Scrum Deutschland 2018 – Domain Driven Scrum

Crossfunktionale, selbstorganisierte und -verantwortliche Teams sind das Herzstück unserer Produktorganisation. Um den damit verbundenen Herausforderungen, insbesondere im Hinblick auf die Verantwortungsübergabe- und übernahme, gerecht zu werden haben wir auch unsere bisherigen Führungsrollen verändert.

Zum Startpunkt unserer agilen Transformation vor 2 Jahren gab es weder Scrum Master, Agile Coaches oder erfahrene Servant Leader bei uns – aber natürlich Teamleiter und andere Führungskräfte:

- Wie sind wir in dieser Situation gestartet?
- Welche Annahmen in Bezug auf Trennschärfe und mögliche Überschneidungen haben wir zwischen den beiden Rollen getroffen?
- Sind für unsere Teams (und uns) die jeweiligen Aufgabenbereiche klar definiert?
- Wie hat sich in dieser Zeit die Zusammenarbeit zwischen Coaches und Leader entwickelt?

Entlang dieser Fragen haben wir bei Scrum Deutschland 2018 berichtet – aus den beiden Sichtweisen Scrum Master und Servant Leader mit konkreten Erfahrungen, Erkenntnissen und Vorschlägen.

Eine tolle Gelegenheit für uns in den Austausch mit anderen Interessierten zu gehen und nach dem Vortrag weiter zu diskutieren und Erfahrungen auszutauschen – es hat Spaß gemacht!

	
Matthias Hochschulz | m.hochschulz@thalia.de	Claudia Landmesser | c.landmesser@thalia.de

---

Nachhaltiger Retro-to-go Automat

Flipcharts sind ein wichtiger Bestandteil unserer täglichen Arbeit als Scrum Master. Sie sind aber auch ein Beitrag zur Reduzierung von Rohstoffen.
Im Sinne von „Wiederverwendung“ sowie „Vermeidung von *Waste*“ bewahren wir gerne unsere Werke auf. Dabei entstehen schnell unübersichtliche Stapel oder zahllose Papierrollen in Regalen und irgendwann tauchen die Fragen auf:

- „Benötigen wir das überhaupt noch?“
- oder „Wo habe ich denn die Vorlage zur Aktivität *Mad, Sad, Glad* hingelegt?“

…vielleicht auch „Was ist das denn da in der Ecke?“

Sollte ich dann wirklich etwas gefunden haben stellt sich oft heraus, dass die Vorlage nicht mehr zu gebrauchen ist, weil:

1. das Papier sich immer wieder einrollt
2. alles total verknickt ist
3. der Bereich mit der Lochung gerissen ist
4. die Ecken verknickt oder gerissen sind
5. die Beiträge mit dem Stift direkt eingetragen wurden.
6. Dot-Voting Punkte mit dem Stift gemacht wurden.

Für die letzten beiden Punkte haben wir keine gescheite *Post-Mortem-Lösung* :), hier hilft nur bereits bei der Vorbereitung zu überlegen, wie die Teilnehmerbeiträge visualisiert werden.
Für die anderen Punkte haben wir uns eine simple Lösung einfallen lassen, die Ihr auch einfach – beispielsweise hier – bestellen könnt.

Die Ausgestaltung in unserem Team möchte ich kurz vorstellen.

Wir haben uns für die Aufbewahrung an einem Kleiderständer entschieden. An jedem Klemmbügel können mindestens 2 Charts gleichzeitig aufgehangen werden. Mit unterschiedlichen Bereichsmarkierungen haben wir nach den Ebenen der Retrospektive sowie besonderen Workshopformaten getrennt. Ein leerer Bügel am Kopf ist immer bereit für die aktuell zu planende Retrospektive.

Aus unserer Sicht bietet diese Lösung folgende Vorteile:

- Alle Aktivitäten und Flipcharts sind auf einen Blick
- Das Papier rollt sich nicht ein
- Es ist sogar als mobiler Katalog einsetzbar (*ein Fahrstuhl ist praktisch zwischen mehreren Etagen*)
- Mit den Kleiderbügeln wird es zu einem *„Retro-to-go“* – Erlebnis

Wie geht ihr mit Euren Flipcharts um? Habt ihr auch eine besondere Lösung?

Musencast II: Die Retrospektive

Die zweite Folge unseres Podcast beschäftigt sich mit dem Meeting „Retrospektive“ aus dem Scrum Framework. Viel Spaß beim Hören! Lasst uns gerne Euer Feedback zukommen.

https://tech.thalia.de/wp-content/uploads/2018/10/2018_10_12_-_musencast_-_retrospektive_-_prod.mp3

<#0100/> <hackathon@thalia/> in Berlin

Ende Mai fand der 2. Hackathon in Berlin statt. Zum Thema waren diesmal alle neuen Technologien gesetz.

Mein Erfahrungsbericht aus dem Event-Marketing: Scrum Workshop in Berlin

„Simulationen sind toll! Sie sind eine große Hilfe, um Verständnis für eine neue Arbeitsform und -kultur zu schaffen!“

Dieses Statement bekam ich vor Kurzem beim Agilen Stammtisch in Dortmund. Im Vorfeld hatte ich unseren Thalia Scrum Workshop vorgestellt. Bei diesem Workshop nutzen wir eine Simulation, um neben viel Sprechen über Theorie und Bilder auch die haptische Seite im Gehirn zu beteiligen. Nach einer kurzen Einführung in das Framework und damit auch in die Spielregeln für die Simulation heißt es Ärmel hochkrempeln. Der Schwerpunkt liegt auf Ausprobieren, Erleben und Reflektieren.

Im Oktober 2017 habe ich hier bereits einen kurzen Artikel geschrieben und das Vorgehen vorgestellt. Im Januar kam mein Kollege Jens vom Standort in Berlin auf mich zu und bat mich diesen Workshop dort auch anzubieten.
Die Reaktionen auf meine erste Anfrage bei den Mitarbeitern am Standort ließen

den Raum noch nicht in Gänze füllen. So richtig fängt der „Spaß“ ab 20 Teilnehmer*innen an. Material und Ausstattung reichen für bis zu 30 Teilnehmer*innen (das entspricht ca. 6 Teams à 5 Mitglieder). Also beschlossen wir aus dem Mitarbeiter*innen-Workshop eine offene und kostenlose Veranstaltung zu machen. Für mich hat sich hier eine tolle Möglichkeit aufgetan auch den Fortschritt unserer agilen Transformation vorzustellen.

Meine erste Erkenntnis: „Marketing is King“

Die Location war einfach gefunden. Unser Standort in Berlin liegt in den Sarotti-Höfen am Mehringdamm und im Erdgeschoss bietet die Event-Agentur „Schmelzwerk“ passende Räumlichkeiten an. Ein Termin war schnell abgestimmt und das Xing-Event war fast zeitgleich online…

…aber irgendwie passierte da nix. *„Vielleicht teile ich das Ganze nochmal über mein großes Netzwerk*.“ dachte ich mir. Leider ohne sichtbaren Effekt. Eine kurze Beratung mit unserer Marketing-Abteilung deckte auf: Wichtigste Kriterien für eine ansprechende Veranstaltung werden nicht erfüllt:

- Warum sollte ich zu dieser Veranstaltung gehen?
- Was kann ich dort mitnehmen?

