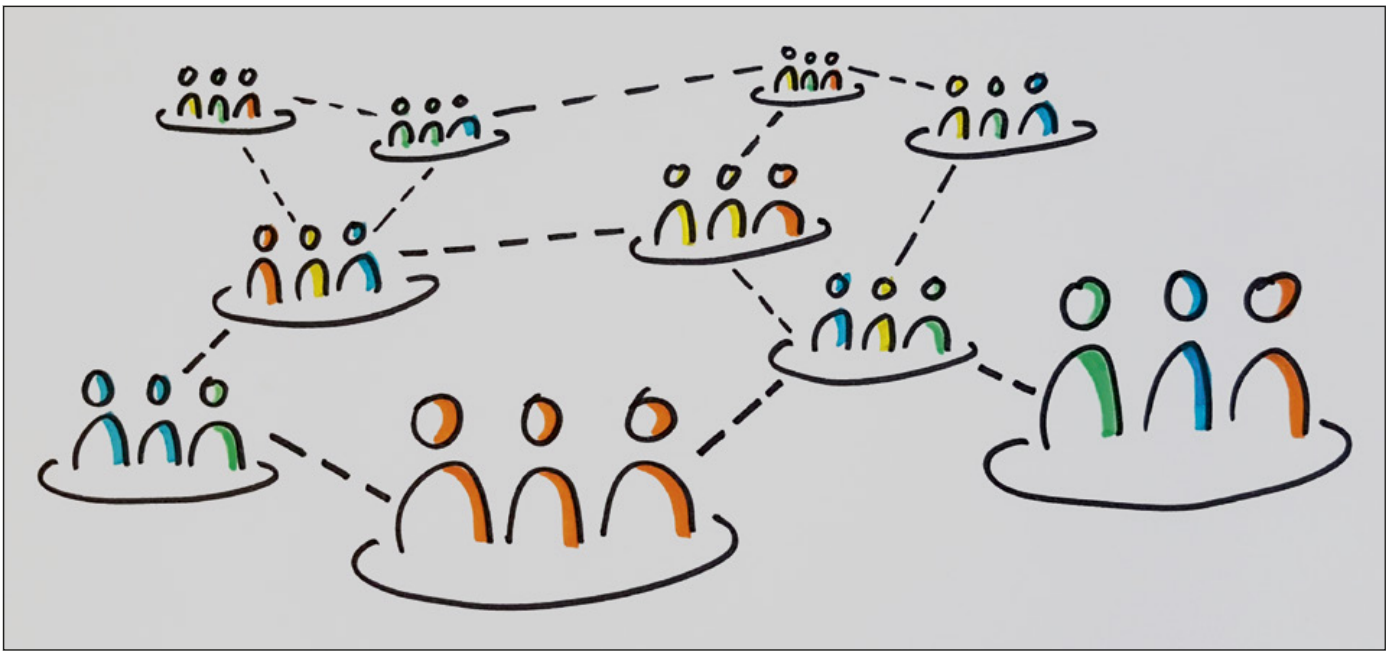
Sonderdruck aus OBJEKTspektrum 01/2018

DevOps – mehr als nur Technik

DevOps einzuführen, betrifft vieles – vor allem Aspekte der Kooperation

Die gute Zusammenarbeit zwischen Entwicklung und Betrieb ist häufig einer der entscheidenden Faktoren für den Erfolg eines Produkts – und nicht immer ein reiner Selbstläufer. Wir, zwei externe Agile Coaches, haben mehrere Entwicklungsteams und ein Team der Betriebsführung auf ihrem Weg in Richtung DevOps und verbesserter Zusammenarbeit begleitet. Dazu bietet DevOps diverse Möglichkeiten.



Von DevOps versprechen sich Unternehmen in erster Linie eine Verbesserung und Beschleunigung der Lieferprozesse, da der Graben zwischen

- Entwicklung (Development oder kurz „Dev“) und
- Betrieb (Operations oder kurz „Ops“)

geschlossen wird. Laut einer aktuellen weltweiten Studie [Pup17] haben bereits 50 Prozent der befragten Unternehmen eigene DevOps-Initiativen gestartet, 21 Prozent der Unternehmen planen solche.

