Vom IT-Betrieb zu Platform Engineering. Ein Reisebericht (3/3)

Vor einer Woche habe ich berichtet, wie wir Basistechnologien, SelfServices und Automaten etabliert haben. Das hat uns dabei geholfen, wiederkehrende Aufgaben zu automatisieren, die Umsetzungsqualität zu erhöhen, die Entwicklungsteams beim Aufbau und Betrieb von neuen Services zu beschleunigen und die Aufwände im IT-Betrieb zu reduzieren. Aber das alles war nur Werkzeug. Auf unserem Weg zum Platform Engineering Team war es auch wichtig, unser Mindset anzupassen. Wir mussten verstehen, wie unsere wichtigsten Kunden, die Entwicklungsteams, denken und was sie brauchen. Auch unsere Prozesse mussten weiter optimiert werden. Es gab also noch mehr zu tun, als nur ein paar Tools zu etablieren. Auch wenn dieser Reisebericht so geschrieben ist, als ob der Mindset-Change als Letztes stattgefunden hätte, so ist das natürlich nicht richtig. Die technologischen und die kulturellen Änderungen im Team fanden mehr oder weniger zeitgleich statt.

Kapitel 3: Mindset, Methoden, Prozesse und

mehr...

PENG! Technik ist nicht alles!

Wenn wir schneller werden wollen, dann müssen wir einiges mit SelfService Schnittstellen machen. Cool, verstanden, fertig? Nope, never, auf keinen Fall! Was ist eigentlich mit Mindset, Kultur, Methoden,...? Auf den Konferenzen reden sie immer von DevOps, Scrum, Agile, Kanban .... Ah, Kanban, da machen wir doch schon was. Und unsere Devs und Ops mögen sich doch auch schon und gehen zusammen Bier trinken. Dennoch fehlt da noch was. Wir müssen uns also überlegen, wie die künftigen Produkt-Teams und der IT-Betrieb zusammenarbeiten sollten. OK, noch mehr Ziele für den Umbau. Wir müssen nicht-technische Themen wie z.B. **Kultur, Zusammenarbeit, Mindset, Prozesse, Zuständigkeiten und die Schnittstellen zwischen den Teams definieren**. Da wartet einiges an Arbeit auf uns. Um die Veränderung unseres Teams nach innen und nach außen zu verdeutlichen, wollten wir uns auch einen neuen Namen geben. Nach einigen Diskussionen war uns klar: aus „IT-Betrieb“ sollte „Platform Engineering“ oder kurz „PENG“ werden.

Im Rahmen des Aufbaus der Produkt-Organisation wurden wir als Platform Engineering Team sehr früh mit einbezogen. Gute Idee! Das hat uns die Chance gegeben, neben den ganzen Team-, Technik-, Kultur- und Prozessumbauten auch die notwendigen Operations Umbaumaßnahmen mit einzubringen. Warum ist das so wichtig dieses früh und offiziell zu machen? Ich habe drei technisch Beispiele beschrieben, wo dringend Änderungen notwendig sind und SelfServices etabliert werden müssen, um schneller zu werden. Das macht man nicht mal eben so nebenbei. Notwendig dafür sind größere Investitionen und Anschaffungen in die Infrastruktur. Es muss bewertet werden, ob man SelfServices selber erstellen will oder irgendwo einkauft (Kosten/Nutzen). Zum Selberbauen brauchen wir mehr Personal, welches temporär extern beschafft und bezahlt werden muss. Auch die Themen Kultur, Zusammenarbeit, Mindset, Prozesse, Zuständigkeiten und Schnittstellendefinition zwischen Teams brauchen einiges an Zeit und sollten möglichst zusammen mit den Umbauten hin zu Produkt-Teams erfolgen.

OK, Bestandsaufnahme: Wir wissen wir sind gut, aber nicht schnell genug -> Handlungsbedarf. Wir haben uns technische und nicht technische Ziele gesetzt. Die Transformation von IT-Betrieb zu Platform Engineering kann beginnen!

Um die Transformation zu unterstützen, haben wir im Büro eine „Transformation Wall“ erstellt mit den Zielen, Regeln, Infos und was sonst mit der Transformation zu tun hat. Diese Wand lebt, hat immer die aktuellsten Themen wie z.B. die Maßnahmen einer Retro. Jeder, der ein Thema hat, darf dieses auch auf die Wand pinnen, so dass wir bei der nächsten Gelegenheit darüber reden können. Kurz gesagt: Die Wand **begleitet unsere Transformation und erinnert uns auch jeden Tag daran, was wir tun wollen und warum.**

Setting the Stage: Agile Knowledge (the Basics)

Alle redeten über DevOps, Agile, SCRUM, Kanban, „You build it, you run it“ und so weiter …. Doch was ist das eigentlich alles? Es gab reichlich Unwissenheit oder – schlimmer – gefährliches Halbwissen. Lasst uns also eine gemeinsame Wissensbasis schaffen, um all die umherschwirrenden Buzzwords zuordnen und verstehen zu können. Auf dieser Basis können wir dann alle anderen Themen aufbauen.