Dank unserer Expertise im Haus und der Unterstützung unserer Grafiker*innen und Texter*innen konnten wir rechtzeitig unser Event in Szene setzen. Das Feedback ließ gar nicht lange auf sich warten. Der Raum war zeitnah gefüllt und die nächsten Vorbereitungen konnten starten. An dieser Stelle: **Vielen Dank für Eure Hilfe!**

Jede Retrospektive benötigt einen Raum oder Scrum Master

Am Workshoptag selbst hatten wir wieder viel Zuspruch durch das Wetter und so konnten alle Teilnehmer*innen mit viel Spaß und spielerischem Wettkampfgedanken ein Produkt ganz nach dem Scrum-Regelwerk erstellen. In den Reviews wurde gerne mit Wasser gespielt und die Abnahmetests durch den Product Owner oder die Product Ownerin überstanden. Das Wichtigste: Alle Teilnehmer*innen hatten 5 Sprints, in denen sie den Rhythmus von kontinuierlicher Entwicklung inklusive Feedback erleben konnten. Zudem haben alle Teams ein „Wir" entwickelt. Beim „Commitment" in der Planung wurde anfangs noch von Einzelnen entschieden zum Ende jedoch immer erst nach Abstimmung untereinander. Eine weitere Verbesserung sehe ich noch hier: Retrospektiven sind das Herz von „Inspect & Adapt" und sie müssen auch hier gelebt werden. Bei dem Vorgehen bis hierher gab es immer wieder folgende Beobachtungen:

- Es wurde weiter gebastelt
- Es wurde nicht über das Vorgehen geredet
- Es wurde nichts verbessert
- Die Zeit für die Retrospektive wurde als Vorlauf für den kommenden Sprint genutzt

Ein separater Tisch / Raum pro Team könnte helfen. Leider findet man solche Locations nur selten. Meine pragmatische Idee: Ein*e „Scrum Master*in" sollte bestimmt werden, der den Prozess überwacht und darauf achtet, dass das Team sein Vorgehen hinterfragt und anstrebt nicht nur das Produkt sondern auch sein eigenes Vorgehen zu verbessern.

Feedback: Meine Teilnehmer*innen möchten auch gerne für sich reflektieren

Nach einer kleinen Pause zur Hälfte des Workshop habe ich eine kurze Runde zur Reflektion des Framework und dem angewendeten Vorgehen gemacht. Das Feedback, was mir meine Teilnehmer*innen nach dem Workshop gegeben haben, war: So eine Runde sollte zumindest am Ende des Workshops wiederholt werden.

Es muss eine Übertragung auf den Alltag geben. Meine Teilnehmer*innen wollen gerne wissen, was sie von hier in den Alltag überführen können. Dieses Feedback hilft mir sehr bei meiner weiteren Entwicklung des Formats.

Für mich war es daher ein voller Erfolg und ich freue mich auf die Verbesserungen in der nächsten Ausprägung.

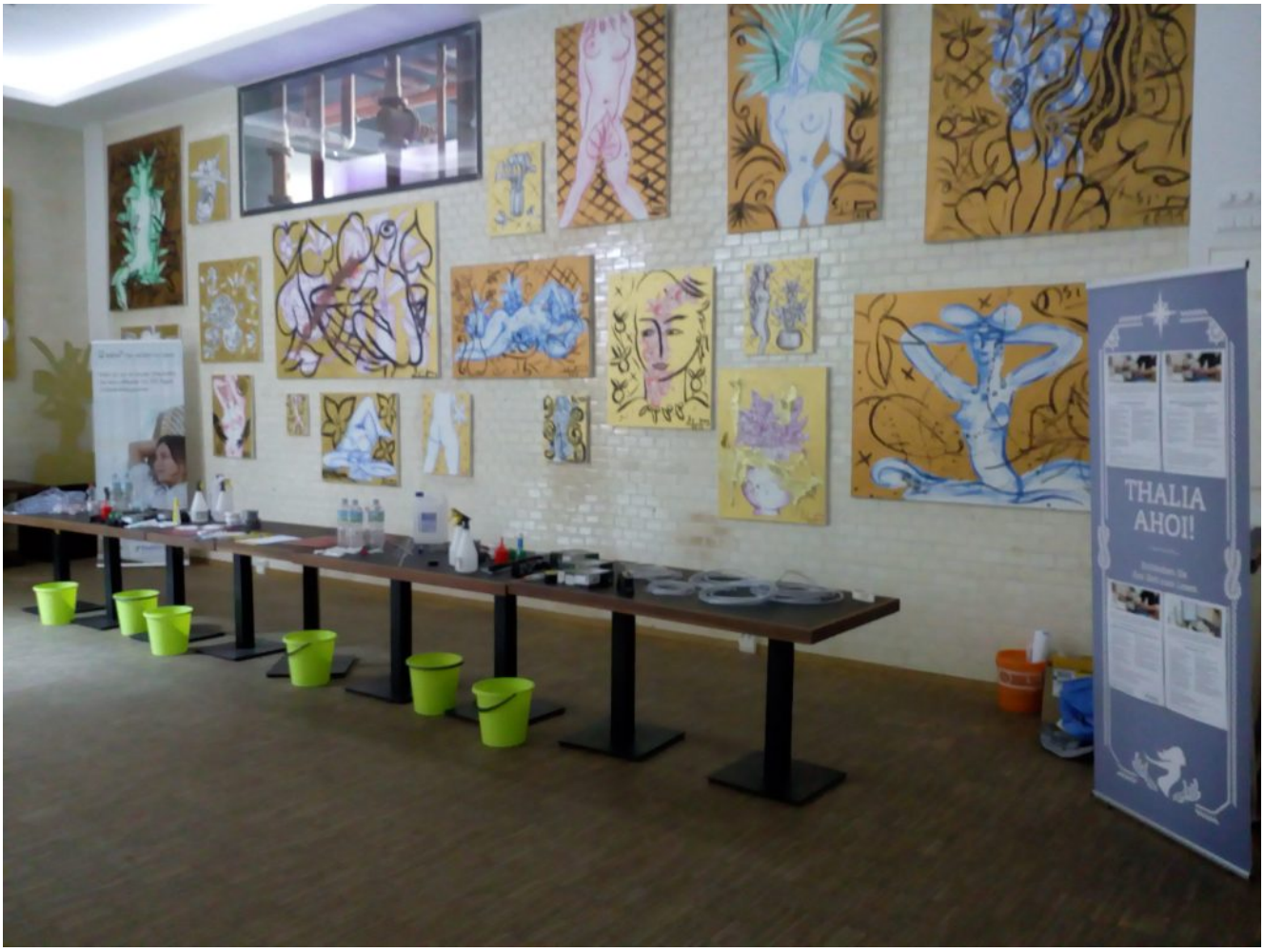
Habt ihr auch Bock drauf? Wollt ihr auch gerne so einen Workshop erleben oder anbieten, kontaktiert mich einfach!

Impressionen

Nachfolgend ein Testimonial, das ich von einem Teilnehmer erhalten habe:

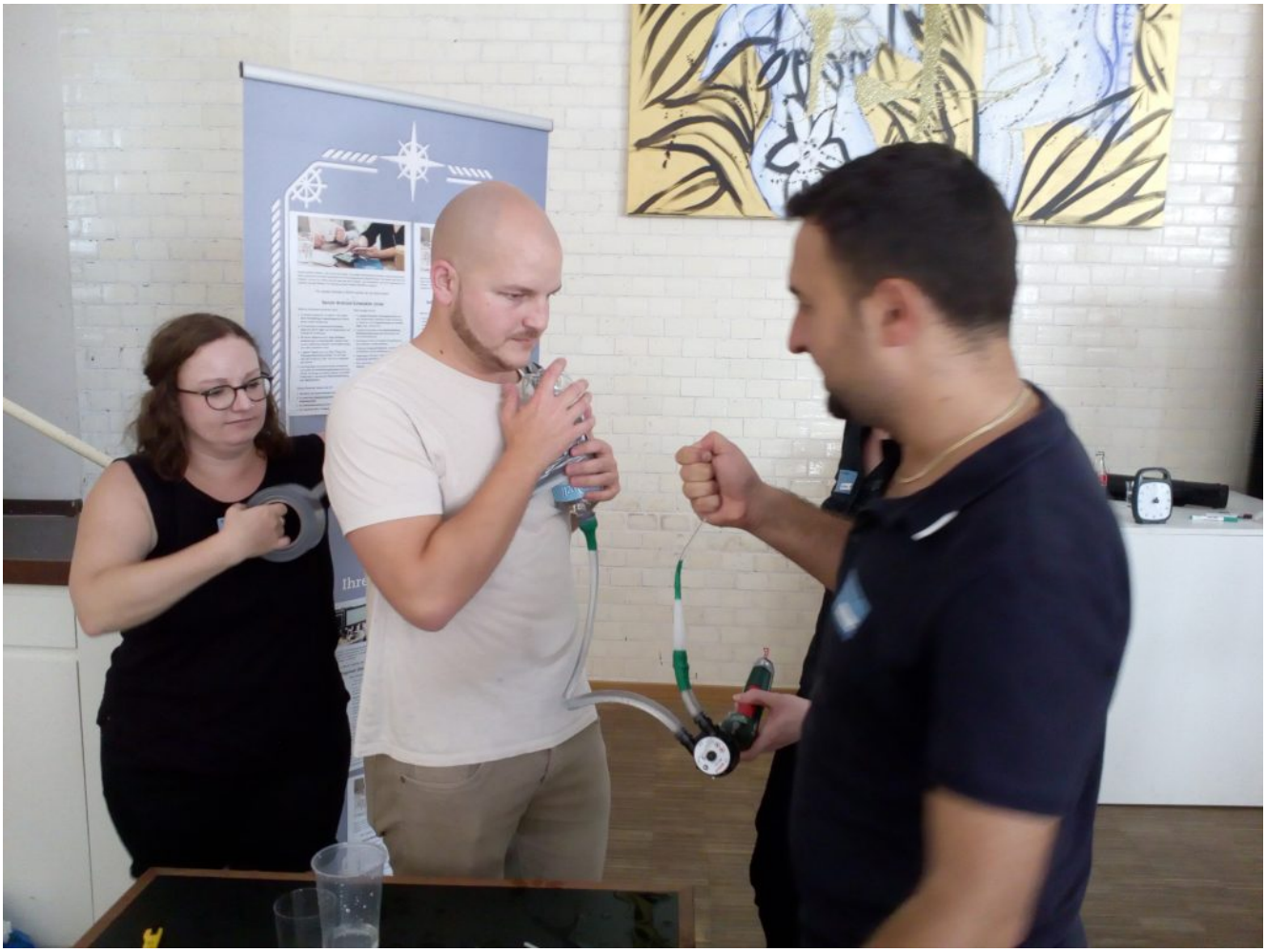
„In drei Stunden habe ich die Grundlagen von SCRUM kennengelernt und das erste Mal praktisch anwenden können. Aufgeteilt in heterogenen Teams wurden wir mit praktischen Herausforderungen konfrontiert, bei denen die Inhalte gut vermitteln wurden und der Spaß nicht auf der Strecke blieb."

Hier findet ihr noch ein paar ausgewählte Impressionen:


THALIA
AHOI!


THALIA
AHOI!


Salvelt



Berlin macht Agil !

Dank der Unterstützung unseres Kollegen Matthias Hochschulz und seiner Organisation Münster macht Agil, konnten wir gemeinsam im August auch in Berlin einen kostenlosen Scrum Wokshop organisieren. Ursprünglich getrieben

von der Notwendigkeit einer Reihe von neuen Kollegen die Grundwerte und -prinzipien von Scrum nahezubringen, entwickelten wir gemeinsam die Idee den Workshop auch für externe zu öffnen. So konnten wir am Ende fast 30 Interessierten aus und um Berlin dazu ermutigen, an einem realen Produkt, mehrere kurze Sprints zu durchlaufen.

Vorallem der Schritt zu einer öffentlichen Veranstaltung war für uns neu. Passte aber sehr gut zu unseren Bemühungen unsere Kompetenz in der Softwareentwicklung nach außen zu tragen. Für viele passt das Bild eines modernen agilen Softwarehauses nicht zu dem angestaubten Bild eines Buchhändlers. Daher haben wir uns sehr gefreut, das wir zeigen konnten, das dieses Bildniss in der Tat sehr veraltet ist und wir eine langjährige Erfahrung im eCommerce und der agilen Softwareentwicklung haben.

Nach einer kurzen Einführung (Stichworte: Sprint,
Review, Retrospektive, Daily-Stand-Up, Inspect & Adapt, Fokus und Transparenz) ging es direkt ans *Selbermachen*! Passend zum heißen Sommer sollten die Teams eine Wasserpistole aus einem Akkuschrauber bauen. Diese konnte über mehrere Sprints verbessert werden. Das ermöglichte, ganz im agilen Sinne, aus den ersten Erfahrungen zu lernen und auf neue Anforderungen zu regieren.

Nach dem erfolgreichen Event konnten wir noch alle Beteiligen noch auf ein Bier und Pizza in unsere Büroräume einladen und uns so für künftigen Erfahrungsaustausch vernetzen.

Link zum Xing Event

Nachtrag:
Matthias hat hier zusätzlich einen sehr umfangreichen Erfahrungsbericht zu dem Workshop verfasst. Dieser reflektiert sehr ausführlich die Erfahrungen die Matthias während des Workshops als Moderator gesammelt hat.

Retrospektive im Freien: Starke Böen berücksichtigen!

Es ist heiß in den Speichern in Münster-Coerde. In Zeiten des Sprintwechsel heißt es dann „die Ventilatoren starten“:

- Viele Meetings hintereinander…
- mit vielen Menschen auf engem Raum…
- gegebenenfalls schwierige Themen…
- keine Eisdiele vor Ort!

Als Scrum Master gehen wir daher „offen“ für das Team mit den Meetings um. Was könnte an einem Tag mit gefühlten 32 Grad im Büro und Meetingraum angenehmer sein, als eine „Inspect & Adapt“-Session unter freiem Himmel. Voraussetzung dafür ist natürlich, dass dem Team bewusst ist, dass dem „**Vegas-Prinzip**“ im Zweifel nicht zu 100% gerecht wird. „Was in der Retrospektive passiert, bleibt in der Retrospektive“ besagt dieses Prinzip und zahlt unmittelbar auf die Sicherheit im Team ein. Daher solltet ihr sofern möglich den Platz für die Retrospektive nicht in unmittelbarer Nähe eurer Büros wählen. In Zeiten von „Storming“ rate ich auch immer zu einer Sicherstellung dieses Prinzips (zur Not doch in geschlossenen Räumen). Entscheidet Euch bei der Auswahl der Visualisierung für eine „einfache Umsetzung“. Ein Flipchart-Papier mit Kreppband an einer Hauswand funktioniert. Ein Flipchart-Ständer, der über Fahrstühle, Treppen und Kopfsteinpflaster für großen Overhead und nur Gelächter sorgt sollte kritisch betrachtet werden.

Einige Teams haben diesen Kontext auch für andere Meetings bereits adaptiert und veranstalten auch ihr „Planning“ vor den Türen auf der Wiese. Die Product Owner sollten hierfür Ihr Thema und die oberen Storys aus dem Backlog natürlich mitbringen. Die Storys und Unteraufgaben werden anschließend mit dem elektronischen System synchronisiert.

