Sebastian Müller
(smueller@autoscout24.com)
ist Softwareentwickler und interessiert sich besonders für aktuelle Technologien und effiziente Zusammenarbeit in Unternehmen. Er ist ein großer Anhänger agiler Entwicklungsmethoden und ist Evangelist für die DevOps-Kultur. Nach Projekten für die Stiftung Warentest, Porsche und mehreren Verlagen ist er aktuell für AutoScout24 in München beschäftigt.

DevOps verändern das Rollenverständnis der kompletten IT

Wie die DevOps-Kultur die Arbeitsweise in IT-Abteilungen grundlegend verändert am Beispiel von AutoScout24.

Die DevOp-Kultur ist die größte Änderung der IT-Landschaft seit dem Aufkommen agiler Softwareentwicklungsmethoden. Dieser kulturelle Wandel betrifft ganzheitlich die gesamte IT und nicht nur die Softwareentwicklung. Das Selbstverständnis vieler Rollen in der IT ändert sich drastisch. AutoScout24 hat den kulturellen Wandel bereits vollzogen. Dabei gab es viele Hürden zu überwinden und umso mehr Erfolge zu feiern.

Vermutlich hat jeder in einem Unternehmen dieses Szenario schon einmal erlebt: Es tritt ein Fehler in der Produktivumgebung auf, der Dienst geht offline, Endkunden beschweren sich, Manager fragen, was da los ist, alle verfallen in Panik.

Noch während der Dienst offline ist und an einer Lösung gearbeitet wird, beginnt die Schuldzuweisung. Sätze wie „Diese ‚Programmieraffen' haben schon wieder fehlerhafte Software geliefert" oder „Diese Admins sollten weniger World of Warcraft zocken, dafür ab und zu mal ihr Monitoring im Auge behalten" schwirren durch die Luft.

Natürlich sind immer die anderen schuld, wenn etwas nicht funktioniert: „Die haben ihren Job einfach nicht richtig gemacht". Die Softwareentwicklung wirft den Administratoren vor, sie hätten zu spät reagiert oder hätten den Fehler schon

DevOp als Rolle vs. DevOps als Kultur

Der Begriff DevOps ist derzeit in aller Munde. Nur leider ist er wie viele Buzz-Words nicht genau definiert. Es haben sich zwei unterschiedliche Auffassungen von DevOps herauskristallisiert: DevOps als Rolle und DevOps als Kultur:

Diejenigen die **DevOps als Rolle** beschreiben, sind der Meinung, dass es dedizierte DevOps geben muss, die am besten in einem eigenen Team zusammenarbeiten. Dort kümmern sie sich um Tooling, Monitoring, Anwendungen einzurichten usw. Sprich die Schnittmenge von Betriebsabteilung und Softwareentwicklung wird in ein eigenes Team extrahiert.

Als **DevOps als Kultur** beschreibt man den Ansatz, dass beide Abteilungen näher zusammenrücken, ihr Wissen und Tools teilen. Eben zusammen an einem Strang ziehen und sich zusammen als Einheit sehen.

AutoScout24 geht einen Weg dazwischen. Es gibt dedizierte DevOps, die aus Softwareentwicklung oder Betrieb kommen und die Kultur etablieren, so dass schlussendlich das Ziel DevOps als Kultur erreicht werden kann.

Tab.1: DevOp als Rolle vs. DevOps als Kultur

auf den Testumgebungen sehen müssen oder haben die Anwendung falsch konfiguriert oder, oder, oder...

Die Produktionsabteilung bringt natürlich das Argument hervor, dass die Software fehlerhaft sei, besser hätte getestet werden müssen und überhaupt hätte man sie viel früher darüber informieren sollen, dass so ein großer Change mit diesem Release live geht.

Bei den ersten Malen beziehungsweise in kleinen Firmen geht man abends zusammen ein Bier trinken und das Problem ist schnell vergessen. Treten solche Szenarien öfter auf oder die Beteiligten haben keine soziale Beziehung zueinander, kennen sich vielleicht nicht mal mit Namen, so ist die Eskalation schon vorprogrammiert.

Besonders gespannt wird die Lage, wenn nun das mittlere oder obere Management nach einem Verantwortlichen sucht und Konsequenzen verlangt. Leider haben wir vermutlich alle zumindest im Bekanntenkreis jemanden der so eine Geschichte erzählen kann.

