Scrum mit User Stories

Ralf Wirdemann und Dr. Johannes Mainusch

9. August 2016

Kapitel 3

Verticals - SCRUM@OTTO

3.1 Einleitung - warum ich über diese Geschichte schreibe

Als ich im Mai 2012 bei der Otto GmbH & Co. KG[1] in Hamburg meinen ersten Arbeitstag begann, war ich der festen Überzeugung, ich könne mit meinem reichhaltigen Know-how als agiler Manager, mit meinen Jahren bei dem sehr erfolgreichen Sozialen Netzwerk XING der Otto-Organisation helfen, Erfolg zu haben. Bei XING haben wir Scrum in 2008 im ersten Projekt sehr erfolgreich eingeführt und seitdem in immer mehr Teams verwendet. 2009 kam Kanban als Methode in betriebsnahen Projekten hinzu. 2012 dachte ich, alles über agile Methoden zu wissen. Weit gefehlt.

Nach außen hin, sah die Otto-IT immer etwas langweilig aus. Das Unternehmen hat in Hamburg den Nimbus, sehr solide aber eben auch etwas behäbig zu sein. Die Performance des E-Commerce Shopsystems war schlecht, und alles wirkte etwas verstaubt. Meinem neuen Chef sagte ich auf die Frage, was ich denn als meine Aufgabe sähe, ich habe als Ziel, die coolste IT-Organisation nördlich der Donau dort zu schaffen [2]. Kurz, ich kam mit viel Bravado, einer gehörigen Portion Arroganz und mit viel Tatendrang in das gerade aufgesetzte Lhotse-Projekt[3] bei Otto.

Tatsächlich habe ich in meinen zwei Jahren bei Otto viel mehr gelernt, als erwartet. Dachte ich vorher, nicht mehr über Agilität lernen zu können, so wurde ich bei Otto schnell bescheidener. Ich fand engagierte Kollegen vor, Experten, Menschen mit ausgeprägten Meinungen, ein gut organisiertes großes Projekt, Projektmanager, die ihre Teams und ihr Vorhaben mit flammender Begeisterung verteidigten, Erfolg und Freunde bis heute. Für mich war Otto und im Besonderen das „Lhotse-Projekt", der komplette Austausch der E-Commerce-Plattform durch eine Eigenentwicklung, ei-

[1] Ich war bei der Einzelgesellschaft Otto GmbH & Co. KG angestellt. Wenn ich im Text von Otto spreche, dann meine ich damit immer die Otto GmbH & Co. KG mit etwa 4500 Mitarbeitern in Hamburg und dem zugehörigen E-Commerce shop otto.de

[2] südlich der Donau ist Google in München und Zürich, das wäre ja vermessen

[3] Lhotse - benannt nach dem vierthöchsten Berg im Himalaya - war der Projektname für die Ablösung des in die JAhre gekommenen otto.de Shopsystems durch ein eigenentwickeltes neues System

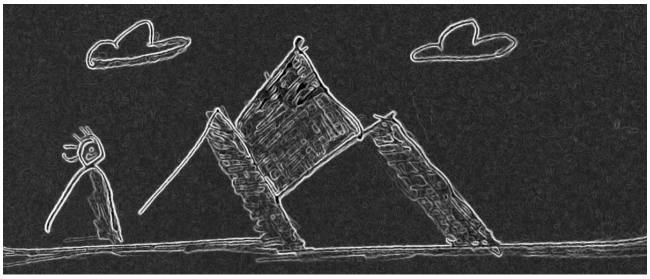


Abbildung 3.1: Lhotse ist nicht nur einer der höchsten Berge der Welt, sondern Lhotse war auch der Name der Projektes, in dem 2012 - 2013, das komplette Shopsystem von otto.de auf ein eigenentwickeltes E-Commerce System umgestellt wurdet

nes meiner beruflichen Highlights. Und heute kann ich sagen, bei Otto ist wirklich die Produktentwicklung auf einem Stand, den ich nördlich der Donau bislang besser nicht kenne.

Ich schreibe darüber, wie wir es mit dem Lhotse-Projekt (Abbildung 3.1) geschafft haben, einen der größten E-Commerce Shops Europas (Abbildung 3.2) erfolgreich durch Eigenentwicklung von einem schlechten Alexa-Ranking >50 in 2012 auf heute Platz 24 (Mai 2016, Deutschland) zu bringen, die Perfomance um über 30% zu verbessern, die Kundenzufriedenheit entsprechend zu erhöhen.

Und ich berichte über die Art, wie wir bei Otto Scrum als agile Methode verwendet haben. SCRUM@OTTO ist Scrum mit einigen Ergänzungen, über die bislang wenig geschrieben wurde, die aber für unseren Erfolg sehr wichtig waren. Dabei lerne ich Dinge kennen, die mir im agilen Kontext bis dato fremd waren. Triaden, Vertikale, Technical-Designer und die erfolgreiche Einbindung eines übergreifenden Portfolio-Managements, all das kannte ich vor 2012 nicht.

In dieses Buch passt das gut, denn der Austausch von otto.de durch eine Eigenentwicklung war der Start für die vielleicht größte und erfolgreichste agile Transition eines Konzerns im deutschsprachigen Raum.

User Stories spielten dabei eine große Rolle, Story-Mapping hingegen war 2012 noch kein sehr verbreitetes Thema. Die bei Otto.de gefundenen Vertikalen Schnitte ergäben sich jedoch sofort aus einem Story-Mapping-Prozess.

3.2 Die Vorgeschichte

Otto verkaufte anfangs Schuhe. Das war im Jahr 1950. Der Knüller war damals, dass Otto einen Katalog der Schuhe mit Bildern und Beschreibungen hatte. Und, dass Kunden mit dem Produkt eine Rechnung bekamen. Das heißt, die Bezahlung erfolgte erst nach dem Empfang des Produktes. Zum Einkauf musste man erstmals das Haus nicht verlassen. Mit dieser Idee wurde Otto in den nächsten 50 Jahren ein weltgroßer Konzern. Viele aus der Generaation X erinnern sich heute noch an die schönen Kataloge aus ihrer Kindheit, die viele interessante Dinge enthielten, die das Bedürfnis nach ,,haben wollen“ hervorriefen.

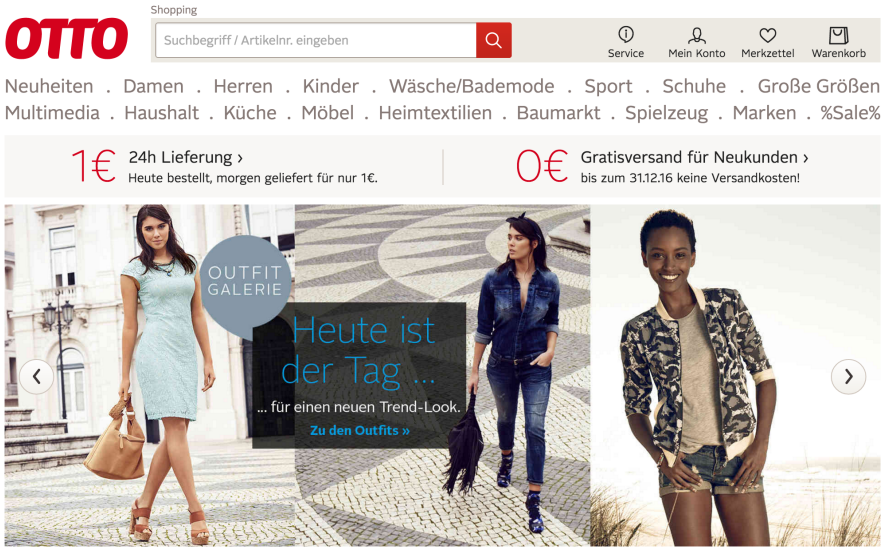


Abbildung 3.2: Die Otto.de Seite 2016

Als um das Jahr 2000 das Internet für Firmen immer bedeutsamer wurde, war Otto mit dabei. Im Jahr 2003 wurde dann sogar eine strategische Partnerschaft mit dem E-Commerce Hersteller Intershop geschlossen und die Verkäufe über das Internet stiegen in den darauf folgenden Jahren stetig an. Im Jahr 2012 wurden schließlich ca. 80% aller Verkäufe über das Internet getätigt. Damit wurde Otto einer der größten E-Commerce Shops Europas. Nach außen ein riesiger Erfolg.

Doch es gab auch damals schon deutliche Signale, die in eine andere Richtung zeigten. Dazu gehörten die schlechte Antwortzeit des E-Commerce Angebots, das Kunden zwischen zwei Klicks über drei Sekunden warten ließ. Dazu gehörte eine große Mannschaft von meist externen Produktentwicklern, die in mühevoller Arbeit den in den letzten 10 Jahren gewachsenen IT-Monolithen in zweiwöchigen Releases auf die neuesten Kundenbedürfnisse anpassten. Trotz eines der ausgefeiltesten Multiprojekt-Managements war die Entwicklung sehr träge und kleinste Änderungen waren stets mit hohem Risiko behaftet. Server ächzten unter der Last der Benutzer.

In der Vergangenheit hatte Otto schon zweimal versucht, mit einem Update auf die neueste Version des verwendeten Shopsstems die Situation zu verbessern. Solche Releasewechsel sind eigentlich normal und verlangen auf einem Handy heute nur einen Klick. In großen Unternehmen hingegen sind sie oft Projekte mit mehr als 12 Monaten Laufzeit.

Zweimal war Otto am Versuch, die Systeme zu erneuern gescheitert. Ein anstehendes Upgrade auf die neueste Intershop Version wurde als hohes Geschäftsrisiko und als mehrjähriges Projekt mit einem Volumen über viele Millionen Euro eingestuft. Kurzum, der Karren war ganz schön tief in den Schlick gefahren.