Als Moderator hat man gegebenenfalls viel zu tun. Zu einer Retrospektive im Stehen draußen fühlen sich die Teilnehmer so befreit, dass sie schon mal „aus der Reihe tanzen“ und ein runder gemeinsamer Kreis schwierig zusammen zu halten ist. Sucht Euch einen Platz, wo alle sitzen können oder alternativ vielleicht einen Punkt, um den sich alle versammeln können. Dort könnt ihr dann ja auch als

zentrales Hilfsmittel Materialien wie Post-its und Stifte bereitstellen. Wichtig ist, dass ihr „gut“ haftende Post-its („Super Sticky Notes“) verwendet oder im Zweifel die Beiträge direkt auf dem Flipchart sammelt. Andernfalls habt ihr anschließend wieder die Herausforderung die Leute einzusammeln, weil sie Ihren Beiträgen hinterher jagen…

Eine weitere Erkenntnis war für mich, dass weiße Flächen sehr gut reflektieren und daher ein Flipchart nicht gegen Sonne und Teilnehmer gehalten werden sollte. Bei einer freien Fläche oder mit nur einer Wand bietet sich dann vielleicht doch ein mobiles Flipchart an oder alternativ die Visualisierung direkt auf dem Boden.

Jetzt seid ihr gefragt: Habt ihr auch schon Retrospektiven draußen veranstaltet? Was sind eure Erfahrungen? Was unternehmt ihr sonst Besonderes für einen Kontextwechsel?

Nachfolgend findet ihr ein paar Impressionen!

Podcast: Grundlage zu Scrum Review aus Sicht der Scrum

Master

Patrick und Matthias möchten in Ihrer ersten Podcast-Episode die Umgebung für eine Podcast-Serie schaffen. Thema heute ist „Scrum Sprint Review“.

https://tech.thalia.de/wp-content/uploads/2018/05/2018-04-24-Podcast-Review-dev.mp3

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

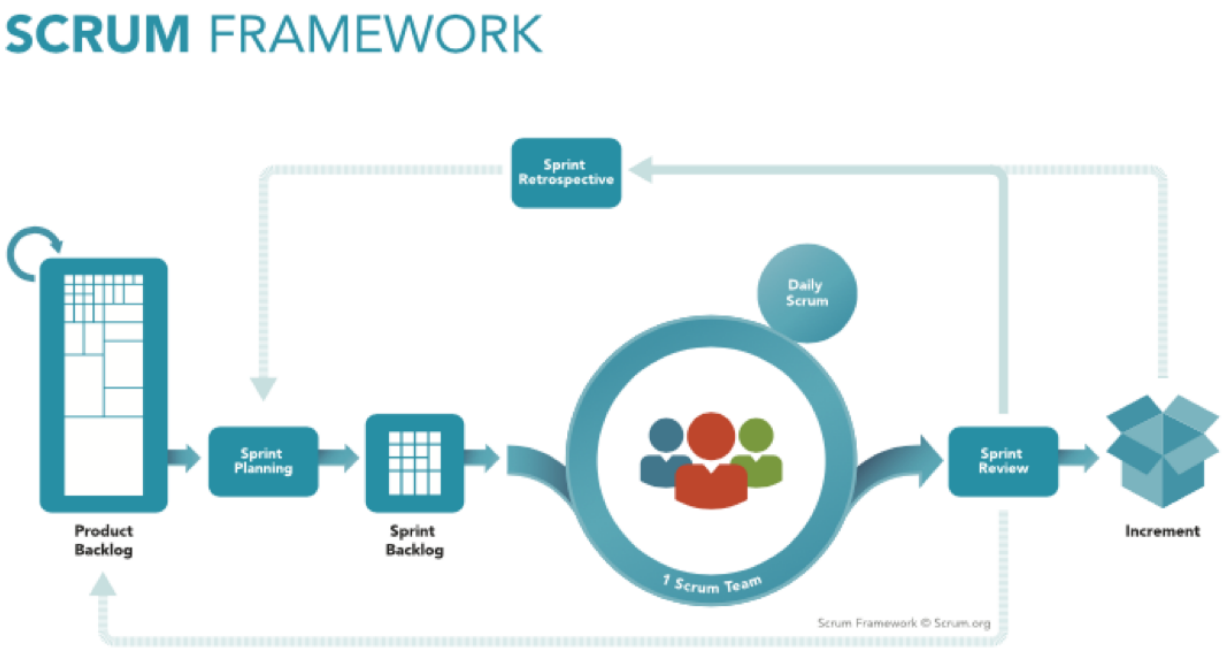
Ein Problem gab es jedoch noch: Unser SCRUM-Testballon wurde in einer reinen Projektumgebung ohne Operations-Tagesgeschäft durchgeführt. Hier gab es keine größeren Störungen, keine spontanen und dringende Anforderungen. Im Operations Tagesgeschäft wimmelt es nur so von unerwarteten Änderungen, worauf 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

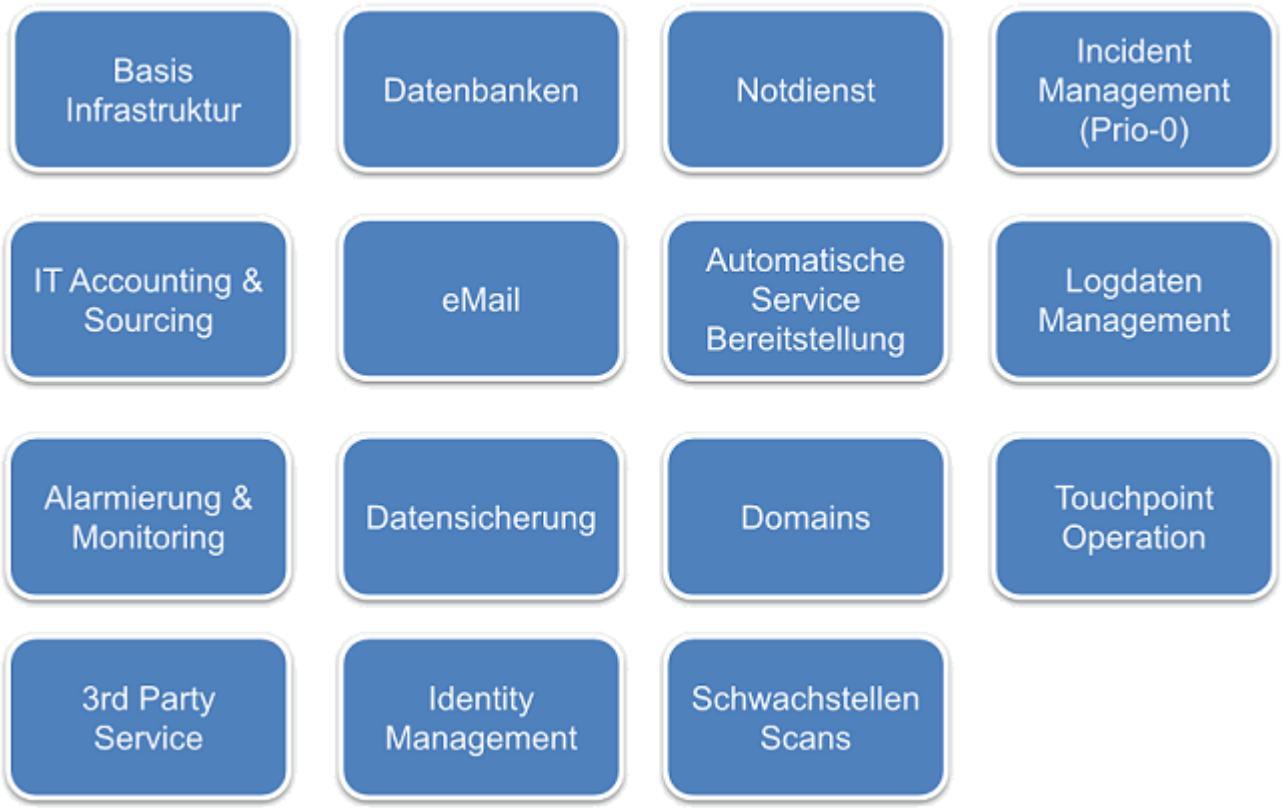
Produkt-Teams sehen, was gerade bearbeitet wird und welches Thema wann kommt. Sie können sich informieren, ihre Fragen loswerden und Feedback geben.