Die Situation am Anfang

Auch in unserem Beispielprojekt findet eine DevOps-Initiative statt. Hierbei arbeiten mehrere Dev-Teams hauptsächlich agil nach Scrum, teilweise einzeln und unabhängig, teilweise aber auch zusammen an einem Produkt in Form von skaliertem Scrum. Unterstützt werden die Dev-Teams durch ein Ops-Team, das zu Beginn die Verantwortung des Betriebs im klassischen Sinne tragen sollte (Infrastruktur, Inbetriebnahme, Betrieb, Support).

Der technische Kontext unseres Projekts ist eine ambitionierte Plattformentwicklung auf der „grünen Wiese“ in der Cloud, basierend auf Big-Data-Technologien, Microservices und eingebettet in einer eventgetriebenen Architektur.

Nach einiger Zeit befand sich das Projekt in einer Situation, mit der die Beteiligten nicht wirklich glücklich waren: Die Liefertermine waren unklar, weil die Entwicklung nicht ganzheitlich betrachtet wurde und mehr und mehr Funktionalität auf spätere Termine verschoben werden musste. Eigentlich benötigte technische Grundlagen (Ausbau der Lieferkette, Testautomatisierung, Monitoring usw.) wurden aufgrund des Zeitdrucks außen vor gelassen. In der Folge arbeiteten sich die Teams sozusagen von Baustelle zu Baustelle.

Generell waren die Zielstellungen von Dev und Ops unterschiedlich und die Teams sprachen nicht dieselbe Sprache. Darüber hinaus war das angestrebte Serviceniveau für den Betrieb der Anwendungen noch weit entfernt.

Beide Teams hatten zu Beginn unterschiedliche Verantwortung bei der Entwicklung des Produkts. Die Entwicklungsteams sollten die Software entwickeln, nach agilen Spielregeln (also potenziell auslieferbar am Ende jedes Sprints), und Ops sollte die Infrastruktur bereitstellen und zu bestimmten Release-Terminen die Software produktivsetzen und den sauberen Betrieb sicherstellen. Um das zu erfüllen, muss Ops sich wiederum „einmischen“ dürfen in den Entwicklungsprozess, um zu gewährleisten, dass man mit dem Ergebnis der Entwicklung umgehen kann. Genau das ist eine Situation, die zu Reibungen zwischen den Teams führen kann.

Bei diesen unterschiedlichen Vorstellungen geriet immer mehr in Vergessenheit, dass

man eigentlich ein gemeinsames Ziel hatte und nur die Zusammenarbeit zum Erfolg führt. Hier sollten mithilfe von DevOps-Prinzipien Verbesserungen erzielt werden.

Was ist DevOps – und wie konnte es uns weiterbringen?

Warum erschien uns in dieser Situation DevOps als Mittel der Wahl? Die DevOps-Community ist breit gefächert und entwickelt das Thema ständig weiter, es gibt keinen „Single Body of Knowledge" wie den bekannten „Scrum Guide" [Suth17]. DevOps versteht sich heute als eine Menge von Prinzipien, Praktiken und kulturellen Werten, die es Unternehmen erlauben, ihre Softwarelieferzyklen zu optimieren, um schnelles Feedback in der Produktentwicklung zu bekommen [Pup17]. Dazu gehören technische Praktiken, wie die Automatisierung von Lieferprozessen, Continuous Delivery, die Modellierung von Infrastruktur (Infrastructure-as-Code) oder umfassendes Monitoring und Datenanalysen.
Darüber hinaus erweitert sich der Blick bei DevOps immer mehr auf den gesamten Wertstrom zum Kunden, den es ganzheitlich zu optimieren gilt. Das Unternehmen muss aber eine Kultur der Sicherheit schaffen, in der die Mitarbeiter die Freiheit haben, kontinuierlich zu experimentieren, um schnell zu lernen und sowohl Produkt als auch Prozesse zu verbessern.
Wir wollten uns bei unserem Start mit DevOps erst einmal auf die Aspekte fokussieren, welche die Zusammenarbeit in und zwischen den Teams sowie die Kultur in der Organisation betreffen:

- *Zielbild, Verantwortung und Führung:* Wie können wir mit dem Management und den Teams gemeinsame Visionen und Zielstellungen erarbeiten, damit wir Silo-Denken und Konflikte zwischen den Teams vermeiden und das System global optimieren?
- *Kooperationsmodelle:* Welche Teamstrukturen und Ansätze können helfen, die Zusammenarbeit von Dev und Ops im Projekt zu verbessern und eine Balance zwischen Teamzielen und Querschnittsthemen zu erreichen?
- *Fokus in Selbstorganisation:* Wie bringen wir mehr Fokus in selbstorganisierte (Nicht-Scrum-) Teams, damit wir effizienter arbeiten und „Goldene Wasserhähne" vermeiden?
- *Kultur der kontinuierlichen Verbesserung:* Wie schaffen wir eine Kultur in unserer Organisation, die alle Beteiligten ermutigt, Verantwortung zu übernehmen, zu experimentieren und neue Dinge auszuprobieren, um schnell zu lernen und etwas zu verbessern?

Kein DevOps ohne Zielbild, Verantwortung und Führung

Als erste Maßnahme erschien es uns wichtig, mit den Entwicklern und Ops-Mitarbeitern ein gemeinsames Verständnis von DevOps zu entwickeln, um für den kommenden Veränderungsprozess eine klare Orientierung zu haben. Das war nötig, da DevOps als Ziel nicht wirklich greifbar ist, weil es keine allgemeine Definition gibt. Zu diesem Zweck organisierten wir einen moderierten Workshop, bei dem die Teilnehmer (inkl. Product Owner und Teamleiter) ihr Verständnis von DevOps klären und wir erste Impulse für den weiteren Prozess generieren konnten.
In den Diskussionen wurde schnell klar, dass man noch weit entfernt war von dem, was viele unter DevOps verstanden. Eine große Herausforderung lag darin, dass die Product Owner der Dev-Teams noch nicht genug „empowered" waren, das heißt, sie hatten noch nicht genug Gestaltungsspielraum, um die Produkte Ende-zu-Ende (von Vision über Anforderung und Entwicklung bis einschließlich Betrieb) zu steuern. Ohne diesen Spielraum ist es jedoch nicht möglich, die Lieferkette beziehungsweise den Wertfluss global zu optimieren.
Das war nicht zu lösen, ohne zuvor eine tiefer liegende Herausforderung anzugehen: „Entwicklung bei Dev" und „Betrieb bei Ops" statt der erforderlichen gemeinsamen Verantwortung. Nach Folgeterminen mit den Product Ownern konnte man sich auf einen ersten Zwischenschritt einigen. Die Entwicklungsteams sollten vorerst die Anwendung in ihrem aktuellen Vor-Produktiv-Stadium selbst betreiben, Erfahrungen sammeln sowie Betriebsführungsprozesse aufbauen und grundlegende Serviceniveaus erreichen. In einem zukünftigen Schritt würde man übergehen zu einem gemeinsamen Betrieb der dann produktiven Software und einer schrittweisen Erhöhung des Serviceniveaus.
Ein weiteres Resultat des Workshops war die Erkenntnis, dass die technischen Grundlagen (Automatisierung, Monitoring, Continuous Delivery usw.) der Teams noch nicht ausgereift waren. Eine Konsequenz war die Gründung einer Community of Practice „DevOps" (siehe auch das Kapitel zu „Kooperationsmodellen"), um sich regelmäßig fachlich und technisch auszutauschen und mehr Bewusstsein für Betriebsführungsthemen in den Dev-Teams zu schaffen.
In der folgenden Zeit spielte das Ops-Team eine immer wichtigere Rolle dabei, das Thema DevOps im Projekt weiterzuentwickeln. Hilfreich war hier eine klar verständliche Vision (anfangs dargestellt auf einem einzelnen, vom Team entworfenen Plakat), mit der das Team bei verschiedenen Stakeholdern für DevOps warb. Engagiert machte sich das Team auch daran, die Zusammenarbeit mit Dienstleistern zu verbessern und selbstbewusster eigene Ideen im Veränderungsprozess einzubringen. Dem Team war bewusst geworden, wie viel es selbst erreichen kann, statt auf einen externen Impuls warten zu müssen. Das war ein wichtiger Schritt auf dem Weg zu effektiver Selbstorganisation.