Vor einigen Monaten hatte ich das Glück, einen Zwei-Tages „Agile Mindset“-Workshop zu besuchen, der Grundlagen zu SCRUM, Kanban, Agilität, agiles Manifest, agile Werte usw. vermittelt hat. Während des Workshops, der angereichert war mit kleinen Übungen, habe ich mir die ganze Zeit überlegt, wie ich das ins Team transportiert bekomme. Am Ende des ersten Tages war die Antwort klar: Gar nicht. Zusammen mit dem Agile Coach, der u.a. einen Operations Background hatte, haben wir uns dann überlegt, wie wir diesen Workshop für unser IT-Betrieb-Team abhalten können. Nach kurzer Zeit stand die Planung, und wir haben zwei echt tolle und spannende Workshop-Tage gehabt. Im Anschluss konnten wir viele „Aha Momente“ verzeichnen und hatten als IT-Betrieb ein recht gutes Basis-Verständnis zu den agilen Methoden, der Idee dahinter, den Unterschieden und auch den Vor- und Nachteilen. Klar waren wir weit ab, agile Spezialisten zu sein, aber wir hatten einen super Werkzeugkasten erhalten, der uns auf den weiteren Wegen sehr geholfen hat.

Ausprobieren: Funktioniert SCRUM für Operations?

So, Workshop fertig, jetzt sind wir agile! Nicht wirklich. Uns war klar, wir durften gerade einmal am Inhaltsverzeichnis schnuppern. Die wirkliche Arbeit wartete nun noch auf uns. Wir als Thalia verkaufen ja jede Menge Bücher, aber ein Buch „Agile Operations @ Thalia“ hatten selbst wir als Buch-Spezialist nicht. Vielleicht kommt das später noch Mit anderen Worten: Wir haben eine Idee bekommen, welche Werkzeuge es gibt. Welche Werkzeuge uns bei unserer Arbeit wirklich helfen (und nicht einfach nur Hip sind), mussten wir in der Praxis selber ausprobieren.

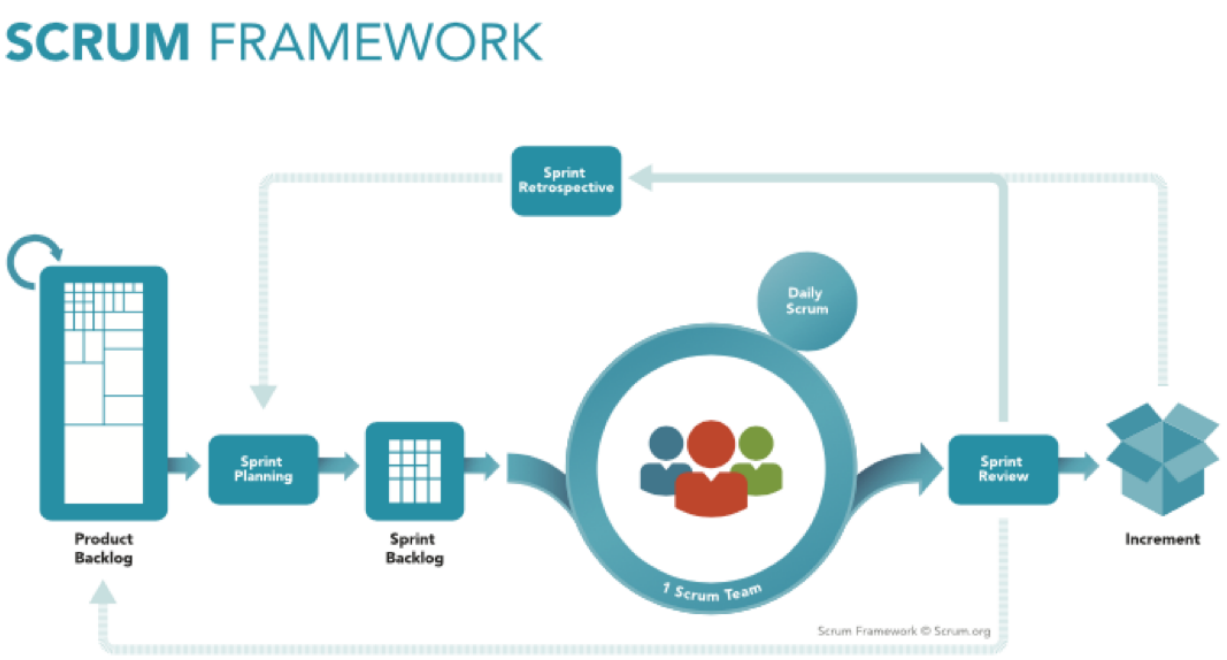
Wir hatten eine Menge über SCRUM, Kanban und Co. erfahren. Kanban hatten wir bereits einige Zeit (zumindest im Ansatz) im Tagesgeschäft praktiziert. Nun wollten wir SCRUM etwas näher kennenlernen. Zusammen mit einem SCRUM Master haben wir unser Projekt zur Einführung der Automatischen Service Bereitstellung (ASB) kurzerhand von einem klassischen Wasserfall-Projekt nach SCRUM umgestellt. Dazu haben wir die notwendigen Meetings aufgesetzt, ein Backlog angelegt und gepflegt, Rollen besetzt, Sprints geplant, durchgeführt, reviewed, verbessert usw… Gerade am Anfang haben wir uns sehr schwer getan. Ganz besonders das Review als auch das Schneiden, Schätzen und Planen von Tickets brauchten einige Übung. Wir haben viel ausprobiert, was gehen könnte. Schätzen wir z.B. Aufwände in Personentagen, Story Points, Anzahl Tickets, …? Wir machten unsere Erfahrungen und fanden raus, was gut funktioniert, aber auch, was wir besser nicht machen sollten. Und so wurde es von Sprint zu Sprint leichter und nutzbringender. Das kleine Test-SCRUM-Team berichtete von erhöhter Transparenz, klarerer Struktur und ruhigerem Arbeiten im Sprint. Der Produkt Owner hatte einen klareren Blick auf was gemacht wurde, was wann kommen kann und hatte die Möglichkeit zu entscheiden, welches Feature er wann haben wollte. Unterm Strich war es anfangs sehr ungewohnt, jedoch sehr spannend und hilfreich. Wir haben es also geschafft, ein Operations-Projekt nach SCRUM zu führen. Die Erkenntnis stimmte uns positiv