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

Definieren unserer Produkte

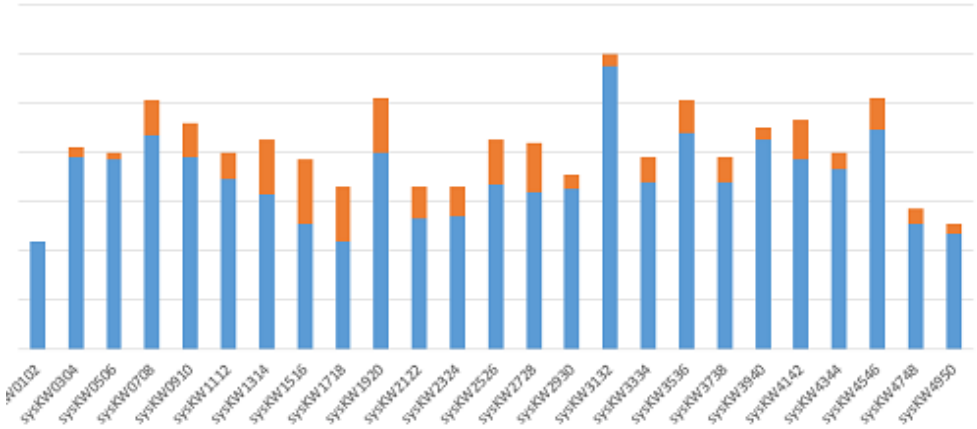
Wir wollen ein Produkt-Team sein. Aber was ist eigentlich unser Produkt? Mit dieser Frage haben wir uns lange auseinandergesetzt. Ist ein Loadbalancer schon ein Produkt? Dann haben wir ja hunderte Produkte. Wenn wir es zusammenfassen, ist dann „die Plattform“ unser Produkt? Hmmm, die Definition hilft 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

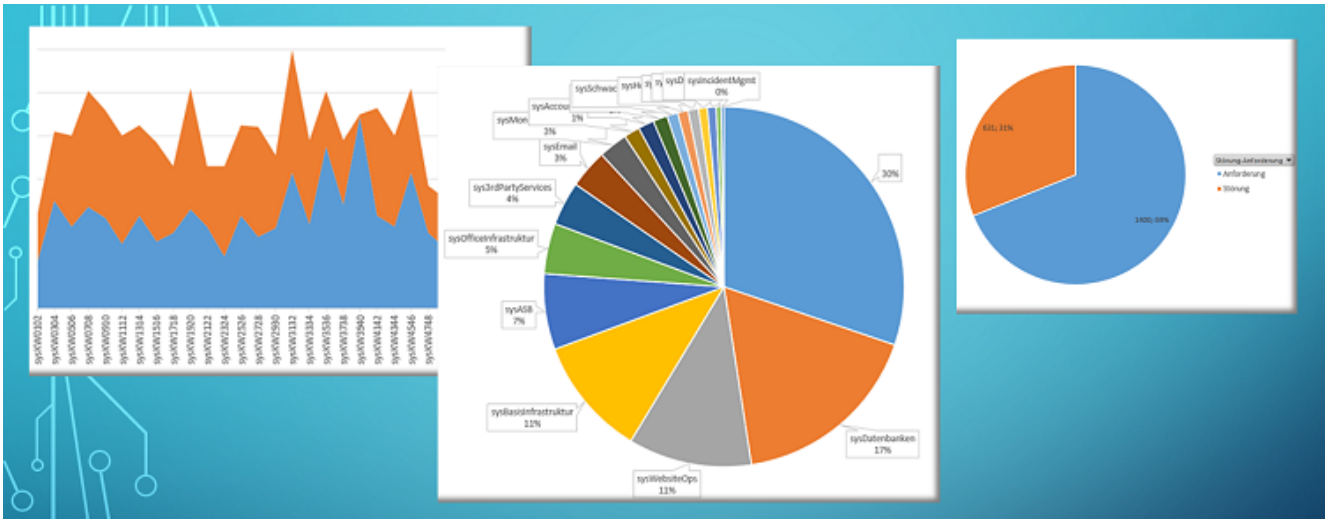


Anzahl der geschlossenen Tickets je Sprint in 2017

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…

Vom IT-Betrieb zu Platform Engineering. Ein Reisebericht (2/3)

Vor einer Woche habe ich berichtet, woher wir gekommen sind, welche Probleme wir hatten und wie wir mit Veränderungen wie z.B. der Einführung einer SOA-Architektur umgegangen sind.

Wir hatten schon einen wichtigen Schritt gemacht, indem wir unsere Arbeit sichtbar gemacht haben. Dadurch konnten wir priorisieren. Wir haben bewusst Themen abgemeldet bzw. erst für später eingeplant, um wiederum andere Themen mit einer höheren Priorität früher zu machen.

Wir hatten aber immer noch Probleme. Die Ticketanzahl stieg, wir waren immer noch zu langsam, und von Weiterentwicklung wollen wir gar nicht reden. Die Reise war also noch nicht zu Ende…

Kapitel 2: Basistechnologien, SelfServices & Automation

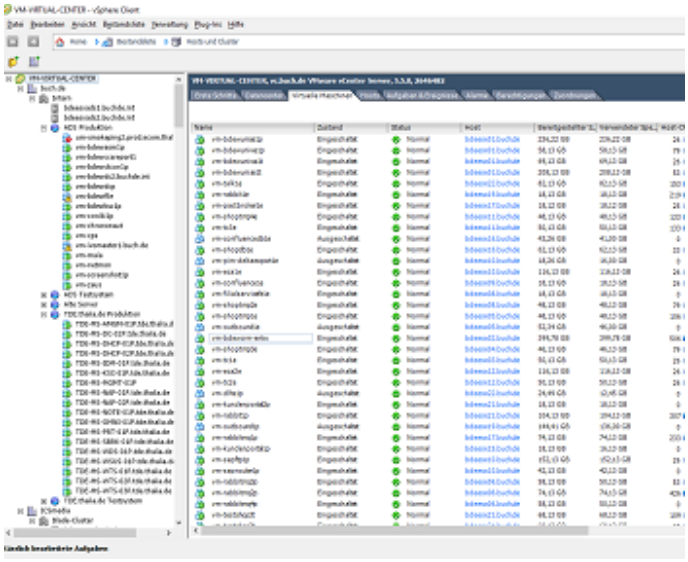
Wir wussten, dass wir nicht schnell genug waren. Die Anzahl der Tickets stieg und acht Wochen Vorlauf für einen neuen Service in Produktion klingt irgendwie nicht zeitgemäß, oder? An welchen Schrauben sollten wir drehen? Die Teams, die wir unterstützen sollen, warteten zu oft auf uns. Immer wieder kam z.B. die Aussage, dass ein Server in der Cloud in wenigen Minuten verfügbar ist. Wir als Team waren mit operativer Arbeit überlastet und hatten nur wenig Zeit für Weiterentwicklung. Irgendwie mussten wir im Tagesgeschäft schneller werden, um mehr Zeit für die Weiterentwicklungen unserer eigenen Themen zu bekommen. Wenn wir einen Mehrwert bieten wollen, dann müssen wir uns mit neuen Technologien auseinandersetzen. Nicht zuletzt machen neue Technologien und Weiterentwicklung auch mehr Spaß als das hoch standardisierte und wiederholte Einrichten von neuen Servern, Monitoring, Datensicherung, Logging, …..