Kooperationsmodelle – Wie können Dev und Ops zusammenarbeiten?

Um die angestrebte Verbesserung der Zusammenarbeit von Entwicklung und Operations schon von der Teamstruktur her zu unterstützen, gibt es bei DevOps verschiedene Möglichkeiten (siehe auch [Kim16]). Folgende Varianten (**siehe Abbildung 1**) haben wir im Projekt ausprobiert:
Liaison: In einem ersten Ansatz versuchten wir eine Art Liaison zu bilden, das heißt, ein oder mehrere Mitarbeiter des Ops-Teams arbeiten mit einem bestimmten Entwicklungsteam enger zusammen. Dementsprechend trifft man sich häufiger, um konkrete Lösungen gemeinsam zu entwickeln oder umzusetzen, wie die Automatisierung einer Datenbank-Konfiguration oder das Lösen eines akuten Firewall-Problems. Der Ops-Mitarbeiter ist aber nicht fester Teil des Teams, das heißt, er nimmt nicht an den Ritualen und festen Team-Meetings teil, sondern arbeitet hauptsächlich im Ops-Team und unterstützt dort auch Aktivitäten für andere Teams oder strategische Themen.
Dieses Modell hat bei uns besonders dann gut funktioniert, wenn das Entwicklungsteam schon teilweise eigenständig Infrastrukturaufgaben bearbeiten kann und nur hin und wieder Unterstützung braucht. Im Idealfall arbeiten die Ops-Mitarbeiter nur noch als technische Coaches oder Berater der Dev-Teams. Oft sind auch die Ressourcen bei Ops begrenzt beziehungsweise vielen Teams zugeteilt. Der Nachteil dieses Modells ist, dass der Ops-Mitarbeiter aufgrund der begrenzten Verfügbarkeit schnell zum „Flaschenhals" werden kann.
Einbettung: Die Einbettung kann man als eine intensivere Variante des Liaison-Modells sehen. Hier hatten wir die Ops-Mit-

arbeiter fest in die Entwicklungsteams integriert, mit Teilnahme an allen Ritualen (Planung, Reviews, Retrospektiven usw.). Dadurch können sie sich besser auf das Team fokussieren, sind entsprechend in die Teamarbeit eingebunden und schneller erreichbar. Dennoch kann sich diese Fokussierung wie in unserem Projekt zum Problem entwickeln, wenn die Verbindung zum Ops-Team und dessen Themen verloren geht. Häufig sind Ops-Themen auch Querschnitts-Themen (z. B. „Sollte jedes Team sein eigenes Deployment-Konzept entwickeln?"), bei denen es wenig sinnvoll ist, wenn sich jedes Team seine individuelle Lösung sucht.

Als wir diese Variante bei uns einsetzten, waren noch nicht genug Ops-Mitarbeiter für die Teams vorhanden. Deshalb übernahmen vorerst einzelne Entwickler der Teams diese Rolle. Dieser Übergang war ein sehr wichtiger Schritt auf dem Weg zu mehr Vertrauen zwischen dem Entwicklungs- und dem Betriebsführungsteam.

Virtuelle Teams: Um dem Problem der Isolation der Ops-Mitarbeiter in den Dev-Teams (siehe Einbettung) entgegenzuwirken, besteht die Möglichkeit, ein virtuelles Team oder eine Community of Practice zu bilden, je nachdem, wie stark die Entwicklungsteams gekoppelt sind. Sind die Teams eher abhängig, arbeiten zum Beispiel am selben Produkt (in unserem Projekt durch einen skalierten Scrum-Ansatz gegeben), kann die Arbeit die Bildung eines virtuellen Teams erforderlich machen. Dieses Team stimmt sich regelmäßig ab, arbeitet synchron zu der Sprint-Taktung der Entwicklungsteams und wird gesteuert aus den Sprint-Backlogs der Teams. Im Gegensatz zu den Scrum-Teams arbeitet es jedoch hauptsächlich getrieben von Unterstützungsanfragen aus den Dev-Teams. Es beantwortet in erster Linie die Fragen „Was ist bis Datum X beziehungsweise zu Sprint-Ende an Betriebsführungsthemen zu tun, was davon ist gemeinschaftlich anzugehen, und wie setzen wir das konkret um?"