Richtige und falsche Konsequenzen

Natürlich ist es der schlimmste Fall, wenn ein Dienst vom Netz geht. Da geht eventuell in kurzer Zeit viel Geld verloren. Wenn das passiert, müssen Maßnahmen getroffen werden, um solch einen Vorfall in Zukunft zu vermeiden. Aber was sind die richtigen Maßnahmen?

Schnell werden Rufe nach mehr Prozess laut. Sollte man die Testphase vor einem Release verlängern? Sollte eine Phase unmittelbar um ein Release eingerichtet werden, bei der sich alle zu 100% darauf konzentrieren, die Produktentwicklung für diese Zeit einzustellen?

Vielleicht helfen Formulare, auf denen penibel jede Änderung aufgelistet wird. Am besten unterschrieben von jedem Betroffenen und zusätzlich dem Management, damit sich jeder darüber bewusst ist, dass er verantwortlich ist, falls die Plattform offline geht. Würde es helfen einen Release-Verantwortlichen einzuführen? Ich frage mich, ob solche Maßnahmen jemals bei einem Unternehmen das Problem gelöst haben.

Man sollte erst einmal die Frage stellen, was eigentlich die Ursache des Problems ist. Fehlt den Mitarbeitern wirklich die Aufmerksamkeit auf ein Release oder das Bewusstsein wie wichtig es ist? Kann man wirklich alle Fehler mit noch mehr Testen finden? Lösen Formulare das Problem eines fehlenden Kommunikationsflusses? Und ist es sinnvoll eine einzelne Person für das Release verantwortlich zu machen?

Sicher ist jedenfalls: Ein besserer Informationsfluss zwischen Development und Applikationsbetrieb kann helfen! Vor allem, wenn es sich um einen Dialog handelt, der ganz natürlich (sprich: keine Info-Meetings) verläuft und ständig stattfindet.

Aber einen Dialog zwischen zwei Abteilungen zu etablieren, die komplett andere Arbeitsweisen haben, stellt sich als schwieriges Unterfangen dar. Manchmal weiß die eine Partei gar nicht, was die andere macht – oder noch schlimmer: wie die entsprechenden Personen „bei denen“ überhaupt heißen.

Worum kümmern die sich eigentlich den ganzen Tag und was sind deren Probleme? Macht Sie das nicht auch stutzig? Liegt vielleicht genau hier die Ursache des Problems? Es ist doch nicht möglich, effizient zusammen an einer Problemlösung zu arbeiten, wenn man seinen Partner gar nicht kennt (siehe Abbildung 1).

Abb. 1: DevOps vs. ‚Klassische Zusammenarbeit‘

Information führt zu Vertrauen, Vertrauen führt zu gegenseitiger Unterstützung

Wie Henry Ford schon sagte: „Zusammenkommen ist ein Beginn, zusammenbleiben ist ein Fortschritt, zusammenarbeiten ist ein Erfolg“. Menschen müssen zusammenarbeiten, um gemeinsam erfolgreich zu sein. Dazu ist es zunächst notwendig, zusammenzukommen, um sein Gegenüber kennenzulernen. Man muss verstehen, was der andere macht, um ihn zu verstehen. Es bedarf einer gemeinsamen Sprache, um ohne Missverständnisse kommunizieren zu können und man muss die Probleme des anderen verstehen, um helfen zu können.

Im Falle von AutoScout24 haben wir damit begonnen, Softwareentwicklern das Basiswissen der Web-Administratoren beizubringen. In zweiwöchigen Bootcamps wurden Entwickler zum Mini-Admin gemacht.

Ein Bootcamp besteht aus zwei Teilen: Einem strukturierten theoretischen Teil, in dem die wichtigsten Betriebstools und das nötige Infrastrukturwissen vermittelt wird. Im zweiten Teil übernimmt der Entwickler für eine Zeit lang alle alltäglichen Aufgaben eines Administrators und erledigt diese zusammen, also im Pair wie man es aus der Softwareentwicklung kennt, mit dem Admin. Dieser fungiert in dieser Zeit primär als Tutor, zeigt was zu tun ist, erklärt die Hintergründe und beantwortet die Fragen.