Was fehlte uns, um schneller zu werden? Wir hatten das Glück, dass wir die Chance hatten, Einblicke in die Arbeitsweise und das Mindset von Otto.de zu bekommen (vielen Dank und viele Grüße an die Kollegen von Otto.de!). Ein Satz ist mir besonders hängen geblieben, weil ich ihn anfangs erschreckend schlimm fand. Kurz zusammengefasst: Kommunikation macht langsam! Das meinten die Jungs jetzt nicht ernst, oder? Gute Kommunikation ist doch die Basis! Klar habe ich es erst missverstanden. Gute Kommunikation ist natürlich sehr wichtig. Gemeint war damit die Idee, dass wenn zwei Teams miteinander reden müssen bzw. Themen ein Team verlassen, im nächsten Team bearbeitet werden und dann wieder zurück müssen, dort geprüft werden, Fehler gefunden, wieder zurück an das andere Team zur Nachbesserung, zurück zum Test, …… Kommt das

jemandem bekannt vor? Klingt das nach Topperformance? Solche Kommunikation an Schnittstellen macht einfach langsam. Versuche, Schnittstellen zu reduzieren. Versetze den Anforderer in die Lage, seine Anforderung selber umzusetzen, ohne Spezialisten-KnowHow zu haben. **Biete SelfServices an!** Ein SelfService basiert dabei auf einem Automaten mit einem einfachen Frontend. Über das Frontend ist der Anforderer in der Lage, seine wiederkehrenden Aufgaben einzugeben. Der Automat erledigt dann die hoch standardisierte Umsetzung der Aufgabe in einer fehlerfreieren Qualität als ein Mensch es könnte.

Nur wo fange ich an? Leichter gesagt als getan. Wir haben uns daran orientiert, wo wir als IT-Betrieb die größten Schmerzen hatten und wo wir den neuen Produkt-Teams den größten Nutzen bringen konnten. Hier drei technische Beispiele, die uns nach vorne gebracht haben.

Beispiel 1: Automatisierte Service Bereitstellung (ASB)

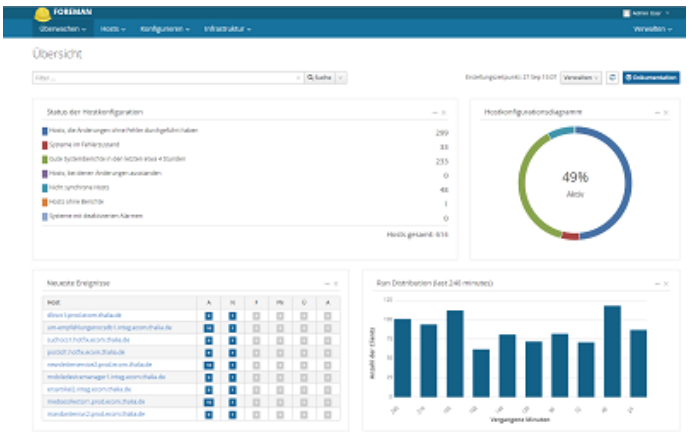


Vorher: Server klonen mit VMware

Wie ihr wisst, haben wir acht Wochen Vorlauf für einen Service in Produktion benötigt. Das muss natürlich auch schneller mit weniger Vorlauf gehen. Warum haben wir acht Wochen Vorlauf gebraucht? Primär aufgrund der Abstimmung zwischen IT-Betrieb und Entwicklung. Von den acht Wochen haben wir drei Wochen für die Abstimmung eingeplant. Eine Woche ist für den Bau der gesamten Infrastruktur eingeplant worden, so dass nach vier Wochen die Infrastruktur für den neuen Service bereitstand. Aber warum dann acht Wochen? Vier Wochen vor Produktion startet das erste Testdeployment. Hier muss die Infrastruktur fertig

sein, damit die Tests starten können. Auch in der Testphase ist immer wieder aufgefallen, dass die bereitgestellte Infrastruktur in sich nicht immer konsistent war. So waren Server trotz Standard VMware Image gerne mal unterschiedlich konfiguriert, es fehlten Berechtigungen oder Benutzer waren nicht angelegt. Um die Fehler in der Massenbereitstellung zu reduzieren, hilft nur eines wirklich: **Automatisiere!** Stelle Services automatisch und ohne Expertenwissen bereit.

Baue einen Automaten mit einem Webfrontend. Gebe dem Entwicklungsteam einen Knopf, auf den man drücken kann und aus dem dann nach wenigen Minuten ein Server rausfällt. Ein Server? Wer will den einen Server haben? Erweitere den Knopf, so dass nach der automatischen Server-Installation (das kann eine public Cloud eh besser) auch gleich noch das Betriebssystem passend für den **Service** konfiguriert wird, dann der Service installiert und konfiguriert wird und in Produktion gestellt wird. Das kann eine public Cloud für unsere Services „out of the box“ nicht -> Cool, Mehrwert generiert. Klingt das gut?



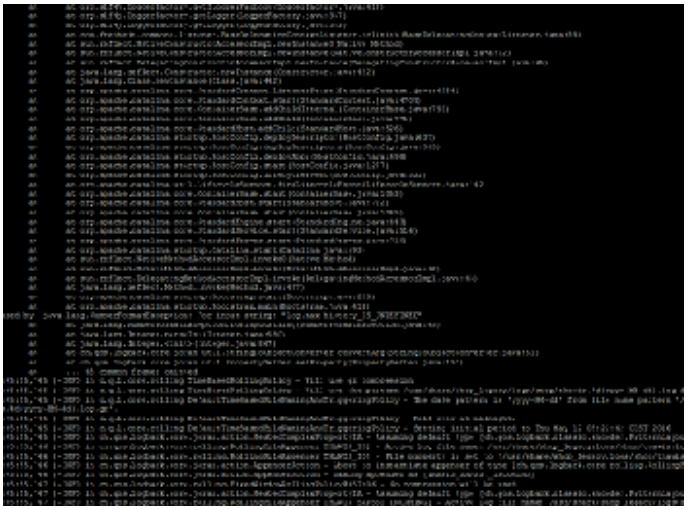
Nachher: Service Bereitstellung auf Foreman & Puppet Basis

Leute aus den Entwicklungsteams und dem IT-Betrieb haben also damit begonnen, basierend auf Foreman, Puppet und ein paar anderen Helferlein eine automatisierte Service Bereitstellung (Intern kurz: ASB) aufzubauen. Der Fokus lag im ersten Schritt auf dem Basis Betriebssystem und JAVA/Tomcat Services. Davon hatten wir die meisten, und es würde somit den größten Nutzen bringen. In dieser Zeit haben wir auch die Basis-Architektur für den Automatismus definiert und aufgebaut. Darauf basieren bis heute alle neu entwickelten Automatismen z.B. für DNS oder Apache Webserver. Wir haben einen starken Fokus auf den Aufbau der ASB-Infrastruktur gegeben, wodurch sicherlich auch

andere Arbeit liegen geblieben ist. Jedoch konnten wir nur so den Entwicklungsteams die Möglichkeit geben, Server inkl. Services ohne Rückfrage beim IT-Betrieb aufzubauen. Egal in welcher Stage. Konsequenz: Der SelfService wird so gut genutzt, dass wir heute über alle Stages weit über 1.000 Server betreiben.

Was haben wir damit erreicht? Wir haben die Server- & Service-Installation nun via einfaches Webfrontend an die Entwicklungsteams gegeben. Wir müssen keine Termine finden, es ist keine lange Abstimmung mehr notwendig, keine langen Rückfragen, keine Nachbesserung, keine acht Wochen Wartezeit… Der Aufwand für den IT-Betrieb und die Entwicklungsteams wurde stark reduziert, und die Durchlaufzeit für Server- und Service-Installationen wurde von Wochen auf wenige Minuten reduziert. Win/Win Situation!