Bei dieser Konstruktion muss man jedoch darauf achten, dass die Steuerung des virtuellen Teams transparent ist (klare Ziele, Fokus), egal ob das selbstorganisiert funktioniert oder eine Art Product Owner braucht, der im Zweifelsfall entscheidet (siehe weiter unten).

Communities of Practice: Ist die Kopplung der Teams eher lose, kann eine Community of Practice (CoP) das Mittel der Wahl sein. Hier treffen sich die Mitarbeiter auch regelmäßig, wenn auch seltener, sind aber stärker auf den Austausch zu den Themen fokussiert. Es geht um Fragen wie „Woran arbeitet ihr? Welche Lösungen/Tools gibt es? Was können wir voneinander lernen?"

In unserem Fall hatten wir mit einer CoP „DevOps" (Team-Ops und Ops-Mitarbeiter) gestartet, dann aber festgestellt, dass es wenig Sinn ergibt, Lieferungs-relevante Aufgaben in einem lockeren Austausch weiterzuentwickeln. So verlagerte sich die Arbeitsweise der Gruppe eher in Richtung eines virtuellen Teams. Die CoP spielte jedoch auch weiterhin eine wichtige Rolle für den Wissensaustausch zwischen den Zuständigen. Sie führte ebenfalls zu dem Beschluss, die Bereitstellung der Umgebungen iterativ-inkrementell anzugehen.

Unabhängig davon, welche Ausprägung die Teamzusammenarbeit im Projekt findet: Wichtig ist die Basis in Form einer gemeinsamen technischen Plattform, in der Ops-Mitarbeiter Services bereitstellen, die die Teams einfach nutzen können. In unserem Fall reichte das von einfachen Automatisierungsskripten über Self-Services innerhalb der Cloud hin zu geteilten Monitoring-Clustern. Außerdem sollten geteilte Artefakte wie Quellcode und Konfiguration (sowohl der Anwendung als auch der Infrastruktur bzw. Prozesse) in der Quellcodeverwaltung zugänglich, diskutierbar und weiterentwickelbar sein.

In jedem Fall sollten die Teamstrukturen nicht top-down von einzelnen entschieden werden, sondern das Resultat von Reflexionen und Entscheidung der Teams selbst sein (z. B. in den Retrospektiven). Bei uns hat es sich bewährt, die eine oder andere Variante oder Kombinationen erst einmal für begrenzte Zeit auszuprobieren.

Fokus und Steuerung in selbstorganisierten Teams

Für unser neues virtuelles Team sowie für unser eigentliches Ops-Team stellte sich die Frage, wie die Teams am besten gesteuert werden. In einer DevOps-orientierten Organisation streben wir selbstorganisierte Teams an, das heißt, sie strukturieren ihre Arbeit selbst und finden eigene Wege/Lösungen für Probleme. Allerdings braucht es den richtigen Rahmen, an dem die Selbstorganisation wachsen kann, also zum Beispiel ein klares Bild der geschäftlichen Ziele des Projekts und andere organisatorische Randbedingungen.

Im Folgenden stellen wir ein paar Konzepte vor, die uns im Projekt geholfen haben, den Teams eine Orientierung zu geben.

Kontext: Ein Team kann nur dann gute Entscheidungen eigenverantwortlich treffen, wenn es den Gesamtkontext der Entscheidung kennt. Deshalb haben die Teammitglieder im Idealfall freien Zugang zu allen relevanten Informationen des Unternehmens.

Ein Beispiel: Vor einiger Zeit sind in unserem Projekt die Cloud-Kosten stark angestiegen. Die Teams nutzten ihre Flexibilität innerhalb der Cloud, um bei Bedarf immer mehr neue virtuelle Maschinen einzurichten. Da ihnen aber niemand transparent gemacht hatte, was die neuen Maschinen kosten und wie viele sie bereits nutzen, spielten die Kosten bei ihren Entscheidungen auch nie eine Rolle. Nach dieser Erfahrung fing man an, die Kosten automatisch zu ermitteln und allen Teams zugänglich zu machen.