Ein Problem gab es jedoch noch: Unser SCRUM-Testballon wurde in einer reinen Projektumgebung ohne Operations-Tagesgeschäft durchgeführt. Hier gab es keine größeren Störungen, keine spontanen und dringende Anforderungen. Im Operations Tagesgeschäft wimmelt es nur so von unerwarteten Änderungen, worauf wir teilweise hochflexibel reagieren müssen. Leider bekomme ich kein Verständnis, wenn wir die Produktions-Störung des Webshops erst im nächsten Sprint in einer Woche bearbeiten. OK, kann ich verstehen, passt aber nicht so super zu SCRUM. Für das Operations-Tagesgeschäft funktioniert 100% SCRUM für uns also nicht so gut.

Nicht Hip, aber hilfreich: An SCRUM orientieren ohne SCRUM zu machen

Warum sind wir nicht einfach bei Kanban geblieben? Das passt doch viel besser zu so unerwarteten Themen. Nun ja, wir fanden einige Elemente von SCRUM sehr spannend und hilfreich. So ist es hilfreich, dass wir uns alle zwei Wochen verbindlich zusammensetzen, um Aufgaben zu planen. Die Backlogfunktion im JIRA SCRUM-Board finden wir super, eine Retro sowie ein Review hilft uns besser zu werden, auch das Messen des Erreichten ist für uns und unsere Planung hilfreich. Das ist ein Auszug, warum wir uns aktuell an SCRUM orientieren. Klar kann es sein, dass es irgendwann Gründe gibt, wieder Kanban zu machen. Im Moment sind wir jedoch glücklich und vor allem sehr transparent. Wir haben auch gelernt, dass wir nun die gleiche Sprache sprechen wie die Produkt-Teams. Folgende Unterhaltung soll es verdeutlichen: [Dev] „Kannst du bitte folgende Ops Aufgabe diese Woche noch für mich machen? Ich erreiche sonst unser Sprintziel nicht“ – [Ops] „Das kommt etwas überraschend, um dir zu helfen müsste ich unseren Ops Sprint verändern und eine geplante Aufgabe entfernen um deine Aufgabe zu erledigen“ – [Dev] „Oh, das ist ja nicht so gut. Nein, dann plane meine Ops Aufgabe bitte in den nächsten Sprint ein. Ich kann im Zweifel warten“. Hurra, ohne das Wort „Nein“ haben sich Ops und Dev auf eine Verschiebung einer Aufgabe geeinigt und sind dabei auch noch zufrieden.

Wir wollten also SCRUM nicht nur für Projekte nutzen, sondern auch für unser Tagesgeschäft. Wie haben wir das gemacht? Erneut mit viel Ausprobieren, Überprüfen, Lernen, Bessermachen und wieder Ausprobieren. Nach einiger Zeit haben wir uns auf folgendes Setup geeinigt:



This image is the copyrighted material of Scrum.org. The original can be found here: https://www.scrum.org/resources/scrum-framework-poster

Wir haben ein **Product Backlog**, dieses ist jedoch nur rudimentär priorisiert. Bislang kommen wir mit den Themen im Sprint gut aus :-). Alle zwei Wochen führen wir ein **Sprint Planning** durch. Wir planen dabei etwas weniger als 50% der Tickets ein, die wir schaffen können. Wir orientieren uns dabei an der Anzahl der Tickets. Wir orientieren uns nicht an genaueren Aufwänden oder Story Points (siehe auch „#noEstimation-Bewegung“ unter „Messen“). Zum einen messen wir, dass die Anzahl an Tickets im Schnitt erstaunlich konstant ist. Zum anderen würden wir zwar mit Story Points o.ä. genauer werden in der Planung, jedoch würde sich unser Planungsaufwand deutlich erhöhen. Für uns ein ungünstiges Aufwands/Nutzen-Verhältnis. Nach dem Sprint Planning starten wir den Sprint und haben ein **Sprint Backlog**. Dieses ist aufgrund der vielen Änderungen (ungeplante dringende Anforderungen oder Störungen) jedoch hoch dynamisch. Unser **Scrum Team** besteht aus Linux-, Microsoft-, Datenbank- und Security-Spezialisten. Diese arbeiten sehr eng zusammen. Das Erweitern der Wohlfühzone ist ausdrücklich erwünscht ohne den Anspruch, Spezialist in jedem Bereich zu werden. Ein richtiges **Daily Scrum** nach Lehrbuch machen wir nicht, jedoch haben wir ein Daily Standup, um den Tag zu planen und um jedem den Raum für Fragen oder für Unterstützung zu geben. Alle zwei Wochen vor dem Planning führen wir ein nicht öffentliches **Sprint Review** durch. Warum nicht öffentlich? Wir bieten allen interessierten den „Blue-Board“ Termin (Details folgen) an, um