Start:	Herbst 2011
Going Live:	24.10.2014
bis dato(2016):	Weiterführung der Produktentwicklung
Personal:	4 Teams à 6-10 Personen in 2012
	8 Teams à 8-23 Personen 2013
	12 Teams à 10-15 Personen in 2016
Softwareentwickler:	ca. 60 am Ende des Projektes in 2014
	ca. 110 in 2016
Gesamte Personalstärke:	ca. 300 inkl. Produktmanagement,
	Projektmangement, Controlling, Managment,
	Operations, ...
Web-Performance:	¡2 Sekunden zwischen Klick und Auslieferung der Seite
Hits pro Sekunde:	2014 gebaut für 500/s
	Highscore (2015) 1337/s (this is not a nerd joke)

Tabelle 3.1: Eckdaten zur Neuentwicklung von otto.de im Lhotse Projekt

In dieser Situation entstand zwischen den Managern der E-Commerce Abteilung die gemeinsame Idee, es mit einer Eigenentwicklung zu versuchen. Ermutigt von einem funktionierenden selbstgebauten Prototypen entstand die Idee des Lhotse-Projektes. Benannt, nach einem der höchsten Berge im Himalaya. Die am Plan beteiligten Manager waren interessanterweise für das Produkt-Management, das Programm-Management und die IT verantwortlich. Der Plan des Lhotse-Projektes entstand und auf Managementebene erfolgte ein bislang ungesehener Schulterschluss. Ein wesentlicher Erfolgsfaktor ist genau darin zu suchen, dass irgendwann in 2011 bei Otto einige Manager das Kriegsbeil begruben und begannen, gemeinsam zu arbeiten.

3.3 Das Lhotse-Projekt - Zahlen, Daten, Fakten

Einige Zahlen, Daten und Fakten zum Lhotse-Projekt bei Otto finden sich in der Tabelle 3.1.

Es gibt einige Besonderheiten dieses Projektes, die mir vorher so noch nirgendwo begegnet sind und die in den nächsten Abschnitten behandelt werden. Dazu gehören:

- Die Otto-Architektur in Vertikalen
- Triaden - das Produktmanagement eines Teams
- Die Rolle des Team-Architekten (Technical Designer)
- Live-Deploment in 10 Minuten

Weiterhin ist erwähnenswert, dass es neben dem sehr prominenten Lhotse-Projekt ein fast ebenso großen Projekt zur Auslagerung der IT-Operations aus dem hauseigenen Rechenzentrum in Hamburg-Wandsbek in zwei Co-Location Rechenzentren im Hamburger Umfeld gab. Heute läuft der gesamte Shop auf einer virtualisierten Umgebung. Im ersten halben Jahr 2014 hat die Otto-Operations Truppe mehr als 70.000 virtuelle Maschinen abgerissen und wieder aufgebaut.

Kommunikation ist wertvoller als Infrastruktur.
Server und Infrastuktur dürfen daher in einem erfolgreichen Projekt niemals zum Engpass werden.

Seit dem jüngsten Siegeszug der Virtualisierung stellt die Verfügbarkeit von Servern im Gegensatz zu früher keine Limitierung mehr dar.

3.4 Das Team - Menschen im Mittelpunkt

Im Mittelpunkt der Produktentwicklung bei otto.de steht das Team. Waren die anfangs vier Teams am Projektstart von Lhotse mit etwa 8 Teammitgliedern noch relativ klein, so wuchsen sie schnell an. Anfang fast reine Entwicklungsteams mit etwas Produktmanagement und einer zu Beginn noch zentral agierenden Qualitätssicherung wurden schrittweise größer. Nicht nur kamen mehr Softwareentwickler hinzu, es wurden auch mehr Funktionen in die Teams integriert. Zunächst die Qualitätssicherung Mitte 2013. Dann Usability und Design, teilweise Operations und weitere Ergänzungen im Produktmanagement. Zwei der Teams hatten zeitweilig über 20 Mitglieder und ächzten unter der Last ausgiebiger Meetings und umständlicher Kommunikation, bevor sie sich durch Team-Splits verkleinerten.

Als Grundregel für die Bildung von Teams wurde verankert, dass jedes Team mit seiner Arbeit direkten Kundenkontakt haben sollte. Das führte konsequenterweise zu einem vertikalen Schnitt der Team-Verantwortungen. Dazu wurde otto.de in Bestandteile entlang der User-Journes aufgeteilt. Diese Bestandteile waren zunächst wie in Abbildung 3.3 dargestellt,

- Suchen,
- persönlich Empfehlen,
- Benutzer identifizieren,
- Bewerten[4],
- Kaufen.

Die Methode der Wahl war 2012 Scrum mit einigen Adaptionen. Das wurde dann SCRUM@OTTO genannt. Alle Teams starteten im selbem Sprint-Rhythmus. Zweiwöchige Sprints mit täglichen Standup-Meetings, Sprint Planning 1 und 2, ei-

[4] Anfangs waren Suche und Bewerten in einem Team. Ebenso Kaufen und Benutzer identifizieren.

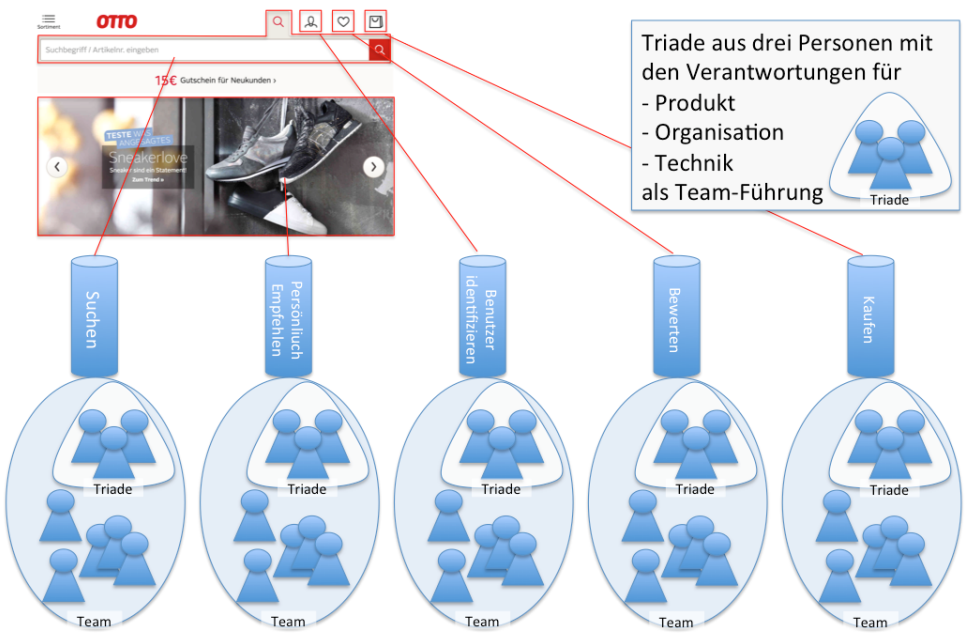


Abbildung 3.3: Die Otto Teams und er vertikale Schnitt durch den otto.de Webshop.

nem Review, was häufig in einem Messe-Format [5] stattfand und der Retrospektive. Dazu kamen jede Woche ein Estimation-Meeting für Dinge, die meist in fernerer Zukunft lagen.

Besonders war, dass die Teams im Gegensatz zu herkömmlichen Scrum-Teams mit drei Führungskräften besetzt waren. Nicht Scrum-Master und Product Owner steuerten das Team, sondern eine sogenannte „Triade“ aus Projektleiter, Produkt-Designer und Technical-Designer. Die Rolle dieser Personen und das Zusammenspiel wird im nächsten Abschnitt genauer erklärt.

Heute arbeiten nur wenige der Teams noch nach dem selben Sprint-Rhythmus wie 2012. Die meisten Teams haben nach und nach ihren Arbeitsprozess angepasst. Das gab zwar anfangs Diskussionen und erzeugte Unsicherheit im Management, letztendlich wurde hier aber dem agilen Gedanken gefolgt, dass jedes Team selbst am besten über seine Arbeitsweise entscheiden kann. Schritt für Schritt löste sich somit auch das klassische Otto-Management von der Tradition, den Arbeitsprozess in Mikromanagementschritten vorzugeben.

Insgesamt sind die Teams bei Otto heute sehr eigenständig unterwegs und erreichen damit eine herausragende Geschwindigkeit und Eigenverantwortung.

[5] Beim Messe-Format stellen zwei Personen aus allen Teams ihre Sprintergebnisse gleichzeitig vor. Alle Besucher der gleichzeitig stattfindenden Reviews können nun wie auf einer Messe herumgehen und die verschiedenen Ergebnispräsentationen besuchen.

Schnelle Expertenteams tragen Verantwortung, treffen selbst Entscheidungen auch in kritischen Situationen und bestimmen selbst darüber, wie sie arbeiten.

3.5 Triaden - die Führung eines Teams

Die Triade ist die Kommunikations-Schnittstelle aus den Teams nach außen in die anderen Teams und in die Organisation. Weiterhin organisiert sie die Aufgaben im Team.

Nur zusammen mit dem Projektmanager, dem Produktmanger und dem Technical Designer ist die Triade, die Führungsmannschaft eines Teams bei Otto vollständig. Dieses Teilen von Verantwortung auf einer Führungsposition ist für klassische Projektleiter ungewohnt. Bei Otto bekamen die Triaden zu Beginn ihrer Arbeit eine umfängliche Triadenschulung. Dazu wurde in Zusammenarbeit mit einem externen Coach und dem Otto-Management eine halbtägige Schulung aufgesetzt, in der die Mitglieder der Triade unter Anderem mit ihren Rollenbeschreibungen (siehe Kapitel 3.6.1, 3.6.2 und 3.6.3) und den Leitplanken (siehe Kapitel 3.10), also mit ihrem neuen Job vertraut gemacht wurden.

Und auch während der laufenden Produktentwicklung wurden die Triaden regelmäßig von einem ausgebildeten Coach begleitet.