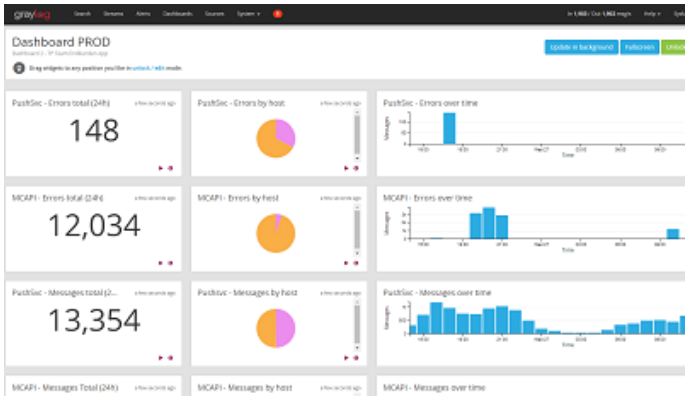
Beispiel 2: Zentrales Logdaten Management (ZLM)



Vorher: Logdaten Analyse auf der Console

OK, ein SelfService, um Services automatisch bereitzustellen, das ist schon mal super. Aber auch nur ein Baustein. Oft ist uns in der Vergangenheit das Logdaten Management System auf ELK-Stack-Basis (ELK = Elastic Search, Logstash und Kibana) unter der Last zusammengebrochen. Das lag weniger an der Technologie „ELK-Stack“ sondern eher an der eigenen Architektur, die, als wir sie vor Jahren initial für IT-Betrieb Systeme designed haben, nicht für die inzwischen vorliegenden Last ausgelegt war. Das verursacht wieder unfröhliche Stimmung in den Entwicklungsteams, weil wichtige Daten fehlen und das System nicht

verfügbar war. Gleichzeit hat der IT-Betrieb wieder ein neues, ungeplantes, dafür dringendes Ticket, um das alte System mit viel Klebeband wieder ans Laufen zu bringen. OK, lass deine Arbeit *jetzt* liegen, kümmere dich (schon wieder) um einen Störfall im Logdaten Management. Macht weder glücklich noch bringt uns die Störung nach vorne. Von schneller werden reden wir hier auch nicht. Aber es gibt ja nicht nur Störungen. SOA und Microservice sei Dank gibt es ja fast täglich neue Services, die auch in das Log-Management möchten. Mal abgesehen von den Kapazitäten im Logdaten Management System beginnt das traditionelle Spiel im Ticketsystem: Neues Ticket: „Ich benötige Logging“, klar baue ich dir gerne, so fertig, hmm ich sehe nichts, ups jetzt aber, OK funktioniert, aber warum kann ich nicht mehr Daten ins System geben, ….. aus den Büchern der menschlichen Schnittstellenkommunikation . Wie kann ich solche Kommunikation vermeiden? Baue einen **SelfService**! Plane und implementiere eine Logdaten Management Infrastruktur, die leistungsstark genug ist, um dein Volumen locker zu handhaben und wenig störanfällig ist. Sorge dafür, dass die Infrastruktur skaliert (nach oben und nach unten). Stelle eine gut dokumentierte Schnittstelle bereit, über die ein Entwicklungsteam selber seine Daten in das Logdaten Management schreiben kann und die Möglichkeit hat, die Daten selber auszuwerten.



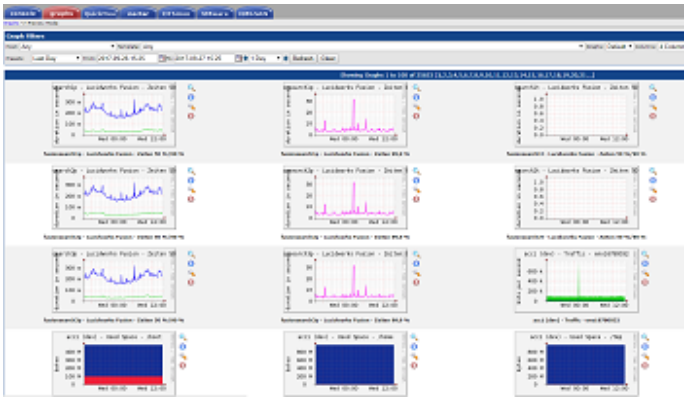
Nachher: Logdaten Management auf Graylog Basis

Auch den Aufbau eines neuen und leistungsfähigen Logdaten Managements konnten wir recht früh starten. Auch hier setzen wir auf das bewährte Prinzip, Entwicklerteams und IT-Betrieb zusammenzubringen. Schnell wurde aber klar, dass es nicht besonders schlau ist, alle Logdaten, die es gibt, ohne Sinn und Verstand in eine Datenbank zu pressen. Das ist zwar möglich, aber nicht schlau. So haben wir uns u.a. mit den Fragen auseinandergesetzt „Was und wie loggen

wir überhaupt?“, „Wann brauchen wir eine Logausgabe?“ und „Was machen die Logdaten?“. Nach vielen Gesprächen über Anforderungen und Optionen haben wir einen Vorschlag erarbeitet, wie die Entwicklungsteams Logging in ihren Services nutzen können. Parallel dazu haben wir eine neue Logdaten Management Infrastruktur basierend auf Graylog aufgebaut. Graylog bietet zusätzlich zum ELK-Stack noch ein paar nette Features wie z.B. ein User Management. Auch erschienen uns das Handling und der Betrieb im Vergleich zum ELK-Stack etwas einfacher. Gesized wurde die neue Infrastruktur für Spitzenlastzeiten. Bei uns ist das neben dem Schulbuchgeschäft natürlich das Weihnachtsgeschäft. So kamen wir auf eine Infrastruktur von u.a. zwei Elastic Search Master Nodes und acht Elastic Search Data Nodes in zwei Datacentern. Durch die Data Nodes ist die Infrastruktur jederzeit ohne viel Aufwand skalierbar. Im Peak kann die Infrastruktur bis zu 100.000 Messages die Sekunde verarbeiten. Um die künftigen Nutzer der Infrastruktur zu informieren und möglichst früh Feedback zu bekommen, haben wir zum Thema eine von uns so getaufte „Bier-Session“ angesetzt. Eine „Bier-Session“ ist in etwa vergleichbar mit einer Brownbag-Session, jedoch war es nicht in der Pause und es gab (alkoholfreies) Freibier. Das Feedback haben wir gerne aufgenommen und in die Infrastruktur eingebaut.

Was haben wir mit der neuen Infrastruktur erreicht? Die Produkt-Teams können nun ihre eigenen Anwendungen durch eine einfache Konfiguration in der „logback.xml“ anbinden. Auch die Dashboards zur Auswertung der Logdaten können sich die Teams ohne Unterstützung des Platform Engineering Teams einrichten. Ein weiterer schöner SelfService, der die Aufwände im IT-Betrieb reduziert und die Arbeit in den Produkt Teams beschleunigt.