sich über unsere Arbeit zu informieren. In unserem Sprint Review bringen wir noch einmal auf den Punkt, was wir erreicht haben und was wir bewusst im letzten Planning raus gelassen haben oder was nicht während des Sprints aufgenommen wurde. Dann prüfen wir zwei Wochen nach der Entscheidung, ob unsere damalige Entscheidung heute immer noch richtig ist. Haben wir das Richtige getan? Es ist kein Blame-Game! Es geht darum zu lernen, um künftig bessere Entscheidungen treffen zu können. Direkt nach dem Sprint Review führen wir eine **Sprint Retro** durch. Was lief gut, was lief schlecht? Welche Maßnahmen leiten wir ab? Diese Maßnahmen landen dann wieder auf unserer Transformation Wall.

Blue Board: Die Fäden zusammenhalten

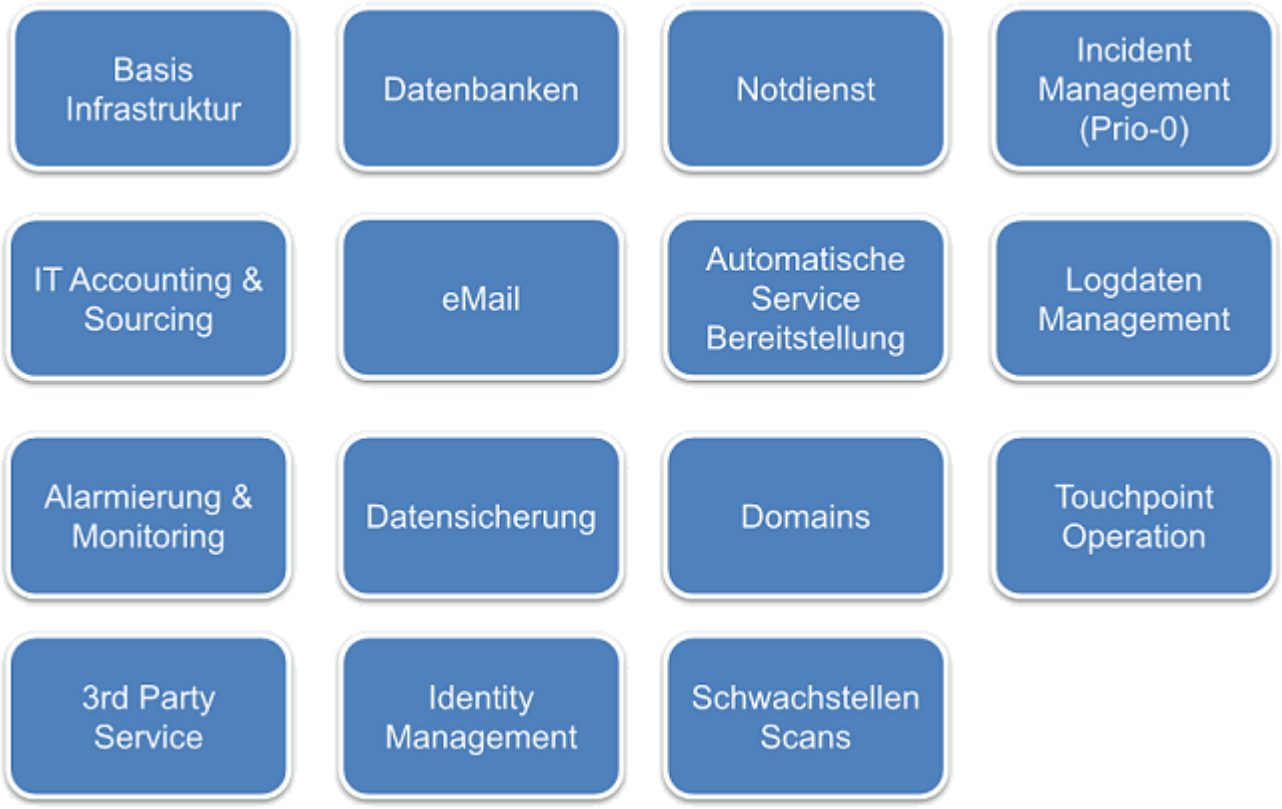
Unser SCRUM-Board hilft uns in der täglichen Arbeit mit den vielen kleinen Aufgaben. Jedoch ergibt es auch Sinn, einen Schritt zurückzugehen, um auch die größeren Themen zu betrachten, die viele dieser kleineren Aufgaben zusammenhalten. Auf einer höheren Flugebene machen wir größere Themen sichtbar. Größere Themen können dabei unternehmensweite Projekte, Operations Projekte aber auch Penetration Test sein. Es sind also Themen, die uns längere Zeit begleiten und Kapazitäten binden. Für jedes Thema haben wir eine gelbe oder eine grüne Karte. Gelbe Karten sind für größere Themen aus dem Tagesgeschäft (Replacements, Updates, Security Scans …) – also Themen, die wir machen müssen, um eine stabile Plattform zu gewährleisten. Grüne Karten sind für größere Themen, die unsere Produkte weiterentwickeln und Mehrwert generieren. Die Themen landen dann an einem physikalischen Board. Lustigerweise ist das ein Kanban-Board, welches wir aufgrund der Board-Farbe ganz kreativ „Blue-Board“ getauft haben. Alle zwei Wochen gibt es zusätzlich zu den Daily Standups eine größere Runde für 60 Minuten, wo wir über den Status der Themen informieren. Diese Runde ist öffentlich, so dass Vertreter aller Produkt-Teams sehen, was gerade bearbeitet wird und welches Thema wann kommt. Sie können sich informieren, ihre Fragen loswerden und Feedback geben.