Das Konzept der Triade soll dem im Gesamtprojekt hohen Kommunikationsbedarf in den drei Richtungen Methode & Portfolio, Produkt und Technik gerecht werden. Die beiden Rollen Projektmanager und Produktmanager sind mit Scrum Master und Product Owner in Scrum schon ähnlich angelegt. Besonders hervorzuheben ist die Rolle des Technical-Designers. Im Laufe des Lhotse-Projektes wurde klar, wie bedeutend gerade diese Rolle für eine gut funktionierende Architekturorganisation ist. Nur durch die Technical-Designer und die weiter unten vorgestellte TD-Runde konnte sich die vertikale Architektur bei Otto etablieren.

Bis die Triaden sich in den Teams zusammengerauft hatten, vergingen in der Regel einige Monate. Auch sie durchlaufen den Tuckman-Zyklus[6]. Danach wurden sie dann ein sehr gut kommunizierendes und sich ergänzendes Führungsteam, das in der Regel unter großer Auslastung arbeitete. Es lohnte sich also, den Triaden und den ebenfalls recht großen Teams bei Otto die nötige Zeit zur Findung einer guten Zusammenarbeit einzuräumen.

Gute Performance braucht Zeit:
Ungeduldiges und unnötiges Umbauen von Teams ist zu vermeiden. Denn mit jedem Umbau eines Teams muss der Tuckman-Cycle erneut durchlaufen werden.

[6] https://en.wikipedia.org/wiki/Tuckman's_stages_of_group_development

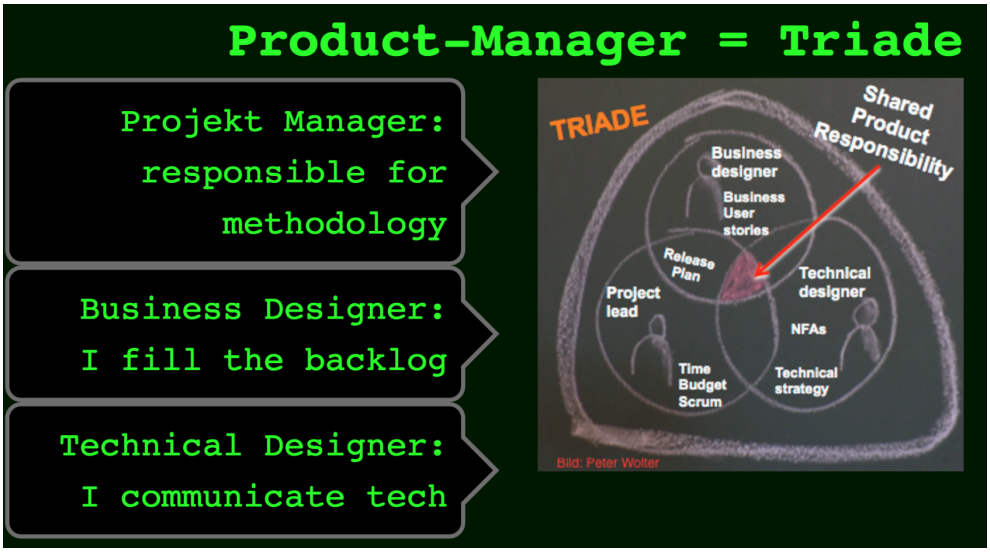


Abbildung 3.4: Die Triade zur Führung eines Teams vereint methodische Kompetenz, Produktkompetenz und technische Kompetenz.

3.6 Die Triade - Rollenbeschreibungen

Eine der Managementaufgaben bestand darin, die Rollenbeschreibungen für die Triade vorzunehmen. Neben den unterschiedlichen Aufgaben wurde darauf geachtet, Kooperationsziele aufzunehmen und auf Überschneidungen hinzuweisen. Während in klassisch aufgebauten Organisationen Rollen immer disjunkt, also Überschneidungsfrei, definiert werden, stellte sich bei Otto heraus, dass der Fokus auf Zusammenarbeit und geteilte Verantwortung zu weniger Verantwortungslücken führte. In den folgenden drei Abschnitten werden die Rollen des Projektmanagerin, des Produktmanagers und der Technical-Designerin beschrieben. Grob entspricht das den Verantwortungen für Methodik, Produkt und Technologie.

3.6.1 Der Projektmanager - Project-Lead

Der Projektmanager(im OTTOspeak Project-Lead oder PL) ist die aus dem klassischen Projektmanagement verbliebene Rolle. Allerdings mit einigen Änderungen. Als Projektmanager in einem Team ist man Teil der Triade, das heißt, die Highlander-Regel aus der PMI-Au sbildung, „es kann nur einen geben“, gilt nicht mehr. Die Aufgaben der Projektmanagerin sind:

- die methodische Verantwortung im Team analog zu einem SCRUM-Master
- das Sammeln von Informationen für das übergeordnete Programm- oder Portfoliomanagement. Etwa die Ergebnisse vor Zeit und Aufwandsschätzungen des Teams.

- das Lösen von organisatorischen Abhängigkeiten zu anderen Teams. Dazu können beispielsweise
- das Management von übergreifenden Abhängigkeiten, die sich nicht im Team lösen lassen. Etwa: Es wird eine Experte aus einem anderen Team benötigt oder es gibt zu wenig Meetingräume während des Sprintwechsels.
- die Kommunikation der Sprint-Ergebnisse nach außen. Beispielsweise die Organisation von teamübergreifenden Reviews zum Austausch von Arbeitsergebnissen.

3.6.2 Der Produktmanager - Business-Designer

Der Business-Designer (oder BD, Otto.de Terminologie) ist analog zum Produktmanager in Scrum für die Fachlichkeit zuständig. Er ist die Nahtstelle zum Kunden und zu den Nutzern des erstellten Produktes. Er schreibt User-Stories, priorisiert sie und befüllt das Backlog.

Bei Otto.de hatten die Business-Designer eine sehr stark auf die Nutzer ausgerichtete Verantwortung. Die Produktabteilung hatte eigene Experten für Search-Engine-Optimization (SEO), User-Research und UX-Design. Dementsprechend waren oft mehr als eine produktverantwortliche Person einem vertikalen Team zugeordnet. Die Koordination dieser Produkt-Experten oblag ebenfalls dem Business-Designer.

3.6.3 Der Team-Architekt - Technical-Designer

Die Rolle des Technical-Designer (TD) oder Team-Architekten in der Triade besteht vor allem darin, Architekturkonformität und Softwarequalität über die Teams hinweg sowie die Betriebsfähigkeit des erstellten Produktes in der Zukunft zu sichern. Im Folgenden wird die Rolle des TD in Form von messbaren Zielen und Aufgaben beschrieben. Aufgabenbereiche des TD.

- Gemeinsame Verantwortung für das Produkt mit dem Business Designer (fachlicher Produktmanager) und dem Project Lead (Projektmanagement und Methodik) und dem gesamten Team. Dabei ist die Triade aus BD, PL und TD für die Kommunikation nach außen und die Organisation nach innen verantwortlich.
- Strategische technische Ausrichtung des Produkts (Technische Marktbeobachtung, Impulsgeber für technische Innovationen im Team, solide Einbindung in die vorhandene technischen Systeme bei Otto)
- Betriebsverantwortung (Sicheerstellen reibungsloser technischer Betrieb und Zusammenarbeit mit der Operations-Abteilung)
- Technische Verantwortung (Einbringen der technischen Sicht in die fachlichen Produktentwicklungsprozesse und Verantwortung die technische Konzeption und Umsetzung von User-Stories)
- Technische Qualitätsverantwortung (Sicherstellen eines effizienten QS-Prozesses in der agilen Softwareentwicklung und der Zusammenarbeit mit zentralen QS-Verantwortlichen.)

Jedes Team bei Otto hat einen TD. Der TD koordiniert für das Team die technischen Aufgaben im Backlog und vertritt das Team nach außen in der TD-Runde, dem regelmäßigen Treffen aller Architekten.

Eine exemplarische Rollenbeschreibung des TD folgt im grauen Kasten:

Rollenbeschreibung Team-Architekt
Der Team-Architekt (TA) hat die Aufgabe eine bessere Verzahnung der Kommunikation zwischen den Teams sowie zwischen den Teams und den [Enterprise-, Zentral- Firmen-, Sicherheits-] Architekten zu erreichen. Wir wollen mit seiner Unterstützung gemeinsam die Regeln (Mikro- und Makro-Architektur und Leitplanken) erarbeiten, von denen wir glauben, dass der TA eine optimale Zusammenarbeit mit vielen Teams und noch viel mehr Komponenten gewähren. Daraus leitet sich für uns der hohe Anspruch nicht nur an die kommunikativen und vermittelnden Fähigkeiten des TA, sondern auch an sein integratives zukunftsorientiertes Handeln ab.
Technische und architekturelle Entscheidungen im Team beeinflusst der TA und wägt die Vor- und Nachteile ab. Alle TAs zusammen bilden ein übergreifendes Architekturboard. Der TA hat Ideen und vertritt die im Team abgestimmte Meinung. Dadurch wird für eine bessere Verzahnung zwischen den Teams und allen anderen architekturbeitragenden Parteien (IT-Sicherheit, QA, Operations, DEV-OPS) gesorgt.
Kernaufgaben:
• Kommunikationsschnittstelle zwischen dem eigenen Team und den anderen Teams
• Ist bei übergreifenden Architekturthemen Kommunikator und Unterstützer
• unterstützt bei der Definition der technischen Rahmenbedingungen und Leitplanken, stimmt sie mit dem Team ab und stellt die Einhaltung sicher.
• Stellt in Zusammenarbeit mit dem Team, den Vertretern aus den anderen Teams und den Senior-Architects die Integrität des Gesamtsystems sicher
• fokussiert auf schnelles Deployment und größtmögliche Unabhängigkeit der Teams
• stellt die Architekturkonformität im Team sowie die Betriebsfähigkeit des erstellten Produktes in der Zukunft sicher
• nimmt am Architekturboard teil, greift Makro-Architektur-Themen im Team auf und bringt sich in Diskussionen und Entscheidungen ein
• ist Moderator für Architektur-Themen im Team

Fähigkeiten und Erfahrungen:
• stark in der Vermittlung von Architekturthemen
• interessierst an technologischen und architektonischen Entwicklungen und Trends und kann deren Relevanz wirtschaftlich abwägen
• verfügt über sehr gute Koordinations- und Kommunikationsfähigkeiten
• Hat ein gutes fachliches und technisches Verständnis der Komponenten im Team und idealerweise einen guten Überblick über das Gesamtsystem
• Hat sehr gute Erfahrungen in der Entwicklung, Konfiguration und Betrieb von IT

3.7 Die TD-Runde

Das Architekturboard (TD-Runde) traf sich während des Lhotse-Projektes zwei mal in der Woche für eine Stunde. Einberufen und moderiert wurde sie durch einen Stefan X., der als zentral agierender Architekt (etwa wie ein Systemarchitekt) für IT-Sicherheit und Performance übergreifend verantwortlich ist. Die TD-Runde hat ein eigenes Backlog gepflegt. Weitere Teilnehmer waren die Vertreter der Qualitätssicherung und Operations.