Natürlich kann man in so kurzer Zeit nur ein begrenztes Wissen vermitteln. Deshalb reicht es, sich auf wichtige Dinge wie Infrastruktur, Deployment Tools, Maintainance-Prozesse oder Ähnliches zu beschränken. Nicht unbedingt Relevantes, z. B. wie der Mail Server jetzt im Detail arbeitet, kann auch später noch erfragt werden. Dieser zweite Teil des Bootcamps ist sehr individuell und agil auf die Vorkenntnisse des Entwicklers abgestimmt.

Als besonders wertvoll hat sich herausgestellt, dass während des Bootcamps der normale Alltag eines Admins durchlebt wird. Auf diese Weise wird keine konstruierte Situation erzeugt, sondern man lernt die echten Prozesse kennen. Vor allem lernt man ganz automatisch, wen man für welches Problem am besten ansprechen

kann. So entstehen kurze Dienstwege und manche Probleme lassen sich plötzlich mit einem kurzen Anruf erledigen.

Das grundlegende Verständnis der Infrastruktur und das Wissen warum etwas so ist wie es ist, hat eine sofortige Auswirkung auf die Softwareentwicklung. Bereits bei der Planung von neuen Funktionen können Aussagen darüber getroffen werden, wie die Hardware dimensioniert sein muss. Die Softwareentwickler können selbst abschätzen, welche Firewall-Regeln, DNS-Einträge oder ähnliches angelegt werden müssen. Im Optimalfall haben sie sogar das Recht, dies eigenständig zu tun.

Ja, Sie haben richtig gelesen: Softwareentwickler sind durchaus in der Lage Firewall-/Routing-Regeln oder DNS-Einträge selbst einzurichten. Wenn sie Zusammenhänge und Hintergründe verstanden haben, spricht nichts dagegen, dies direkt von den Entwicklungsteams machen zu lassen.

Es ist natürlich empfehlenswert, dies im Pair mit den Spezialisten zu machen oder zumindest von ihnen ein Review einzuholen. Automatisches Deployment der Konfigurationsänderungen über die Testumgebungen mit Configuration as Code wäre natürlich ein erstrebenswertes Sahnehäubchen.

„Die interessieren sich ja wirklich für unsere Probleme“

Mit der DevOps-Kultur sind viele Dinge möglich, von denen Sie heute noch überzeugt sind, sie seien unmöglich. Der Schlüssel ist fast immer: Vertrauen! Wenn Vorurteile fallen, wächst das Vertrauen! Gemeinsames Arbeiten erzeugt Verständnis für die Belange des anderen. Oftmals haben Mensch aus anderen Bereichen ganz simple Lösungen für die Probleme, die einem selbst unlösbar erscheinen. Auf diese Weise lassen sich schnell Erfolge feiern.

Nach den ersten erfolgreichen Bootcamps für Entwickler beim Anwendungsbetrieb wurde schnell Interesse nach einem Austausch in die andere Richtung artikuliert. Allerdings stellte sich gleichzeitig die Frage, wie so ein Austausch funktionieren könnte. Schließlich ist die Einarbeitungszeit deutlich höher, jemand mit nahezu keiner praktischen Programmierausbildung in den produktiven Entwicklungsprozess einzugliedern.

Wer nicht wagt, der nicht gewinnt. Also haben wir es einfach ausprobiert – anstatt lange zu diskutieren. Kollegen aus dem Anwendungsbetrieb wechselten ebenfalls für zwei Wochen in Entwicklungsabteilungen. Im Pair-Programming wurden die Kollegen ins kalte Wasser geworfen und siehe da: sie schwammen!

Die nötigsten Kenntnisse waren schnell angeeignet und wurden produktiv genutzt. Gleichzeitig konnten die Administratoren die gelernten Programmier-Best-Practices für Automatisierungs-Scripts in ihrem Alltag nutzen (siehe Tabelle 2).

Sehr wertvoll ist das Wissen von Betriebsmitarbeitern bei der Optimierung von Entwicklungsprozessen. Zum einen zerstört das Wissen, wie viel Aufwand in Softwaretests gesteckt wird, das Vorurteil, die Software wäre nicht ausreichend getestet, wenn sie produktiv geschaltet wird. Zum anderen hilft das Expertenwissen bei der Optimierung.

Zwei Beispiele:

- Eine geschickte Konfiguration von Routing-Regeln, neuen Test Agents mit optimalem Hardware Sizing sparte uns ein Drittel an Zeit bei den automatisierten Frontend-Tests.
- Durch gemeinsame Überarbeitung des Release-Prozesses und Austausch des Deployment Tools konnte die Zeit von über einer Stunde auf acht Minuten verkürzt werden.