„Blue-Board“ für die Transparenz und Planung von größeren Operations Themen

Definieren unserer Produkte

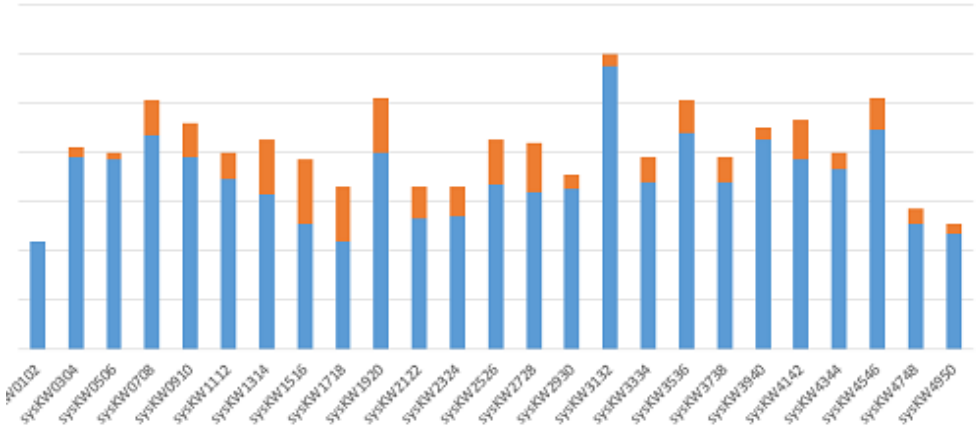
Wir wollen ein Produkt-Team sein. Aber was ist eigentlich unser Produkt? Mit dieser Frage haben wir uns lange auseinandergesetzt. Ist ein Loadbalancer schon ein Produkt? Dann haben wir ja hunderte Produkte. Wenn wir es zusammenfassen, ist dann „die Plattform“ unser Produkt? Hmmm, die Definition hilft nicht so richtig weiter bei der Planung, Strukturierung und Messung. Mit Hilfe unseres Agile Coach haben wir definiert: Ein Produkt ist all das, was einem anderen Team ein Mehrwert bietet. Etwas, wofür jemand im Zweifel auch Geld bezahlen würde.

Und so sind wir losgezogen und haben nach und nach Technologien zu Produkten zusammengefasst, die es uns erlauben, zu strukturieren und sowohl uns als auch allen anderen zu verdeutlichen, was wir eigentlich alles machen. Dabei herausgekommen ist folgende Übersicht:



So haben wir z.B. den Loadbalancer, Firewall, Storage, Netzwerk, Rechenzentrum … zum Produkt **Basis Infrastruktur** zusammengefasst. Wir bieten den anderen Teams unser Produkt **Notdienst** an, welches die Teams nutzen können. Wir bieten den Teams als Produkt eine **Alarmierungs & Monitoring**-Infrastruktur an, die sie nutzen können. Und wie in jedem guten Haushalt so gibt es bei uns auch einen Bereich „Sonstiges“ mit dem klangvollen Produktnamen **3rd Party Service**. Da wir leider noch nicht alle Teams mit einem eigenen Operations-Spezialisten versehen konnten, machen wir für einige **Touchpoint Teams** noch den Operations-Teil.