Typische Themen der TD-Runde sind:

- Teams sollen ihre Ergebnisse selbst deployen können
- Consumer-Driven-Contract (CDC) Tests zwischen Vertikalen sollen Regressionen verhindern
- Wir möchten im Team Scala als Programmiersprache nutzen, bislang steht aber Java auf der Makro-Architektur. Also möchten wir die Regeln für die Makro-Architektur lockern
- Sicherheits-Awareness Schulung aller Entwickler
- Java8 upgrade steht an. Was tun?

In der TD-Runde konnten die Technical-Designer dafür arbeiten, dass ihre jeweiligen Teams immer unabhängiger und schneller deployen konnten.

Johannes Mainusch: ”ich erinnere mich daran, wie wir im Sommer 2012 die von mir später so benannten Deployment-Wars führten. Damals gab es noch eine zentrale QA-Verantwortung vor dem Live-Stellen der Software. Bei jedem Deployment musste die QA-Abteilung eine Batterie von zentral angelegten Regressionstests (in Selenium) abfahren, um so sicher zu stellen, dass mit dem neuen Software-Stand nichts kaputt ging. Nun hatten wir damals schon 5 vertikal aufgestellte Teams. Auf die Frage, wie lange ein Integrationstest dauern solle, sagte die QA, unter 4 Stunden gehe das nicht. Somit war klar, dass mit dieser Methode maximal 8-10 Deployments pro Woche möglich waren, realistisch eher nur fünf.
Da auf der Integrationsumgebung entweder nur getestet oder nur deployt werden konnte, wurde eine Art Schrankenwärter als Ansprechpartner für die Teams zentral etabliert. Nachdem der Schrankenwärter einige Male kurz vor dem benötigten Deployment überhaupt nicht auffindbar war, wurde schnell klar, dass dieser Prozess so nicht funktionierte.
Schritt für Schritt wurde in den folgenden Monaten die QA-Verantwortung auf die Teams verteilt, der eine Schrankenwärter wurde zu einem Schrankenwärter pro Team, die zentralen Regressionstests wurden auf die Teams verteilt, usw.
Heute im 2016 läuft Otto.de mit etwa 150-180 Deployments pro Woche aus 14 Teams heraus. ”

Dezentralisierung:
Schrittweise Dezentralisierung von Verantwortung beschleunigt die Produktentwicklung.

Durch die TD-Runde und die Technical-Designer entstand bei otto.de ab 2012 schrittweise eine Architektur-Organisation, die neben der klassischen Führungsstruktur parallel existierte. Diese Organisation etablierte die bis heute bestehende Otto-Architektur in Vertikalen.

3.8 Die Otto-Architektur in Vertikalen

Vertikale Architekturen waren für mich 2012 neu. Die Entscheidung, die Teams und Komponenten vertikal zu schneiden beruht auf der Entscheidung, das jedes Team und jede entwickelte Komponente echte Nutzer haben soll. Das heißt, jede Komponente implementiert ein Stück Frontend und jedes Team ist für einen Teil des Einkaufserlebnisses im E-Commerce verantwortlich. Diese Entscheidung war gleichzeitig der Todesstoß für die bislang verwendete monolithische Schichten-Architektur. Konsequenterweise änderte das auch die gesamte Organisation weg von der klassischen Projektorganisation (Ablauforganisation) mit Projekten und Projekt-Ketten hin zu einer PRoduktorganisation mit vertikal-aufgestellten cross-funktionalen Teams.

3.8.1 Warum die klassische IT versagt

Fast immer, wenn über IT-Architekturen gesprochen oder gelehrt wird, tauchen Schichten auf. Im Informatik Studiengang lernen wir das OSI/ISO Schichtenmodell kennen. Im Unternehmen dann die Unterscheidung zwischen Frontend und Backend und früher gern auch 3-Tier Architekturen kennen.

Unweigerlich denken also ausgebildete IT-Experte in Schichten, wobei in den höheren Schichten durch Abstraktion die Komplexität der unten liegenden Schichten verborgen wird.

Hinzu kommt, dass die klassischen IT-Organisationen diese IT-Schichten mit eigenen Abteilungen bedienen. Es gibt Abteilungen, die für Netzwerke, Datenbanken, Operations, Entwicklung et cetera zuständig sind. Skalierung von IT-Vorhaben wird durch Ausbau der Abteilungen und damit respektive der Schichten gemacht. So wächst jede Schicht nur in die Breite (siehe Abbildung 3.5). Im Laufe der Zeit werden zwischen den Schichten dann immer mehr Verbindungen/Abhängigkeiten geschaffen. Es entsteht der Monolith. Kein einzelnes Teil ist mehr zu ändern, ohne dass es weit verzweigte Abhängigkeiten gibt. Kleine Änderungen in einem Teil haben Auswirkungen an ungeahnten Stellen auf der anderen Seite.

Irgendwie erinnert das an den metaphorischen Flügelschlag des Schmetterlings in Brasilien, der einen Tornado in Texas auslöst. Dieser Schmetterlingseffekt[7] ist eine

[7] Schmetterlingseffekt: https://de.wikipedia.org/wiki/Schmetterlingseffekt

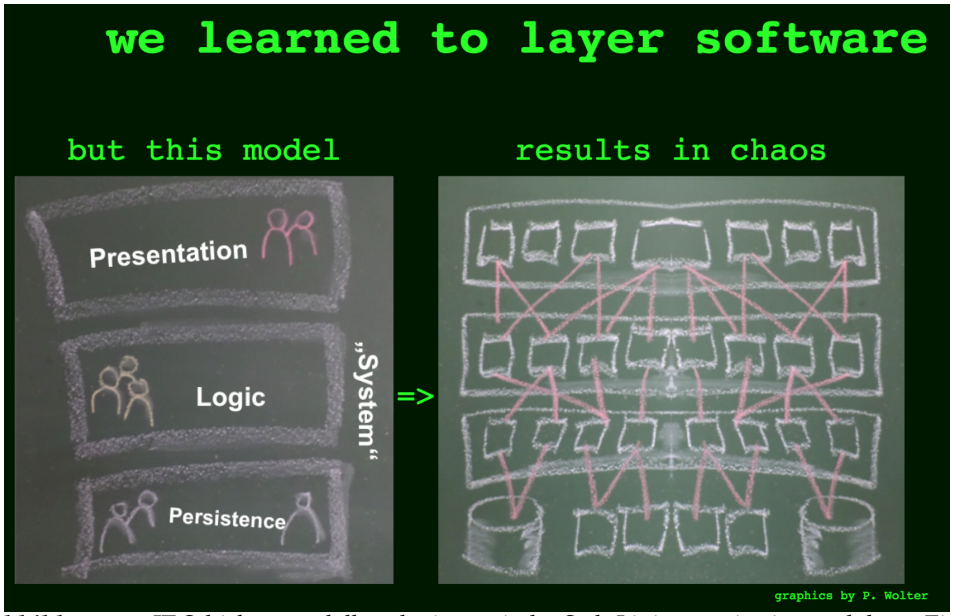


Abbildung 3.5: IT-Schichtenmodell und wir es mit der Stab-Lininorganisation und deren Einheiten korrespondiert.

Metapher, die für chaotische Systeme und Chaostheorie verwendet wird. Und in ähnlicher Weise haben wir in der IT in den letzten 30 Jahren Systeme und Organisationen gebaut, die alle Merkmale von Chaos in sich tragen. Kleinste Änderungen an einer Stelle haben ungeahnte Auswirkungen an anderer Stelle. Oder, wie ein Kollege bei der Lufthansa Technik früher sagte: ”Das System ist wie ein erkalteter Teller Spagetti. Immer wenn wir versuchen, nur einen herauszuziehen, kommen alle anderen mit vom Teller”.

Nun versucht die IT-Architektur die Herrschaft über dieses Chaos wiederzuerlangen. Nur leider mit einem völlig falsch definierten Ziel. Liest man in Wikipedia nach, so steht unter dem Wikipedia Eintrag für IT-Architektur[8] zusammengefasst und etwas pointiert Folgendes:

(klassische) IT-Architektur ist die Kunst alter Männer, Bilder von alter Software zu malen.

Das ist falsch. Wir brauchen eine neue oberste Direktive der IT-Architektur.

Die neue Direktive für IT-Architektur:
Make IT changeable

Und das wiederum bedeutet für IT-Architekten, dass sie zunächst das Deployment im Unternehmen, in den Teams verstehen müssen, und das Deployment entspre-

[8] https://de.wikipedia.org/wiki/IT-Architektur

Abbildung 3.6: Wir brauchen in der IT-Architektur eine neue oberste Direktive: Make IT changeable

chend schnell machen müssen. Mit Deployment ist der Prozess der Live-Stellung neuer Software von der Entwicklungsumgebung bis zum Live-Server gemeint.

Bei Otto.de dauert (Stand 04/2016) das Deployment einiger Teams weniger als 10 Minuten. In einer Woche werden bis zu 180 Deployments auf durchgeführt. Das ist nur durch eine sehr hohe Unabhängigkeit der Teams und durch eine sehr starke Automatisierung aller beteiligten Prozessteile möglich.