Ein Wissensaustausch in beide Richtungen ist auf jeden Fall einen Versuch wert. Auch wenn auf den ersten Blick das Potenzial gar nicht so groß erscheint.

Vorteile des Wandels

- Entlastung von widerkehrenden Aufgaben
- Schnellere Produktentwicklung
- Kurze Dienstwege
- Große Hilfsbereitschaft füreinander
- Gemeinsame Sprache
- Produktverantwortung der Entwicklungsteams bis in die Produktivumgebung
- Keine Schuldzuweisungen und besseres Arbeitsklima

Tab.2: Vorteile des Wandels

Tools, Tools, Tools!

Wie gerade schon angeklungen spielen Tools bei der Einführung von DevOps eine wichtige Rolle. Tools verhindern Fehler, automatisieren Dinge und abstrahieren Komplexität so, dass kein Experte mehr für den Zweck des Tools notwendig ist (siehe Tabelle 3).

Beispielsweise ist es für Entwicklerteams oft notwendig, zu überprüfen, ob in der Produktivdatenbank Werte korrekt gesetzt sind. Dies ist ein typischer Fall für ein Tool, da es utopisch wäre, jedes mal einen Datenbankadministrator zu bitten, die Überprüfung zu machen – genauso wie es übertrieben wäre, jedem Entwickler Zugriff auf die Produktivdatenbank zu gewähren.

Es bietet sich also an, ein Tool einzuführen, das den entsprechenden Anforderungen gerecht wird, aber gleichzeitig so simpel und sicher ist, um jedem Mitarbeiter Zugriff darauf zu gewähren. Die Softwareentwicklung kann sich dann schnell und einfach selbst helfen und die Datenbankadministratoren haben mehr Zeit, um sich mit größeren Projekten zu beschäftigen.

Es lohnt sich auch der Blick auf die Tools, die bereits existieren und nur von einer der beiden Parteien genutzt werden. Monitoring Tools, die üblicherweise nur

Wichtige Tools, die geteilt werden sollten

- Monitoring
- Tool zum Zugriff auf Logfiles
- Tool für Datenbankauswertungen
- Automatische Ausführung von DDL-Scripten (Data Definition Language)
- Tool zum Lesen und (wenn sinnvoll) Schreiben von Firewall-/Routing Regeln
- Web-Analytics
- SEO-Tools
- Tools zum automatischen initiieren von Applikationen

Tab. 3: Wichtige Tools, die geteilt werden sollten

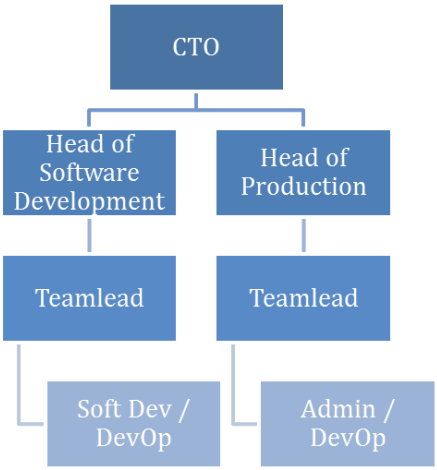

Abb. 2: Hierarchiestruktur

von der Betriebsabteilung genutzt werden, bieten sich an, mit den Produktteams geteilt zu werden.

Produktteams sollten ein Interesse daran haben, wie sich ihre Software in der echten Welt verhält. Das können sie nur wissen, wenn sie auch die Möglichkeit haben, dies zu monitoren. Außerdem können nur die Entwickler der Software genau sagen, wie man feststellt, ob eine Anwendung sauber läuft. Schon aus diesem Grund sollten Monitoring Tools geteilt werden.

Bei AutoScout24 haben wir die Demokratisierung der Tools so simpel wie möglich gemacht: Alle haben die notwendigen Lese- und wenn nötig Schreibrechte bekommen. Der Wissenstransfer wurde durch eine initiale Schulung angestoßen, aber dann vor allem durch gemeinsame Nutzung (also wieder im Pair) mit den Spezialisten gefestigt und verbreitert.

Am Anfang kamen die Aufgaben über den gewohnten Weg à la: „Kannst du bitte mal schnell…?“. Doch dann hieß die Antwort meistens: „Lass uns das doch einfach gemeinsam machen“. Diese Methodik führt zu einer sehr effizienten Wissensverteilung und nimmt vor allem die Berührungsängste mit den neuen Tools, da man die Schritte schon mehrfach zusammen mit den Spezialisten gemacht hat.