Beispiel 3: Automatisiertes Monitoring



Vorher: CACTI Monitoring hat seinen Dienst getan

Monitoring war auch so eine super Sache. Wir monitoren ja echt viel – mit einigen unterschiedlichen Monitoring- & Alarmierungssystemen bekommen wir einen super Überblick über den Zustand unserer Services aus technischer Sicht und aus Kundensicht. Jedoch mit was für einem Aufwand! Sämtliche Sensoren wurden von einem Menschen (immerhin haben wir für das Monitoring einen dedizierten Menschen) manuell in die Monitoring Systeme eingetragen. Also erstellt das Entwicklungsteam ein Ticket für das Monitoring, da fehlen aber noch Information im Ticket, wieso fehlen da noch Information, ich erkläre es dir, jetzt habe ich ein Ticket mit allen Information, die Sensoren wurden eingebaut, da fehlt aber noch ein Sensor, OK, warum hast du das nicht gleich gesagt, nun ist der Sensor auch eingebaut, aber wo ist der Aktionsplan für die Alarmierung, der kommt später, …. Hatten wir das mit den Schnittstellen nicht schon mal? Gibt es eine Lösungsidee? Klar: Bau eine **SelfService** Schnittstelle! Im gleichen Zuge, in dem wir die SelfService Schnittstelle gebaut haben, haben wir auch unser solides, jedoch in die Tage gekommenes CACTI durch eine moderne Lösung abgelöst. Von der neuen Lösung versprachen wir uns mehr Möglichkeiten, Dashboards, Graphen, Informationen etc. bedarfsorientiert zu visualisieren, um dadurch schneller und einfacher entstören und planen zu können.

Nachher: Performance Monitoring mit InfluxDB & Grafana

Dank der frühen Planung und der Rahmenbedingungen, die geschaffen wurden, konnten wir auch damit beginnen, einen weiteren SelfService für das automatisierte Monitoring zu bauen. Möglich wurde das u.a. dadurch, dass wir uns externe Unterstützung ins Team geholt haben, die uns den Rücken im Tagesgeschäft frei gehalten hat, so dass die internen Mitarbeiter sich um die Entwicklung der neuen Technologie kümmern konnten. Das klingt alles super und hat uns sehr geholfen, ein Überschuss an Mitarbeitern war aber dennoch nicht zu erkennen. Wir haben einige Themen parallel bearbeitet. Doch wir haben das Beste daraus gemacht. Und so kam es z.B., dass wir u.a. einen unserer Datenbank Spezialisten überzeugen konnten, zusammen mit einem Monitoring-Spezialisten und Software-Entwicklern ein neues Performance Monitoring System basierend auf InfluxDB, Telegraph und Grafana zu bauen. Die Arbeiten wurden stark unterstützt von Produkt-Teams, die ihre Ideen und Anforderungen eingebracht haben. Nachdem der erste brauchbare Prototyp stand, haben wir vor der Pilotphase eine weitere „Bier-Session“ durchgeführt. Die Session hat allen Nutzern einen frühen Blick auf das künftige System gegeben und uns gutes Feedback gegeben, welches wir einarbeiten konnten.

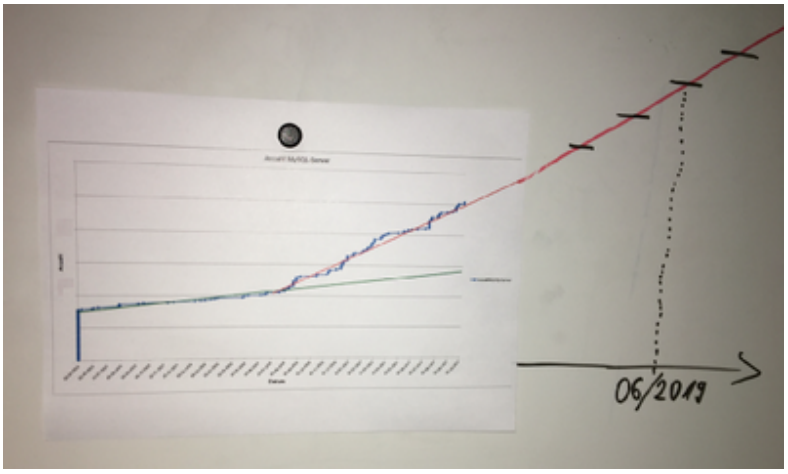
Ein neues Performance Monitoring hilft aber nur in Teilen weiter. Es ist leistungsstark, voller Features und kann ohne Hilfe des IT-Betriebs genutzt werden. Jedoch fehlt noch eine Lösung für unser Alarmierungssystem und dessen Aktionspläne. Daher haben wir zusätzlich zusammen mit Software Entwicklern, Qualitätstestern und Administratoren einen komplett neuen Service gebaut, der es erlaubt, aus Informationen aus einer YAML-Datei die Sensoren in der Alarmierung sowie dem Performance Monitoring zu erstellen und zu konfigurieren. Dadurch haben wir den Entwicklungsteams eine Schnittstelle gegeben, die sie aus ihrem Tagesgeschäft kennen, haben die Monitoring

Konfiguration zu Code gemacht und versioniert, die manuellen Aufwände reduziert und die Qualität erhöht. Klingt nach einem coolen SelfService? Ist er auch!

Was haben wir mit den Änderungen an den Basistechnologien erreicht?

Wir haben eine Menge Arbeit in die Bereitstellung von SelfServices und Automaten investiert. Dadurch haben wir es geschafft, die aufwändige Abstimmung zwischen Entwicklungsteams und IT-Betrieb zu reduzieren. Wir haben durch Automation und technische Schnittstellen die Qualität in der Umsetzung von Anforderungen erhöht und den Bedarf für Nacharbeiten reduziert. Dadurch haben wir einen wichtigen Betrag zur Beschleunigung der Entwicklungsteams geleistet. Die Aufwände für wiederkehrende Arbeit wurden im IT-Betrieb reduziert, wodurch mehr Zeit für Weiterentwicklung entstanden ist.

Gibt es auch Schattenseiten?



Wachstum DB-Systeme der letzten 2 Jahre inkl. „Prognose“

Natürlich ist nicht immer alles rosarot, und auch wir haben durch Schmerzen gelernt. So wird z.B. unser SelfService zur automatischen Service Bereitstellung sehr gerne und viel genutzt, was erst einmal sehr positiv ist. Hier ist ein Beispiel, wie sich die Anzahl unserer Datenbanksysteme über die letzten 2 Jahre geändert

hat und welches Wachstum wir noch erwarten. Ratet einfach mal wann die automatische Service Bereitstellung verfügbar war Eines der verbundenen Probleme ist, dass die Anforderung an alle Komponenten der Plattform (Beispiel: Storage, CPU, RAM, Virtualisierung, …) deutlich gestiegen sind. Auch die Anforderungen an die für den Betrieb notwendigen Systeme (Beispiel: Datensicherung, Monitoring, Logging, …) sind davon nicht ausgenommen. Natürlich haben wir mit einem spürbaren Anstieg der Systeme und somit auch mit den Anforderungen gerechnet. Dennoch wurden wir ein Stück vom Erfolg überrascht. Fehlende Ressourcen in der Plattform haben zu Störungen in den Services geführt, was wir nachträglich im laufenden Betrieb mit Schmerzen in allen Teams korrigieren mussten. Auch die steigenden Kosten sowie die Kostenkontrolle für Plattform Ressourcen müssen bei der Einführung und dem Ausbau von SelfServices berücksichtigt werden. Aktuell verrechnen wir die Kosten nicht an die Teams, jedoch stellen wir über Reporting sicher, dass jedes Team weiß, was ein System kostet und welche Kosten für Plattform Ressourcen ein Team produziert.

Kapitel 3: Mindset, Methoden, Prozesse und mehr…

In einer Woche werde ich im dritten und vorerst letzten Kapitel unserer Reise darüber berichten, dass neben der Technologie auch kulturelle und methodische Änderungen notwendig waren. Z.B. war unser Mindset das eines klassischen IT-Betriebs. Das Mindset unserer wichtigsten Kunden, der Entwicklungsteams, war jedoch das einer agilen Software-Entwicklung. Und was ist eigentlich dieses DevOps? Hier gab es also noch etwas zu tun. Abgerundet wird das dritte Kapitel durch ein Fazit, was wir gelernt haben. To be continued …

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…