Priorisierung: Üblicherweise kann ein Team nicht alle anfallenden Aufträge sofort erle-

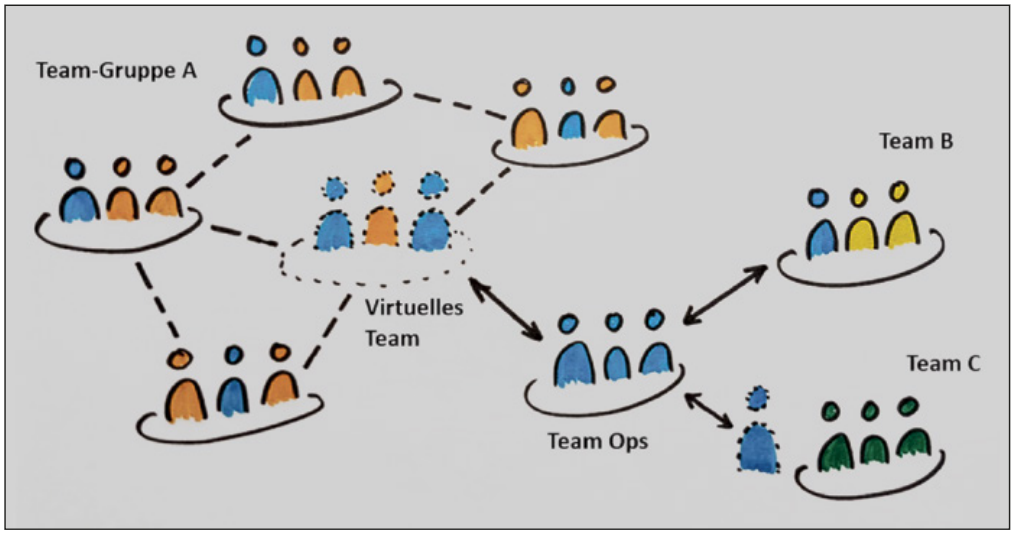


Abb. 1: Kooperationsmodelle in DevOps – Virtuelles Team (Team A), eingebettet (Team B), Liaison (Team C), nicht abgebildet sind die Communities of Practice

digen, und so entsteht ein Backlog von Tickets, das aktiv verwaltet werden sollte. Im Falle unseres Ops-Teams waren das hauptsächlich Aufträge von allen Dev-Teams des Projekts, bei dem virtuellen Team Aufträge aus den Sprints des skalierten Teilprojekts. „Aktiv verwalten" bedeutet, regelmäßig das Backlog zu ordnen und zu säubern, damit es immer den aktuellen Stand des Wissens repräsentiert. Das kann im Rahmen eines dedizierten Meetings passieren, in dem alle Betroffenen anwesend sind, oder bei Bedarf (z. B. nach dem Daily Stand-up). Während der Priorisierung sollte man auch immer die Etappenziele (siehe unten) im Blick haben und gegebenenfalls Dinge bewusst nicht tun, die nicht auf die Ziele einzahlen.

Product Owner/technischer Coach: Je nach Reifegrad des virtuellen (Nicht-Scrum-) Teams benötigt das Team auch eine Art von technischem Coach, der es begleitet, Dinge hinterfragt, Fokus und Verbesserung einfordert. In unserem Fall war dieser Coach allerdings fast so etwas wie ein Product Owner (PO), der mit dem Team zusammen plant, was als Nächstes ansteht, und im Zweifelsfall schnell entscheiden kann.

Diese „PO-Coach-Kombination" hat Vor- und Nachteile. Vorteilhaft ist, dass auch ein Quasi-PO normalerweise enger mit anderen POs beziehungsweise den Business Ownern des Projekts zusammenarbeiten kann und so mehr Geschäftswissen in das Team bringt. Nachteile können entstehen, weil mit diesem PO ein weiterer Entscheidungsträger involviert ist, der möglicherweise gerade andere Prioritäten hat als die „echten" POs. Außerdem kann ein vorher rein selbstorganisiert arbeitendes Team den zusätzlichen PO eher als Behinderung und Demotivation empfinden denn als Unterstützung. Beides kann – wie in unserem virtuellen Team im Projekt – zu Konflikten führen.