Software in Schichten, Stab-Linienorganisationen ohne weitere Kommunikationsstränge und die fehlende visionäre Ausrichtung der IT-Architektur sind drei strukturelle Fehler in den IT-Systemen heutiger Industrieunternehmen. Werden diese nicht systematisch repariert, so steht es schlecht um die Zukunft der betroffenen Unternehmen.

Bei Otto manifestierte sich der Systemfehler in den bis 2012 mehrfachen vergeblichen Versuchen, das bestehende IT-Shopsystem zu erneuern. Weiterhin gab es im Betrieb des alten Systems alle Kennzeichen von Chaos:

- Teure Bugs waren aus dem System nur im 2-Wochen Rhythmus entfernbar.
- Hohe Frustration aller Beteiligten.
- Viel Know-How bei externen Dienstleistern im Gegensatz zur relativ geringen internen Kompetenz.
- Relativ hohe Kündigungsrate.

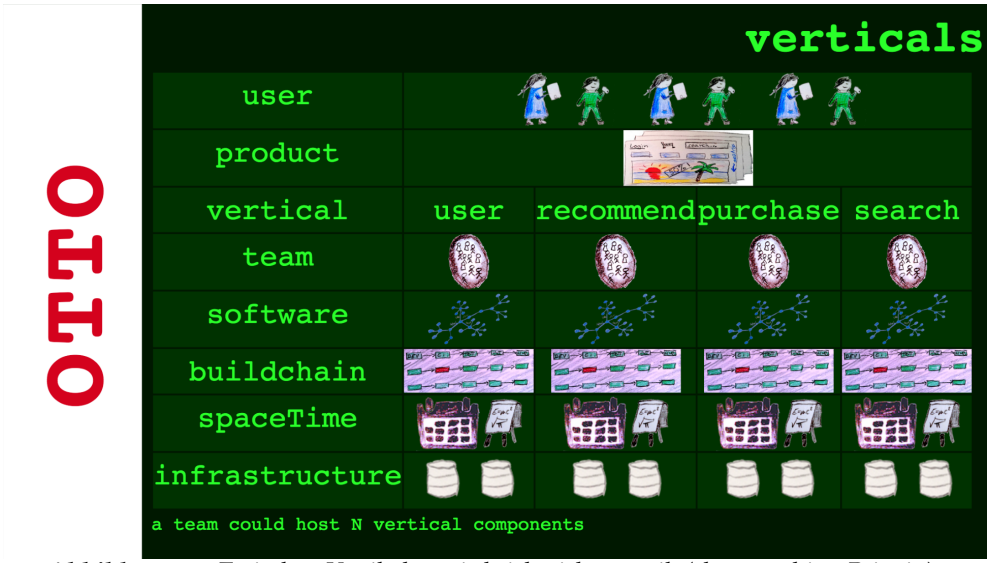


Abbildung 3.7: Zwischen Vertikalen wird sich nichts geteilt (share-nothing-Prinzip).

- Wenig Attraktivität als Arbeitgeber aufgrund der von außen wahrgenommenen Trägheit und Innovationsferne.

Aber Otto hatte auch starke Pluspunkte. Dazu gehörten eine sehr hohe Wertschätzung der Mitarbeiter, eine gute finanzielle Ausstattung, wahrgenommene Sicherheit im Job und von allem der Wille zu Verbesserung der Situation. Das ist erwähnenswert, da Veränderung immer einen fruchtbaren kulturellen Boden im Unternehmen braucht.

3.8.2 Warum vertikale Schnitte helfen

Vertikalisierung der IT-Strukturen ermöglicht die Skalierung von Bereichen in Personalstärken jenseits 30 Mitarbeiter ohne dabei Abhängigkeiten und in der Folge Chaos zu schaffen. Das Ziel vertikaler Strukturen ist dabei immer maximale Unabhängigkeit von Teams und IT-Komponenten.

Dazu gehört unbedingt das Share Nothing Prinzip zwischen vertikalen Teams. Dabei ist jedes Team in seinen Entscheidungen, in seiner Arbeitsweise und mit den erstellten Produkten maximal unabhängig von Nachbarteams. Nur durch diese Unabhängigkeit kann sicher gestellt werden, dass die erstellten Produkte/Artefakte unabhängig voneinander ausgeliefert werden können.

Für die Software bedeutet das, ein Team kann jederzeit ein Release machen und live stellen, ohne andere Teams zu fragen. Das geht natürlich nur, wenn das Team auch sicher ist, dass in anderen Teams durch das eigene Deployment keine Fehler entstehen. Und das wiederum geht nur durch Unabhängigkeit, sehr hohe Automatisierung und eine umfassende Abdeckung mit automatisierten Tests innerhalb der Produkte, die das Team baut.

Vertikale Schnitte ermöglichen die nachhaltige Skalierung von IT-Organisation und IT-Architektur.

Nur durch das gleichzeitige Betrachten von Organisation und IT-Architektur lassen sich große Monolithen kleiner schneiden. Da bislang in Organisationen immer nur entweder die Organisationsstuktur oder der IT-Architektur angefasst wurde, scheiterten diese Ansätze fast immer.

3.8.3 Was eine Vertikale ist

Was nun kennzeichnet eine Vertikale? Folgende Zutaten machen eine Vertikale aus:

- Ein agiles und cross-funktionales Team, in dem möglichst alle Teilnehmer zu 100% ihrer Arbeitszeit zugeordnet sind
- Ein eigener Technologie-Stack. Kein Teilen von Datenbanken oder Servern mit anderen Teams.
- Eigene IT-Infrastruktur, beispielsweise als virtualisierte Server oder AWS-Instanzen.
- Eine eigene Triade, die das Team leitet.
- Ein agiler Entwicklungsprozess und damit die Hoheit über die Ausgestaltung dieses Prozesses.
- Regeln und Gremien zur Zusammenarbeit mit Nachbar-Vertikalen. Beispielsweise die regelmäßig stattfindenden TD-Runden 2xpro Woche.
- Regeln und Gremien zur Zusammenarbeit mit der restlichen Unternehmensstruktur.
- Regeln und Gremien zur Zusammenarbeit mit dem klassischen Management, das diese Freiräume erst Schritt für Schritt schaffen muss und dabei voller Hoffnung, Spannung und Ungewissheit auf diese Vertikalen schaut. Und das im Notfall jederzeit am Steuerruder ist (denn das ist eine der Aufgaben des klassischen Managements, ähnlich der Aufgabe einer Stewardess im Flugzeug. Da wird in der Notsituation dann bitte auch nicht diskutiert, ob man diese Masken nun wirklich über Mund und Nase ziehen muss...).
- generische Vorgehensweise - vertikal organisierte Teams bauen vertikale Produkte und teilen sich bei Wachstum wieder in vertikale Teams.

Eine Vertikale ist also ein eigenständiges Team. Dazu gehören Aspekte von Organisation, Technolgie und Arbeitsmitteln mit dem Ziel, eine große Organisation aus kleinen, möglichst unabhängigen Einheiten zu bauen. Denn kleine schlagkräftige Einheiten sind in chaotischen Verhältnissen überlebensfähiger als große schwerfällige Armeen.

Vertikal organisierte Teams sollten folgende Regel beachten. Die von einer Vertikale erstellten Produkte sollten ebenfalls vertikal gebaut sein. Beispiel: ein Team ver-

antwortet in einem E-Commerce-System einen Warenkorb. So ist das Team für alle Webseiten/Prozessteile verantwortlich, die auf den Klick des Warenkorb-Symbols bis hin zur Bezahlen an der Kasse folgen. Nun soll für den Kunden ein Merkzettel im Shopsystem gebaut werden. Dafür gäbe es ja die Möglichkeit, den Merkzettel als eine Ausprägung des Warenkorbes zu bauen. Das entspräche einem Warenkorb, dem die Bezahlung an der Kasse fehlt. Somit hätte man bestehende Programmteile effizient wiederverwertet. Das hätte dann natürlich den Preis einer etwas höheren Komplexität der Komponente. Das passt so nicht ins Konzept von Vertikalen.

Ein vertikal denkendes Team würde diese Möglichkeit verwerfen und stattdessen ein eigenständiges System für den Merkzettel bauen. Somit würden bestimme Funktionalitäten redundant gebaut werden. Das neue Merkzettel-System hätte seine eigene Datenbank, eigene Server, ein eigenes Repository und eigenes Deployment. Das erscheint in der klassischen Schichtendenke von Architektur zunächst widersinnig, werden hier doch redundante Systeme gebaut. Die Vorteile Überwiegen jedoch.

- Zellteilbarkeit:
  Vertikale Komponenten in einem vertikalen Team ermöglichen die Skalierung von Organisationen durch selbstorganisierte Zellteilung. Wird das vertikale Team zu groß, kann es sich teilen und die vertikalen Komponenten auf die zwei neuen Teams verteilen. Das geht dann sehr leicht ohne den üblichen Know-how-Verlust.
- Resilienz:
  Beim Ausfall von einzelnen Systemen sind die Auswirkungen beschränkt und Fehler schneller auffindbar.
- Organisatorische Skalierbarkeit:
  Durch die Zellteilbarkeit kann eine Organisation so sehr groß werden, ohne dass die üblichen kommunikativen Bruchstellen auftreten.

Voraussetzung hierfür ist, dass die Vertikalisierung zum Bestandteil der Leitplanken und Makro-Architektur der Organisation wird. Wachstum der Organisation kann dann wie bei der Zellteilung in der Biologie erfolgen. Zunächst baut und betreut ein größer werdendes Team mehrere vertikale Komponenten. In der Regel werden sich dann innerhalb des Teams Substrukturen bilden, indem einige Teammitglieder eher an einem Teil der Komponenten arbeitet, die anderen Teammitglieder an anderen Komponenten. Das Team sollte das natürlich völlig autonom entscheiden. Beispiel: Bei Otto.de teilte sich so das **Suchen und bewerten** Team in die Teams **Suchen** und **Bewerten**. Es gab auch wesentlich schwerer teilbare Teams, in der Regel dort, wo die fachliche Domäne nicht teilbar war. Etwa im **Recommendations-Team**. Die Schritte der Teilung können wie folgt aussehen:

1. ein Team, eine Komponente
2. mehrere vertikale Komponenten existieren im Team
3. das Team bekommt schrittweise mehr Mitglieder. Etwa, indem weitere Entwickler ins Team aufgenommen werden. Es seine Arbeit in Subteams zu organisieren.