Messen: Die Basis für eine gute Planung

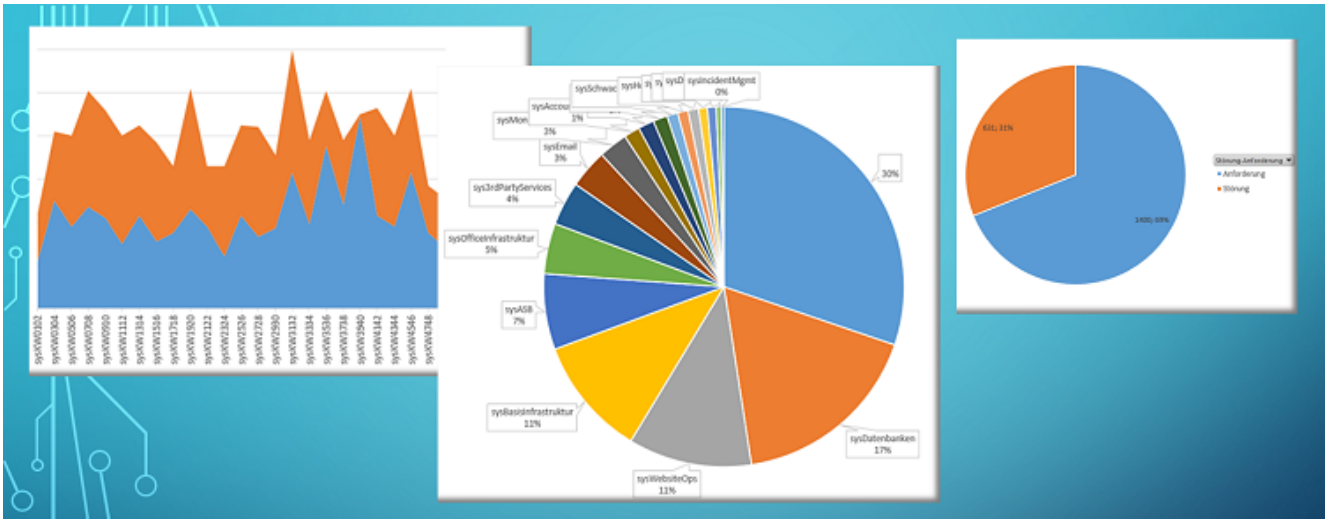


Anzahl der geschlossenen Tickets je Sprint in 2017

Anfangs haben wir überlegt, wie wir unsere Arbeit messen wollen und wie wir Aufwände planen können. Auch brauchten wir eine Möglichkeit zu prüfen, ob unsere Maßnahmen einen positiven Effekt haben. Weiter wollten wir wissen, wie viele Tickets wir in einen Sprint nehmen können. Hier haben wir einiges ausprobiert, verworfen und neu probiert. Um es möglichst einfach zu halten, um möglichst wenig Aufwand zu investieren, haben wir uns darauf geeinigt, die Anzahl der geschlossenen Tickets je Sprint zu messen. Natürlich ist uns klar, dass ein Ticket 5 Minuten oder 5 Stunden dauern kann. Jedoch haben wir festgestellt, dass bei uns die Anzahl der Tickets, die wir schließen, relativ konstant ist und somit eine Größe ist, mit der wir arbeiten können. Klar können wir die Genauigkeit weiter erhöhen, indem wir z.B. auf Storypoints gehen würden, jedoch scheuen wir den Mehraufwand für die Bewertung. Auch erscheint uns der Mehrwert, den wir damit erzeugen, zu gering. Auf der Basis der Anzahl geschlossener Tickets haben wir dann eine Anzahl Tickets abgeleitet, die wir fest in den Sprint einplanen. Leider liegt dieser Wert noch bei unter 50% der Tickets je Sprint. Das Problem ist leider immer noch, dass uns immer wieder ungeplante Aufgaben erreichen, die ihre Berechtigung haben. Bislang funktioniert die Planung basierend auf der Anzahl der Tickets recht gut und stabil. Es gibt auch noch keine Bemühungen, den Aufwand für die Planung (z.B. mit Storypoint) zu erhöhen. So sind wir mehr oder weniger geplant zu Anhängern der #noEstimation-Bewegung geworden.

Neben der Anzahl der Tickets, die uns erreichen, messen wir aber noch mehr – z.B. wie viel Tagesgeschäft und wie viel Weiterentwicklung im Sprint steckt. Wir messen, woher die Tickets kommen, für welches unserer Produkte wir die Arbeit leisten, wie viele der ursprünglich für den Sprint geplanten Tickets tatsächlich

auch fertig werden, das Verhältnis Störung zu Anforderung …



Eine Auswahl an Messungen

Zur Messung exportieren wir die geschlossenen Tickets aus JIRA nach Excel, wo wir sie dann via Pivot mit wenig Aufwand in ein paar Graphen visualisieren. Um die Tickets auswerten zu können, nutzen wir u.a. drei JIRA Stichwörter: (1) Ein Stichwort für die Kalenderwochen, in denen das Ticket bearbeitet wurde, (2) Ein Stichwort, um das Ticket einem unserer Produkte zuzuordnen und (3) haben wir ein Stichwort, um ein Ticket als Entwicklungsticket zu kennzeichnen (ohne dieses Stichwort ist es ein Tagesgeschäftsticket).

XFT – oder der Microsoft Spezialist der MySQL DBs auf Linux installiert

Super, jetzt haben wir ein wenig „agiles Mindset“, einen Agile Coach und ein paar Boards. Wir können nun priorisieren, um das Richtige zur richtigen Zeit tun. Aber was, wenn der Spezialist für das Richtige gerade im Urlaub ist? Im Team waren wir so organisiert, dass wir für Linux-, Datenbank-, Security- und Windows-Themen Spezialisten haben, die ihr Handwerk wirklich gut verstehen. Dadurch hatten wir jedoch, wenn man so will, vier Teams in einem Team. Diese künstlichen Hürden wollten wir entfernen und dazu die Wohlfühlzonen, die wir hatten, erweitern. Warum sollte nicht der Windows-Spezialist eine MySQL-Datenbank auf einer Linux-Maschine installieren, wenn das gerade die wichtigste

Arbeit ist? Als kleiner Nebeneffekt hat er seine Wohlfühlzone und sein KnowHow erweitert. Zudem macht es auch noch Spaß, neue Sachen zu versuchen. Aber funktioniert das auch im Alltag? Um das herauszufinden, haben wir jeden Ausflug raus aus der Wohlfühlzone auf einer einfachen Matrix gemessen und in der Retro besprochen. Die Reaktionen im Team waren ziemlich positiv, und die Fokussierung auf die umzusetzenden Themen im Sprint hatte positive Auswirkungen auf die Bearbeitungsgeschwindigkeit. Grenzt das nicht an ein Cross-Functional-Team (kurz XFT)?

Erweitere deine Wohlfühlzone. Wer hat mit wem zusammengearbeitet?