Gesamtziel (bzw. Vision/Mission): Jeder im Team sollte es kennen und den Zusammenhang zwischen dem Ziel und der täglichen Arbeit verstehen. In unserem Kontext war die Mission, dass die Teams zum Termin X ein bestimmtes Serviceniveau (d. h. Recovery-Zeit, Reaktionszeit, Ausfallzeit usw.) in der Betriebsführung für ihre Software erreichen sollten. Dieses Ziel sollte von Anfang an direkt an alle kommuniziert werden (z. B. in einem Review- oder Planungs-Meeting, bei dem alle Teams und Stakeholder anwesend sind), es sollte allgegenwertig sichtbar sein (z. B. Plakat an der Wand, Bild/Logo im Wiki usw.) und immer wieder bei der Ableitung von Teilzielen oder -aufgaben als Maßstab dienen.

Etappenziele: Abgeleitet vom Gesamtziel – aber oft beeinflusst vom Tagesgeschäft – sollten Etappenziele definiert werden, die dem Team für den Zeitraum von etwa zwei bis vier Wochen ein oder mehrere konkrete Ziele geben, in etwa vergleichbar mit einem Sprint Goal bei Scrum. Das ist wichtig, da sonst strategische Themen in einem von Reagieren und Zuliefern getriebenen Umfeld gerne unter den Teppich fallen.

In unserem Fall definierten die Team-Ops und der technische PO Etappenziele gemeinsam und brachen sie herunter in konkrete Anforderungen, und zwar regelmäßig alle zwei Wochen.

Experimentieren und kontinuierliche Verbesserung – eine Frage der Kultur

Die erfolgreiche Einführung und Anwendung von DevOps in einem Unternehmen ist in der Regel durch die bloße Nutzung bestimmter einzelner Praktiken oder Tools nicht zu erzielen. Erforderlich ist eine tief greifende und andauernde Veränderung der Kultur im Unternehmen. Diese Veränderung muss in erster Linie vom Management gewollt und unterstützt sowie von den Mitarbeitern gelebt und gepflegt werden.

In unserem Projekt wurden beispielsweise von Projektleitung und Mitarbeitern Leitlinien definiert, die den Rahmen für die Zusammenarbeit im Projekt absteckten, unter anderem durch Führung als Coaching, Selbstorganisation, Agilität usw. Doch auch Leitlinien allein reichen natürlich nicht aus, entscheidend ist ihre Integration in die tägliche Arbeit.

Ein zentrales Element der DevOps-Kultur ist die Idee des kontinuierlichen Experimentierens und Lernens. In einem komplexen Umfeld wie der Produktentwicklung gibt es keinen vorgefertigten Master-Plan, man muss immer wieder neue Wege gehen, Hypothesen im Entwicklungsprozess durch Tests validieren oder verwerfen. Das Experimentieren bezieht sich hier nicht nur auf Variationen des Produkts, sondern natürlich auch auf die Zusammenarbeit, Prozesse und die Organisationsstruktur.

Mit der Zeit wurden mithilfe von Experimenten in unserem Projekt organisatorische Hürden abgebaut. Gab es beispielsweise zu Beginn noch einen wöchentlichen Statusbericht jedes Teilprojekts, wurde dieser mit der Zeit umstrukturiert und auf zwei Wochen und später auf vier Wochen erweitert, da mit der Zeit eine Vertrauensbasis entstanden war. Im Ops-Team wurde die Selbstorganisation Stück für Stück erweitert, indem die Teammitglieder immer mehr eigene Entscheidungsbefugnisse bekommen hatten. Jede Erweiterung wurde dabei erst für begrenzte Zeit ausprobiert und dann in den Retrospektiven bewertet.