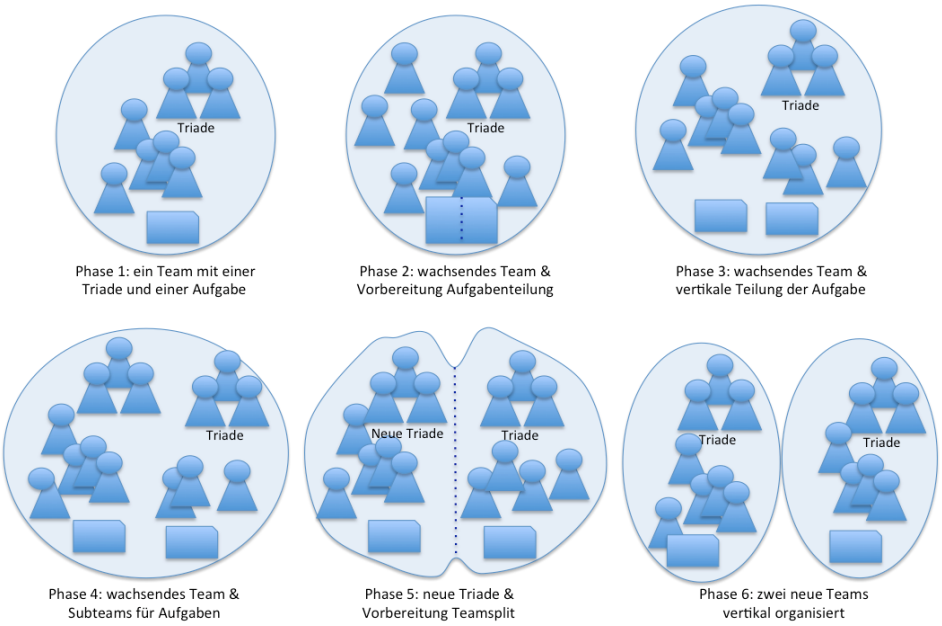


Abbildung 3.8: Phasen den Teilung eines vertikal organisierten Teams in einer skalierbaren Organisation.

4. zur Vorbereitung der Zellteilung werden für die Subteams fachliche Domänennamen gesucht.
5. eine neue Triade wird für das Team etabliert. Aus dem Team heraus oder mit Hilfe von Außen. Das Team wir personell weiter aufgebaut.
6. Zellteilung

Auf Abbildung 3.8 ist diese Form der Zellteilung eines Teams dargestellt.

Vertikalisierung als Teil der Unternehmens-DNA hilft bei der Skalierung von Produktentwicklung. Das Wachstum der Organisation geht über Zellteilung.

3.8.4 Wie vertikale Schnitte gefunden werden können

Bei otto.de wurde der Ansatz gewählt, das Produkt entlang der User-Journey in einzelne Teile zu zerlegen. Es galt das Paradigma: **Jedes Team hat einen Kunden**: Dadurch wurden die Verantwortungsschnitte der Teams so gewählt, dass es kein Backend- und Frontend-Team gibt, sondern dass das Gesamtangebot in kleine am Kundenprozess orientierte Teile zerlegt wird. Auf Abbildung 3.9 ist der gewählte Schnitt skizziert.

So zerfällt ein E-Commerce System beispielsweise in die Teile

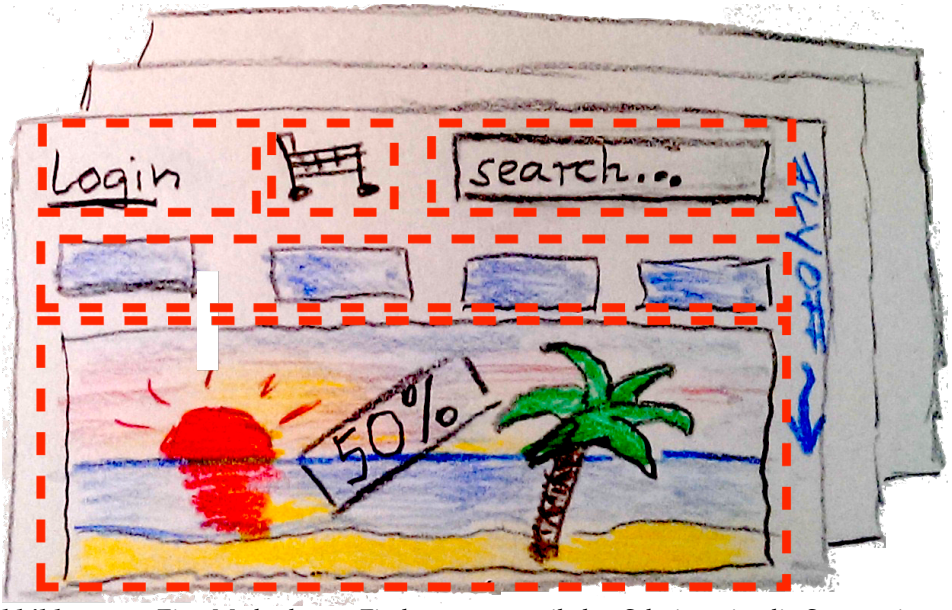


Abbildung 3.9: Eine Methode zur Findung von vertikalen Schnitten ist die Segmentierung entlang der Nutzung der Produkte durch den Kunden. Rot-gestrichelt die Aufteilung des Produktes in Vertikalen aus Benutzersicht.

- User
  Hierunter fällt die Registrierung von Benutzern, das Login und die Authentifizierung für alle Programmteile, die nur im eingeloggten Zustand erreichbar sein sollen. Später könnte hier eine weitere Unterteilung in die eben genannten drei Teile erfolgen. Die so entstehende Vertikale hat die Datenhoheit über alle Benutzerdaten.
- Suche
  Die Suche findet Artikel im Shop. Sie ist eine völlig eigenständige Funktion. Die Suche hat in diesem Beispiel keine Datenhoheit über die Artikeldaten, sondern nur lesenden Zugriff darauf. Aus Performance-Gründen wird die Suche wahrscheinlich über kurz oder lang eine eigene Kopie der Artikeldatenbank aufbauen.
- Artikeldetails
  Die Seite mit den Artikeldetails hat die Datenhoheit über alle Artikel im Shop. Für den Kunden stellt sie diese dar, für andere Konsumenten (wie bspw. die Suche) gibt es eine REST-Schnittstelle.
- Recommendations
  Hier wird das Schaufenster des Shops dargestellt. Das könnte im einfachsten Fall eine statische Seite sein. Oder hier kann eine sehr spezialisierte Recommendation-Engine für jeden Besucher ein persönliches Schaufenster erzeugen.

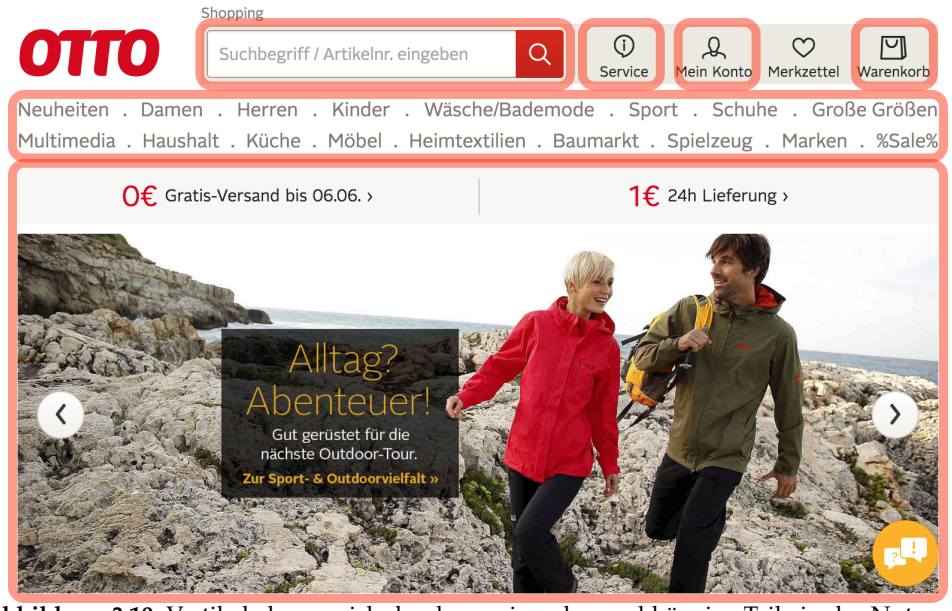


Abbildung 3.10: Vertikale lassen sich durch voneinander unabhängige Teile in der Nutzung eines Produktes finden. Im Bild durch die roten Linien angedeutet.

- Check-out
  Im Check-out wird der Warenkorb und der Prozess bis zum Verkauf implementiert. Im übertragenen Sinne alles vom Einkaufswagen bis hinter die Kasse.
- After-Sales
  After-Sales beinhaltet alle Service-Funktionen nach dem Einkauf. Beispielsweise den Versand, Retouren, Klärungsfälle etc.
- Navigation
  Hier wird die Navigation im Shop implementiert.

Vertikale Schnitte orientieren sich an der Nutzung des Produktes aus Benutzersicht.