DevOps ist keine Rolle, oder? Operations KnowHow in die Produkt-Teams

Dank der guten und nicht ganz einfachen Arbeit der Kollegen wissen wir bereits, welche Produkt-Teams es gibt und welche Software-Services sie entwickeln und

betreiben sollen. Auf dieser Basis konnten wir Team-Mitglieder suchen, die Lust auf ein neues Abenteuer haben. So haben wir Linux Spezialisten gefunden, die bereit waren, zwei oder auch drei (je nach Teamgröße) Produkt-Teams zu betreuen. Aber was bedeutet das? Um die Teams zu betreuen, sind die Linux Spezialisten sowohl fachlich als auch physisch in die Produkt-Teams gegangen und fester Bestandteil der Teams geworden. Hurra! Endlich haben wir unseren eigenen Linux Spezialisten, der alle Operations Aufgaben erledigen kann! Nope! Die Idee ist, dass er zwar der Operations Spezialist im Team ist. Jedoch ist seine Aufgabe, sein Operations-KnowHow so weit wie möglich zu verbreiten und das Team in die Lage zu versetzen, eigenständig alle für ihr Produkt notwendigen Operations Aufgaben umzusetzen. Wir erinnern uns: Wenig Reibungsverlust an Schnittstellen. Möglichst viel selber machen. Gilt auch im Kleinen. Wollen wir nun z.B. die Entwickler zu Operations Spezialisten machen? Nein! Wir wollen, dass die Teams in der Lage sind, alles, was nötig ist, selber zu machen. Im Optimalfall gibt es einen einfachen SelfService mit einer Entwickler Schnittstelle, die man mit geringem KnowHow bedienen kann. Die ganze Operations Magie passiert dann automatisch im Hintergrund. Um zu verstehen, wie die optimale Schnittstelle zwischen Platform Engineering und dem Produkt-Team aussieht, haben wir zusammen mit den Linux Spezialisten in den Teams definiert, was in den Teams in welcher Tiefe passieren soll und wo Platform Engineering eine einfache Schnittstelle bereitstellen muss. Auch wenn jede Schnittstelle im Optimalfall perfekt mit einer großartigen GUI oder API automatisiert ist, so ist das besonders zum Start nicht realistisch. Im Zweifel kann eine Schnittstelle auch ein Ticket oder eine gute Dokumentation sein. Automatisiert aber so schnell und so früh wie möglich die Schnittstellen. So, nun sind die Linux Spezialisten in den Produkt-Teams, weg, raus aus dem Platform Team. Und wie bleiben die Ops Spezialisten in den Teams nun am Operations Puls der Zeit? Das ist recht einfach. Zum Einen nehmen sie weiter an den bereits etablierten Meetings teil.

Unser Daily Standup mit Teilnehmern vor Ort, in Hagen, Berlin und im Homeoffice.

So können Sie an den Daily Standups oder an den alle zwei Wochen stattfindenden „Blue Board“ Meetings teilnehmen (da wo wir die größeren Themen und deren Priorität und Fortschritt besprechen). Das ist alles nice und hilft. Besonders wichtig ist jedoch ein neu geschaffenes Meeting. Noch ein Meeting? Och nöö! Doch, ergibt Sinn. Tut nicht wirklich weh, bringt dafür aber viel. Neu etabliert haben wir die „Operations Community of Practice“ – kurz: OpsCoP. Hier treffen sich alle zwei Wochen die Ops Spezialisten aus allen Produkt Teams sowie ein Vertreter aus dem Platform Engineering Team. Dort tauschen wir unsere Erfahrungen zwischen den Teams aus. Was kann ich aus Team A lernen und für Team B übernehmen? Welche Entwicklung passiert gerade im Platform Team oder welche Known Issues gibt es? Und so weiter ….

Irgendwie ist es dann doch noch passiert, das unsere Linux-Spezialisten in den Produkt-Teams nur noch „die DevOps“ genant werden. Im Grunde total falsch! DevOps ist keine Rolle. Wir haben sogar einen Versuch gestartet das wieder raus zu bekommen. Erfolglos. Die falsche Bezeichnung DevOps erfreut sich großer Beliebtheit bei uns. Nun lassen wir es einfach so, jedoch immer mit der Ergänzung dass es eigentlich falsch ist