Damit die Teams Experimente machen und damit auch Risiken eingehen können, brauchen sie einen sicheren Rahmen (vgl. Safe Work Environment [Kim16]), in dem zum Beispiel auch fehlgeschlagene Experimente oder gar Fehler als (Lern-) Erfolg wertgeschätzt werden. Weniger abstrakt: In unserem Projekt geschah es zum Beispiel, dass Mitarbeiter versehentlich Daten gelöscht hatten. Einigen Teams entstanden dadurch Schäden. Trotzdem wurde nicht die Frage gestellt, wer schuld sei, oder gar die Mitarbeiter sanktioniert. Stattdessen hat man systematisch gemeinsam analysiert, wie es dazu kommen konnte, und Maßnahmen für eine zukünftige Vermeidung entwickelt („Post Mortem"-Analyse).

Für eine gute Zusammenarbeit aller Einheiten müssen auch die organisatorischen Rahmenbedingungen stimmen: Es braucht gemeinsame Verantwortung und Ziele, damit die Teams anfangen können, zusammen und nicht mehr isoliert voneinander zu arbeiten. In unserem Kontext resultierten zuerst viele Probleme aus der ungünstigen Verteilung der Verantwortung für die Entwicklung und Betriebsführung. Erst eine Management-Entscheidung und mehr Verständnis von DevOps brachten hier den Durchbruch.

Fazit: Was haben wir erreicht und wie geht es weiter?

In einer schwierigen Situation im Projekt halfen uns DevOps-Prinzipien und -Praktiken dabei, entscheidende Verbesserungen zu erzielen. Zusammen mit dem Management konnten wir ungünstige Verantwortungskonstellationen zwischen Dev und Ops aufheben und uns auf gemeinsame Ziele (das Produkt zum Kunden bringen) fokussieren.

Im Rahmen eines kontinuierlichen Verbesserungsprozesses wurden verschiedene Teamkonstellationen für die Integration von Dev und Ops ausprobiert. Das erhöhte das Bewusstsein für Betriebsführungsthemen in den Dev-Teams (und umgekehrt) und verbesserte auch die Stimmung und die gegenseitige Wertschätzung der Teams weiter. Wir unterstützten selbstorganisierte (Nicht-Scrum-) Teams dabei, konkrete Ziele beziehungsweise Etappenziele zu definieren und ihre Entwicklung fokussierter zu steuern. Den Teams fällt es heute leichter, strategische Themen

Literatur & Links

[Kim16] G. Kim, J. Humble, P. Debois, J. Willis, The DevOps Handbook: How to create World-Class Agility, Reliability, & Security in Technology Organizations, IT Revolution Press, 2016

[Pup17] Puppet & DORA, State of DevOps Report 2017, siehe: **https://puppet.com/resources/whitepaper/state-of-devops-report**

[Suth17] J. Sutherland, K. Schwaber, The Scrum Guide™, scrum.org, 2017, siehe: **http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf**

schneller weiterzuentwickeln, weniger relevante Dinge erst gar nicht anzufangen und Arbeiten an hinreichend ausgebauten Komponenten früher zu beenden.

Durch einen kontinuierlichen Verbesserungsprozess und das Coaching mit den Teamleitern/POs haben wir die Grundlagen für eine Lern- und Experimentierkultur in den Teams geschaffen. Wir können heute besser mit Fehlern umgehen und das Experimentieren hilft uns dabei, schneller Entscheidungen zu treffen und Veränderung herbeizuführen.

Damit zukünftig die Zusammenarbeit noch besser werden kann, muss im Folgenden unter anderem die Systemarchitektur entflechtet werden. Das würde die individuelle Rolle der POs stärken und somit den Entscheidungsprozess beschleunigen. Ebenso müssten weniger Abhängigkeiten gepflegt werden. ||

Die Autoren

Steffen Brandt
(steffen.brandt@andrena.de)
begann seine berufliche Laufbahn als Agile Software Engineer bei der andrena objects ag in Frankfurt. Dort ist er mittlerweile als Agile Coach tätig und unterstützt Unternehmen dabei, Scrum einzuführen und Prozesse sowie Vorgehensweisen der agilen Softwareentwicklung zu etablieren.

Christoph Jung
(christoph.jung@andrena.de)
arbeitet seit mehreren Jahren als Agile Coach/Scrum Master bei der andrena objects ag, wo er Kunden hilft, agile Vorgehensweisen in ihrem Kontext umzusetzen und davon zu profitieren. Er hat außerdem jahrelange Erfahrung im Bereich Forschung & Entwicklung und Projektmanagement.