So kann die gesamte Domäne in kleinere und unabhängige Teile zerlegt werden. In einigen Fällen wird das so allerdings nur schwer gelingen. Es gibt immer schwer teilbare oder schlichtweg übergreifende Dinge. Bei otto.de wurde beispielsweise eine Pattern-Library gebaut, die alle Interaktionsmuster auf der Webseite für alle Vertikalen enthält. Da alle Vertikale von dieser Pattern-Library abhängen, muss diese eine Stabilität und Unabhängigkeit aufweisen, wie etwa die jQuery-Library oder ähnliche Open-Source-Produkte. Warum Pattern-Libraries trotzdem helfen beschreibt Wolf Brüning in seinem Blog[9] am Beispiel der otto.de Pattern Library.

[9] http://www.produktbezogen.de/bauanleitung-pattern-library-1/

3.9 Makro- und Mikro-Architektur

Makro- und Mikro-Architektur beschreiben die Regeln für den Einsatz und die Verwendung von Technik.

Die Idee zur Unterscheidung von Makro- und Mikro-Architektur wurde bei Otto.de erstmals in 2012 durch Stefan Tillkov eingebracht und ist im Otto-Blog beschrieben[10]. Makro- und Mikro-Architektur beschreiben, welche Architekturregeln übergreifend vorgegeben werden (Makro-Architektur) und welche Dinge im Team entschieden werden können (Mikro-Architektur).

Jede IT-Organisation hat Architekturregeln. Oft werden diese jedoch nicht explizit ausgesprochen und Produktentwicklungsteams müssen dann oft schmerzhaft herausfinden, was möglich ist, und was nicht. Eine explizite Benennung dieser Architekturregeln und die Aufteilung in Makro (übergreifend) und Mikro (von Team entscheidbar) hilft enorm. Außerdem ist das einer der ersten Schritte, um später mehr Entscheidungsfreiheit in die Teams zu geben und somit sukzessive der Gesamtorganisation mehr Handlungskompetenz zu ermöglichen.

3.9.1 Makro-Architektur

Die Makro-Architektur eines Unternehmens sind alle technischen Regeln, die übergreifend für alle Teams gelten. Etwa die Festlegung auf eine Programmiersprache als Standard oder auch eine Datenbank-Technologie.

Technologien, die zur Marko-Architektur gehören, sind sehr aufwändig und teuer, wenn sie geändert werden müssen. Beispielsweise die Entscheidung, Oracle 12c im Unternehmen zum Standard zu erheben, wird dann sehr aufwändig und teuer, wenn der nächste Release-Wechsel kommt. Denn dann wird die gesamte DB-Infrastruktur des Unternehmens angepasst werden müssen.

3.9.2 Mikro-Architektur

Die Mikro-Architektur sind technische und architektonische Regeln, die sich ein einzelnes Team selbst auferlegt. Etwa: wir arbeiten nach BDD oder wir verwenden Spring.

Wenn man das Führungsprinzip, Entscheidungen auf die Kompetenzebene und somit auf die Arbeitsebene zu verlagern, konsequent verfolgt, wird man ebenso versuchen, wenig Makro- und viel Mikro-Architektur zu betreiben. Somit schafft man starke und autonome Teams. Das bedeutet, weniger Unternehmensstandards in der IT fest zu legen.

Hier genau ist ein wesentlicher Unterschied zur klassischen IT, die mit dem Argument von übergreifenden Synergien gerne viele unternehmensweite Standards und Verfahren festlegt. Das kann dann bis hin zu einem starken Regulierungswillen führen. Bei Otto.de gilt:

[10] url https://dev.otto.de/2013/04/14/architekturprinzipien-2/

Dezentralisierung:
Dezentrale Kompetenz ist wichtiger als zentrale Standards.

An dieser Stelle wird oft als Gegenargument eingebracht, dass bei das wechseln von Experten in andere Teams durch Team-individuelle Mikro-Architekturen erschwert wird. Das ist korrekt. Dem kann entgegengesetzt werden, das die Besetzung neuer Teams mit neuen Technologien im Recruiting oft einfacher ist, als ständig mehr Experten mit gleichem Skillset zu finden. Weiterhin ist die Möglichkeit, Neues zu erlernen einer der größten Motivationsfaktoren für Nerds. Nur in einer Organisation, die Vielfalt zulässt, geht das. Bei Otto.de sind die in den letzten Jahren immer größeren technologischen Freiheiten eines der Erfolgskonzepte der heute so schlagkräftigen Organisation.

3.10 Werte und Leitplanken statt Richtlinien und Governance

Leitplanken sind Regeln für Menschen und ihre Zusammenarbeit. Im Lhotse-Projekt wurde von Otto-Management viel Wert darauf gelegt, das große Ziel vorzugeben und weniger den Weg dahin genau vorzuzeichnen. Somit wurde Freiraum auf der Arbeitsebene geschaffen, das Mananement kümmert sich um die Beantwortung der Fragen nach dem ”warum” und dem ”was”. Wie genau Dinge zu erledigen sind, wird den Experten in den Teams überlassen. In den Leitplanken wurde die gemeinsam erarbeiteten Regeln zur Zusammenarbeit festgelegt. Also der Handlungsspielraum der Mitarbeiter und des Managements definiert.

Leitplanken waren im LHOTSE-Projekt Vereinbarungen zu Verhalten Zusammenarbeit. Leitplanken sollen Richtlinien und Governance ersetzen, und waren eine Neuerung zum klassischen Managementverständnis, wie Mitarbeiter geführt werden. Und vor allen Dingen waren die Leitplanken wie eine Magna-Charta der Zusammenarbeit vormals zerstrittener Bereiche.

Leitplanken zwischen IT, Produktmanagement und Projektmanagement könnten beispielsweise die Folgenden sein.

- Entscheidungen über Dinge, die Auswirkungen auf Nachbarbereiche haben, werden gemeinsam getroffen.
- Alle unterstützen nach bester Möglichkeit die vertikalen Teams
- Konsens ist wichtiger als Schnelligkeit.
- Wir haben regelmäßige fachbereichsübergreifende Management-Retrospektiven
- Ich schenke jedem meiner Kollegen 100% Vertrauen.
- Informationen werden immer allen transparent zur Verfügung gestellt.

- Personalentscheidungen über die Besetzung von Triaden werden gemeinsam getroffen

Diese Leitplanken sind Beispiele. Wichtig ist, dass Leitplanken gemeinsam erarbeitet und für gut befunden werden.

3.11 Was macht das klassische Management in der agiler werdenden Organisation?

Agilität kommt häufig als grassroot-Bewegung in ein Unternehmen, in dem zunächst eines und später mehrere Teams anfangen agil zu arbeiten. Das geht oft so lange gut, bis entweder die agilen Teams an den Glasdeckel des inzwischen verunsicherten Managements stoßen, oder durch zahlreiche schnell gewachsene IT-Architekturen neue und komplexe IT-Architekturen im Umfeld alter Systeme entstehen.

Die Aufgabe des klassischen Managements sollte es sein, die Rahmenbedingungen für eine agile Transition eines Unternehmens abzustecken und für die entstehenden Teams eine optimale Arbeitsumgebung zu schaffen. Besonders im IT-Management muss ein starkes Augenmerk auf die Organisation der Architektur gelegt werden.

Johannes Mainusch: „Als IT-Manager hatte ich wenig mit der operativen Steuerung der agilen Teams zu tun.“. Die klassische Management Rolle als Team- oder Abteilungsleiter lies daher viel Zeit, sich unterstützend um folgende Themen zu kümmern:

- Recruiting
- Personalmarketing
- Aufbau und Unterstützung einer Architekturorganisation (TDs, TD-Runde, Mikrpo- und Makro-Architektur)
- Beschaffung externer Experten
- Budgetsteuerung
- Innovationen (strategische Ausrichtung, Web-Performance Monitoring, Pair-Programming-Coach, Hackathons, Offsites, Konferenzen, Partys, ...)
- langsames De-Staffing der alten Plattform
- Verhandlungsführung bei Umstaffing von Experten in neue Teams
- Initialisierung neuer Teams
- Beziehungsmanagement mit Nachbarabteilungen
- generelle Lückenfüllerei, z.B. die Initiierung eines Produktteams für mobile, oder ein Assessment der Sinnhaftigkeit des Einsatzes bestimmter Werkzeuge. Stellvertretungen in Urlaubssituationen, etc.

Anstatt von einem Engpass oder Notfall zum nächsten zu Laufen, konnte sich das Management mit der so frei gewordenen Zeit um seine Kernaufgabe kümmern:

Die oberste Direktive von Management:
Den Erfolg des Unternehmens und seiner Mitarbeiter in der Zukunft sichern.

Und damit gilt eben nicht, heute die eigene Position durch Mikromanagement zu betonieren. In klassisch aufgestellten IT-Organisationen werden die Manager hingegen häufig zum Kommunikationsengpass. Und das trotz teils sehr hohem Arbeitseinsatz von weit mehr als 50h/Woche. Oft sind so agierende Manager in klassischen IT-Organisationen durch vielschichtige kommunikative Aufgaben und zugestopfte Kalender operativ so fest eingebunden, dass für die Hege des Erfolgs in der Zukunft keinerlei Zeit und Energie übrig bleibt.

3.12 Scrum@Otto: 100 Sprints später

Inzwischen hat Otto laut Aussage von Peter Wolter, Bereichsleiter IT im E-Commerce die ehemalige Projektorganisation in eine Produktorganisation gewandelt. Dort wo 2012 mit vier Teams gestartet wurde, arbeiten heute 17 Teams an der Weiterentwicklung von otto.de. Jedes Team vertritt einen Teil des angebotenen Produktes und hat somit direkten Kundenkontakt[11].

Dort wo früher ein sehr ausgefeiltes Multiprojektmanagement eine festgefahrene IT-Situation mit einem langsamen und in die Jahre gekommenen Shopsystem managte, arbeiten heute 17 Teams an einem der schnellsten und größten E-Commerce Systeme Europas. Und das wurde komplett selbst entwickelt. Auch die Arbeitsweise der Teams hat sich in dieser Zeit verändert.