Neues Potenzial und wie man es nutzt

Da die Softwareentwicklung mit dem Betrieb nun effizient zusammenarbeitet, eröffnet sich ein ganz neues Potenzial, das genutzt werden möchte. Werden gemeinsame Visionen und Ziele entwickelt, rücken unerreichbare Dinge plötzlich in greifbarer Nähe. Schlagworte wie 100% Erreichbarkeit oder Continous Deployment erscheinen leicht zu realisieren, wenn alle an einem Strang ziehen und das gleiche Ziel vor Augen haben (siehe Abbildung 2).

Spätestens hier ist aber wieder das Management gefragt. Gemeinsame Visionen und Ziele kann es nur geben, wenn auch im Management eng zusammengearbeitet wird. Eine Hierarchiestruktur wie in Abbildung 2, macht es schwer, eine gemeinsame Strategie zu finden.

Das Management muss verstehen, dass die Funktionsorientierung im krassen Kontrast zur DevOps-Kultur steht. Dies bedeutet nicht, dass man sofort den großen Schritt zur Kundenorientierung durchführen muss. Vielleicht lassen sich viele Potenziale schon mit einer Matrix-ähnlichen Ausrichtung erreichen. Das ist aber abhängig vom jeweiligen Unternehmen und muss individuell entschieden werden.

Status quo bei AutoScout24

Bei AutoScout24 haben bereits einige Iterationen mit DevOps durchlebt. Das Bootcamp wurde auf wenige Tage optimiert, dafür gibt es regelmäßige Infoveranstaltungen zur Vertiefung oder Auffrischung des Wissens. Jedes Produktteam hat mindestens einen DevOp.

Dieser DevOp hat das Bootcamp durchlaufen, hat also das nötige Wissen und verbreitet es weiter. Er ist erster Ansprechpartner für alle anfallenden Aufgaben, die den Betrieb oder die Einrichtung der Applikationen des Teams betreffen. Die restliche Zeit geht er seinem ursprünglichen Aufgabenprofil nach oder optimiert die Deployment-/Betriebsprozesse.

Die Auswahl an Tools ist inzwischen ziemlich fix und wird immer weiter verbessert. Die Produktteams sind nun auch verantwortlich für den Produktivbetrieb der Anwendung. Das bedeutet, in jedem Team gibt es mindestens eine Person, die für Rufbereitschaft zur Verfügung steht.

Wir haben uns nun vorgenommen endgültig den Schritt von DevOp als einer Rolle zur DevOps als Kultur zu gehen, indem wir jedem Softwareentwickler das nötige Wissen vermitteln möchten. Auch bezüglich von Rechten wird es dann keine Unterscheidung mehr geben. Die bisherigen DevOps werden dann als ‚Evangelisten‘ fungieren und die Weiterentwicklung der Kultur vorantreiben.

Das Ergebnis ist mehr als die Summe der Einzelteile

Die Softwareentwicklung erlebt durch die Einführung der DevOps-Kultur einen stetigen Wandel. Krass ausgedrückt startet man meistens bei einem Zustand des „Software über die Mauer hin zum Applikationsbetrieb werfen“ und durchlebt einen Wandel zu „Gemeinsame Produktverantwortung von der Planung bis in den Betrieb“ (vgl. Tabelle 2).

Ich will nicht verschweigen, dass jeder Wandel, gerade wenn er kultureller Natur ist, auf Ängste und Widerstände stößt. Es ist manchmal anstrengend, immer wieder Kollegen von den Vorteilen zu überzeugen. Gerade wenn die Auswirkungen so groß sind, empfiehlt es sich, kleine Schritte zu gehen und Erfolge sichtbar zu machen.

Oft höre ich Ängste von Unternehmen, die DevOps einführen, Ängste, ihr Job würde dann nicht mehr benötigt. Spricht man mit den gleichen Menschen einige Wochen später, sind sie euphorisch, weil sie endlich Zeit haben, um sich großen, wichtigen Projekten zu widmen und die nervigen wiederkehrenden Aufgaben endlich weg sind. Bringt man Softwareentwickler und Admins dazu, effizient zusammenzuarbeiten, ist das Ergebnis ganz sicher größer als Beide für sich genommen. ■