Die Schnittstelle zwischen Platform Engineering und den Produkt-Teams

Cool, jetzt haben wir die Produkt Teams mit Operations KnowHow ausgerüstet, verteilen dieses, machen die Teams dadurch schneller. Auch die Schnittstellen sowie der Informationsfluss zwischen Platform Engineering und Produkt Teams ist geklärt. Aber was machen eigentlich die Operations-Kollegen, wenn die Produkt Teams ihre Sachen selber betreiben? Wir brauchen ein neues Ziel. Früher war unser Ziel, alle Systeme und Anwendungen 7x24h zu betreiben. Nun machen das (zumindest in großen Teilen) die Produkt Teams. Die Antwort ist einfach, die Angst völlig unberechtigt. Wir sind inzwischen kein echter IT-Betrieb mehr. Da war doch was. Hatten wir uns nicht umbenannt? Platform Engineering…. Was ist das eigentlich? Wir engineeren (gibt es das Wort im deutschen überhaupt? Sorry) eine Plattform für alle Services. Aber gehört alles zur Plattform? In großen Teilen helfen uns da die Schnittstellen, die wir zusammen mit den Produkt Teams definiert haben. Alles was ein Produkt Team braucht, um einen Service zu bauen und zu betreiben, machen die Produkt Teams. Alles andere macht das Platform Engineering. Beispiel: Das Produkt-Team braucht einen Knopf, aus dem ein Server mit einem Service rauskommt. Den Knopf mit all seiner Logik und seinen Automatismen baut das Platform Engineering Team. Ein neuer Blickwinkel. Wir bauen und betreiben nun Services, die es erlauben, Server zu erstellen. Den erstellten Server betreibt nun jedoch jemand anderes. Oder ein anderes Beispiel aus dem Bereich Datenbanken: Wir definieren nun, welche MySQL Versionen automatisiert bereitgestellt werden. Wir definieren verpflichtende Sicherheitskonfiguration. Aus unserer Erfahrung schlagen wir Best Practice Konfigurationen vor, die von Produkt Teams genutzt werden können, nicht müssen. Wir stellen getestete Patches und zugehörige Dokumentation bereit. Die Produkt Teams müssen jedoch all diese Änderungen selber implementieren und betreiben. Dadurch ergeben sich für uns neue Möglichkeiten. Endlich haben wir eine Chance, uns auf neue Innovationsthemen zu stürzen. Wir haben die Chance, die Automation weiter auszubauen. Wir als

Platform Engineering verstehen uns als Anbieter von Technologie für die Produkt Teams, ohne diese auf unsere Technologie limitieren zu wollen. Warum sollte ein echt schnelles Produkt-Team Monate darauf warten, von uns eine Technologie zu bekommen? Warum sollte nicht auch ein Produkt-Team zusammen mit dem Linux Spezialisten in Abstimmung mit dem Platform Engineering Team eine neue Technologie evaluieren und diese Technologie dann über das Platform Engineering Team allen anderen Teams zur Verfügung stellen? Wichtig ist hier wieder die enge Abstimmung der Bereiche. Sobald mehr als ein Team eine neue Technologie nutzen möchte, so übernehmen wir aus dem Platform Engineering den aktuellen Stand und stellen es allen Teams zur Verfügung. Müssen wir deswegen jede Technologie selber entwickeln? Nein, natürlich nicht. Was spricht dagegen, eine Monitoring Lösung, ein CDN, einen DDoS-Schutz und so weiter einzukaufen und den Produkt-Teams zur Verfügung zu stellen? Warum nicht Technologie aus der Public Cloud einkaufen und allen anbieten. Das macht uns wieder schneller. Es ist immer eine Frage des Preis-/Leistungsverhältnisses. Hier darf man jedoch nicht vergessen, dass auch der billiger aufgebaute eigene Service am Ende sehr viel teurer sein kann als der teuer eingekaufte Service. Ausschlaggebend ist natürlich auch der Faktor „time to market“. Wenn der teuer eingekaufte Service es uns ermöglicht, zwei Monate schneller am Markt zu sein, und somit zwei Monate mehr Umsatz generiert, ist das ein wichtiges Entscheidungskriterium.

Was haben wir gelernt?

- Mache kein SCRUM, nur weil alle es machen. Mache es, weil es dir hilft.
- Sei mutig. Mache Fehler. Lerne.
- Lass dir helfen. Helfe anderen. Vertraue deinem Team.
- Probiere aus. Laufe vor die Wand, um zu verstehen, was man besser machen kann.
- Reduziere Schnittstellen auf ein Minimum und automatisiere den Rest und noch mehr.
- Sei kein Blocker. Wenn du ein Blocker bist, verändere es.

- Messe, was du tust, und mach die Ergebnisse sichtbar.
- Tue das Richtige zur richtigen Zeit. Mache dazu deine gesamte Arbeit sichtbar und priorisiere sie nach dem Nutzen, den die Arbeit erzeugt.
- Mache sichtbar, was du nicht machst.
- Stelle die Aufgabe in den Mittelpunkt und bearbeite sie, auch wenn du länger brauchst als der Spezialist.
- Der Klassiker: Stop starting, start finishing. Es macht dich langsam, wenn du alle Themen gleichzeitig machst.
- Gib Raum für Eskalationen. Eskalationen sind hilfreich und nicht böse.
- Spreche die gleiche Sprache wie dein Dev Kollege.
- Erweitere deine Wohlfühlzone und habe Spaß dabei.
- Lass dich inspirieren. Gehe auf Konferenzen, rede mit Kollegen aus anderen Unternehmen oder ganz einfach: Rede mal mit deinen Entwicklern

Sicherlich habe ich einige Punkte vergessen. Ich möchte mit diesem Reisebericht jedoch unsere wichtigsten Eckpunkte, Erfahrungen und Gedanken vom IT-Betrieb hin zum Platform Engineering teilen. **Ich würde mich total über Feedback in den Kommentaren freuen.** Was sind eure Gedanken zu unserer Reise? Was sind eure Erfahrungen? Was können wir noch besser machen? Wir sind auch offen für einen persönlichen Erfahrungsaustausch. An dieser Stelle ein großes Dankeschön an Kollegen von Otto.de oder dem LVM für tolle inspirierende Gespräche. Ich hoffe, wir konnten auch etwas zurückgeben.

Alle drei Kapitel im Überblick

Kapitel 1: Woher kommen wir? 6 Jahre im Schnelldurchlauf

Kapitel 2: Basistechnologien, SelfServices & Automation

Kapitel 3: Mindset, Methoden, Prozesse und mehr…