Deutlich sichtbar wird die Veränderung der Arbeitsweise der agilen Teams durch den Vergleich der Team-Kalender 2012 und 100 Sprints später im Jahr 2015. Auf Abbildung 3.12 wird der zweiwöchige Rhythmus der vier Start-Teams von Otto.de im Jahr 2012 dargestellt. Die Teams begannen mit Scrum im zweiwöchigen Sprint-Rhythmus. Der Sprint startet Donnerstags mit Sprint-Planning 1 und 2. Insgesamt gab es zum Start des Projektes 2012 folgende Meetings:

- Sprint-Planning 1 und 2 flankiert von zusätzlichen Scrum of Scrums[12].
- Täglich um 10:00 ein Standup-Meeting.
- Das tägliche Scrum of Scrum Meeting (SOS) um 10:30.
- An den beiden Dienstagen ist vormittags ein Estimation-Meeting.

[11] derzeit gibt es hier ein Team, dass eine Ausnahme bildet und einen klassischen Service erbringt.

[12] Scrum of Scrums (SOS) Meetings sind eine Methode zur Koordination und der Beseitigung von Arbeitshindernissen zwischen Scrum-Teams in einer größeren agilen Organisation. Dazu entsendet jedes Scrum-Team einen Botschafter in ein SOS-Meeting. Oft findet das SOS-Meeting nach dem täglichen Standup der Scrum-Teams statt.

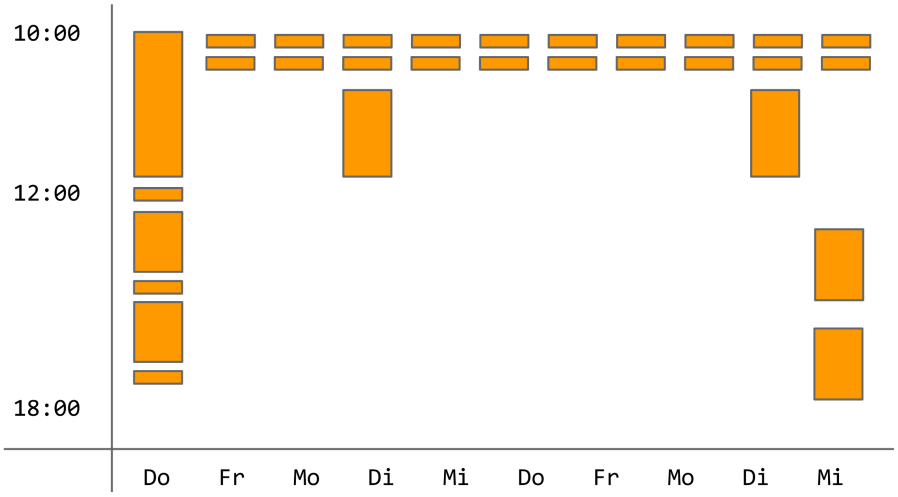


Abbildung 3.11: der Scrum-Teamkalender im Lhotse Projekt zu Projektbeginn.

- Am Mittwoch am Sprintende (alle zwei Wochen) erst das Scrum-Review, dann die Retrospektive.

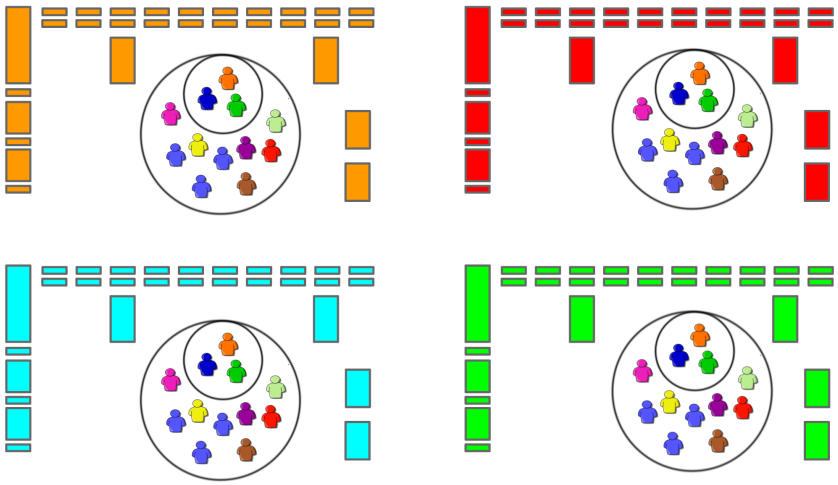
Damit kamen die Teams in Summe auf ca. 18 bis 20 Stunden Meetings pro Sprint. Das wurde besonders bei den Entwicklern oft als sehr hoch empfunden. 20 Stunden entspricht 25% der im Sprint zur Verfügung stehenden Arbeitszeit. Dazu kamen dann noch alle außerplanmäßigen Treffen und Veranstaltungen (Abteilungstreffen, Offsites, Recruiting-Hilfe, Interviews mit externer Verstärkung, Fortbildungen,...).

Im Laufe der Jahre veränderten die meisten der Teams ihre Arbeitsweise Schritt für Schritt. Aus den Retrospektiven und in Abstimmung mit dem Management und den umgebenden Teams kamen verschiedene Änderungsvorschläge, die nach und nach teamindividuell umgesetzt wurden. In Abbildung 3.12 ist diese Veränderung anhand eines Vergleiches von vier Teams dargestellt. Nur eines der Teams (im Bild oben links) hat den ursprünglichen Scrum-Prozess nahezu unverändert beibehalten. Eines der Teams (unten rechts) arbeitet 2015 in einem Prozess, der nur die folgenden Meetings aus dem urspünglichen Scrum-Prozess beibehalten hat:

- Täglich ein Standup-Meeting.
- Plannings, wann immer nötig und kurz.
- Die Retrospektive.

In Summe kommt dieses Team (2015) nur noch auf ca. 11 Stunden Meetings pro Sprint (etwa 15%). Es zeigt sich, dass die Meetings **Daily-Standup**, **Planning** und **Retrospektive** in agilen Prozessen auch bei Methodenwechsel erhalten bleiben. Ihnen kommt daher eine besondere Bedeutung zu.

Teamkalender Otto-Scrum im Jahr 2012



Teamkalender Otto-Scrum im Jahr 2015 (meist kein Scrum mehr...)

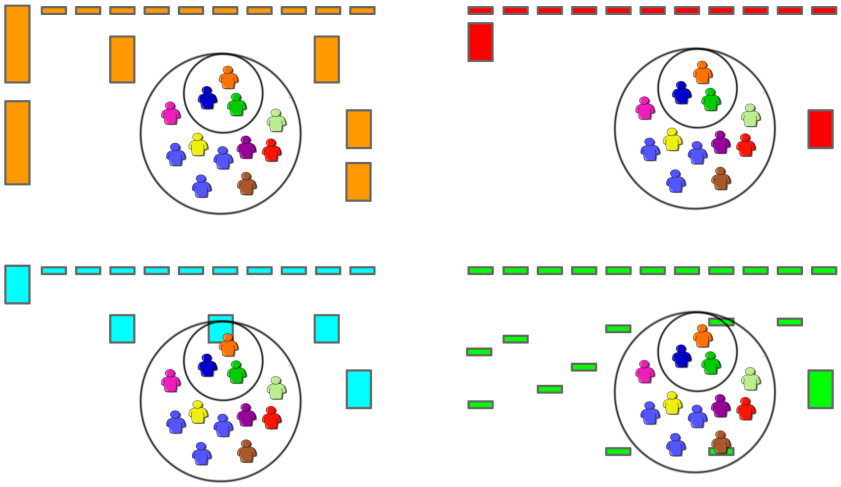


Abbildung 3.12: der Team-Kalender der vier Lhotse-Teams im Jahr 2012 und 100 Sprints später im Jahr 2015. In 2012 starteten alle Teams mit dem gleichen Scrum-Teamkalender, für 2015 sind nur vier Teamkalender der damals schon 12 Teams abgebildet. Die Bilder wurden von Christian Stamm zur Verfügung gestellt und kommen aus dem Vortrag ”100 Sprints Otto.de”

3.13 Fazit

Otto hat es geschafft, vom Katalogversender zu einem Major-E-Commerce Player zu werden. Damit ist Otto dort erfolgreich, wo Quelle oder Neckermann gescheitert sind. Allein das ist bemerkenswert, ein internationaler Konzern mit ca. 50.000 Mitarbeitern überlebt einen Paradigmenwechsel seines Geschäftsmodells.

Weiterhin hat Otto mit dem Lhotse-Projekt, einem IT-Großprojekt, Erfolg gehabt. Lhotse und das begleitende Infrastrukturprojekt zur Virtualisierung der IT liefen über zwei Jahre und hatten jeweils ein Budget im zweistelligen Millionen-Euro-Bereich. Auch das schafften in den letzten Jahren im IT-Bereich nur wenige. Die meisten IT-Projekte in dieser Größenordnung scheitern.

Bemerkenswert ist weiterhin die Kununu-Bewertung von Otto. Bei Kununu werden Arbeitgeber aus Sicht von Mitarbeitern bewertet. Otto steht hier mit 4 (Stand Mai 2016) sehr gut da. Damit ist Otto ein attraktiver Arbeitgeber in Hamburg, einer Region, in der die Rekrutierung von IT-Experten ein harter Job ist.

Auch die Bewertung des otto.de shops mit einem Alexa-Ranking von Platz 23 in Deutschland ist sehr gut (2012 war das etwa Platz 50). Auf Alexa werden Internet-Plattformen hinsichtlich ihrer Nutzung verglichen. Zalando.de wurde von Otto seit ihrer "Schrei vor Glück" Marketing Kampagne als starke Konkurrenz wahrgenommen und hat im Mai 2016 den weit zurückliegenden Platz 54.

Am wichtigsten jedoch erscheint es, das Otto es mit dem Lhotse-Projekt und der nachfolgend gewachsenen Organisation um otto.de geschafft hat, in Deutschland ein herausragendes Beispiel einer agilen Transformation eines etwas verstaubten E-Commerce-Produktes zu zeigen. Heute gilt Otto damit als ein Musterunternehmen, das für andere Branchen mit ähnlichen Herausforderungen Modellcharakter hat.